9

QUATRE ÉCRITURES DE *PIANO PHASES*

Nous proposons d'illustrer les fonctionnalités du langage à travers quatre programmes correspondant tous à la même pièce : *Piano Phase* du compositeur Steve Reich, initialement prévue pour deux pianos. Cette pièce est construite sur la répétition d'une séquence de douze notes, jouée par les musiciens M_1 et M_2 en parallèle, où M_2 se déphase par rapport à M_1. Le processus compositionnel peut être décrit algorithmiquement de la façon suivante en adéquation avec la partition originale (figure 36) : Pendant les n_p premières périodes ($12 \leqslant n_p \leqslant 18$), M_1 et M_2 jouent les mêmes notes au même moment. L'étape de déphasage peut alors commencer : M_2 accélère légèrement, se déphase par rapport à M_1 jusqu'à ce que le déphasage atteigne après n_d périodes ($4 \leqslant n_d \leqslant 16$), une valeur équivalente à une double-croche. À ce moment, les deux voix se resynchronisent et M_2 reprend le tempo initial (celui de M_1). La superposition des notes formant une nouvelle couleur musicale pendant n_p périodes. Ces deux étapes sont ensuite répétées douze fois, exécutant les douze superpositions possibles. La pièce se finit sur la première étape à l'unisson. Dans tous nos exemples, nous prendrons $n_p = n_d = 4$.

Le premier programme est autonome et extensionnel. Le deuxième programme est équivalent au précédent mais il est cette fois-ci intentionnel. La troisième et quatrième versions sont interactives. La troisième utilise un enregistrement pour séquencer la voix qui se déphase tandis que la quatrième réinjecte la propre performance du musicien pour créer le déphasage.

9.1 CONTRÔLE DYNAMIQUE DES DÉLAIS

Dans la première version présentée ci-dessous, deux boucles électroniques remplacent les deux musiciens. Chacune d'elles correspond à une voix qui séquence des messages envoyés à un synthétiseur. Les actions jouées par le synthétiseur sont définies dans le corps de boucle avec une période de trois pulsations pour la première et une

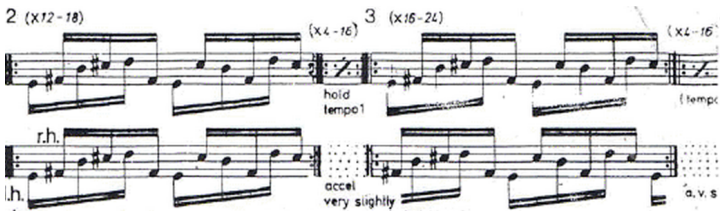

FIGURE 36: Extrait d'une partition de *Piano Phase* du compositeur Steve Reich

période définie par `$period` pour LoopDec, la seconde. Cette dernière variable est initialisée à trois pulsations, la durée d'une mesure. Les délais entre chaque action de la première boucle valent $1/4$ de pulsation ; pour la deuxième boucle, ils sont spécifiés par la variable `$delay` (initialisée à une double-croche). La variable `$cpt` permet de repérer le moment où l'on passe d'une étape de déphasage à une étape en phase. Le compteur est incrémenté dans la boucle. Une fois que la valeur 4 est atteinte, on diminue la période et les délais. Lorsque le compteur est égal à 0 modulo $(4+4)$, la période et les délais sont remis à leurs valeurs initiales.

La fonction `beat2ms` est une fonction auxiliaire qui permet de convertir des pulsations en secondes pour transmettre la durée d'une note au synthétiseur. `mnote1` et `mnote2` sont le noms des receveurs correspondant au deux synthétiseurs.

```
@fun_def beat2ms($X) {
        1000*$X*60/$RT_TEMPO
}
   $periode := 3
   $delai := 1/4
   $compteur := 0
   antescofo::tempo 100

   loop LoopRef 3 {
      mnote1 64 120 @beat2ms(1/4)
      1/4 mnote1 66 120 @beat2ms(1/4)
      1/4 mnote1 71 120 @beat2ms(1/4)
      1/4 mnote1 73 120 @beat2ms(1/4)
      1/4 mnote1 74 120 @beat2ms(1/4)
      1/4 mnote1 66 120 @beat2ms(1/4)
      1/4 mnote1 64 120 @beat2ms(1/4)
      1/4 mnote1 73 120 @beat2ms(1/4)
      1/4 mnote1 71 120 @beat2ms(1/4)
      1/4 mnote1 66 120 @beat2ms(1/4)
      1/4 mnote1 74 120 @beat2ms(1/4)
      1/4 mnote1 73 120 @beat2ms(1/4)
}

loop LoopDec $periode {
   mnote2 64 120 @beat2ms(1/4)
   $delai mnote2 66 120 @beat2ms($delai)
   $delai mnote2 71 120 @beat2ms($delai)
   $delai mnote2 73 120 @beat2ms($delai)
   $delai mnote2 74 120 @beat2ms($delai)
   $delai mnote2 66 120 @beat2ms($delai)
   $delai mnote2 64 120 @beat2ms($delai)
   $delai mnote2 73 120 @beat2ms($delai)
   $delai mnote2 71 120 @beat2ms($delai)
   $delai mnote2 66 120 @beat2ms($delai)
   $delai mnote2 74 120 @beat2ms($delai)
   $delai mnote2 73 120 @beat2ms($delai)
```

```
   $compteur := ($compteur+1) % 8

   if ($compteur = 4) {
      $periode := $periode-(1/4)/4
      $delai := $periode / 12
   }
   if ($compteur = 0) {
      $periode := $periode+ (1/4)/4
      $delai := $periode / 12
   }
}
```

9.2 CONTRÔLE DE DYNAMIQUE DU TEMPO + PROCESSUS

Le deuxième exemple est équivalent au premier. Cette fois-ci on a défini un processus récursif qui va jouer une note puis se rappeler lui-même après un certain délai pour jouer la note suivante. Les notes à jouer sont stockées dans les tableau `$notes`.

Le déphasage est contrôlé à travers les variations de tempo du groupe englobant l'appel du processus (liaison dynamique des paramètres temporels).

Le changement de tempo est réalisé à l'aide d'un `whenever` qui « écoute » la variable globale `$cmpt`, modifiée par les processus correspondant à la deuxième voix.

La valeur de la variable `$i` (indice courant du tableau), locale à chaque voix, est accessible en liaison dynamique grâce à la commande `$MYSELF` permettant de récupérer un pointeur sur la coroutine courante.

La fonction `@command` permet d'évaluer une expression pour définir le nom du receveur.

```
@proc_def ::rec_proc($ind) {
   @command("mnote"+$ind) ($notes[$MYSELF.$i]) 120 (1000*1/4*60/$
       local_tempo)
   $MYSELF.$i := ($MYSELF.$i + 1)%12
   if($ind = 2){
        $cmpt := ($cmpt + 1) % 96
   }
   1/4 ::rec_proc($ind)
}

$notes := [64,66,71,73,74,66,64,73,71,66,74,73]
$cmpt:=-1
$tempo1:=100
$tempo2:=$tempo1

group @tempo = $tempo1 {
        @local $i
        $i := 0
```

```
        ::rec_proc(1)
}

group @tempo = $tempo2 {
        @local $i
        $i := 0
        $proc2 := ::rec_proc(2)
}

whenever($cmpt = 48) { //dephasage
        $tempo2 := $tempo1 * (3*4+1/4)/(3*4)
}
whenever($cmpt = 0) {  //en phase
        $tempo2 := $tempo1
}
```

9.3 AVEC SUIVI DE PARTITION ET CONTRÔLE CONTINU D'UN FLUX AUDIO

Dans le troisième exemple on synchronise un accompagnement avec le jeu d'un musicien. Le musicien joue la voix de référence tandis que l'électronique joue la voix qui se décale. Le jeu du musicien est suivi par la machine d'écoute qui détecte les notes jouées. On utilise le vocodeur de phase *SuperVPScrub~* pour contrôler la position d'un fichier audio dans lequel la boucle de 3 pulsations a été préalablement enregistrée. On déphase le fichier audio par rapport au musicien en contrôlant la période comme précédemment.

```
   $periode := 3
   $compteur := 0

NOTE 64 0.25 begin
   loop l $periode @target [6s]
   {
      ScrubPos 0.
      abort c
      curve c
         @Grain := 0.05 s,
         @Action := ScrubPos ($x * 1000) (0.05 * 1000)
      {
         $x
         { { 0.} $periode { 1.4912 } }
      }
      $compteur := ($compteur+1) % 8

      if($compteur = 4)
      {
         $periode := $periode - (1/4)/4
      }

      if($compteur = 0)
```

```
      {
         $periode := $periode + (1/4)/4
      }
   }
NOTE 66 0.25
NOTE 71 0.25
NOTE 73 0.25
NOTE 74 0.25
NOTE 66 0.25
NOTE 64 0.25
NOTE 73 0.25
NOTE 71 0.25
NOTE 66 0.25
NOTE 74 0.25
NOTE 73 0.25 @jump begin
```

9.4 AVEC SUIVI D'UNE PULSATION ET RÉINJECTION

Le dernier exemple est également une situation où le musicien joue la partie de référence et *Antescofo* contrôle un processus correspondant à la voix qui se déphase. La synchronisation n'est cependant pas réalisée grâce à un suivi des événements du musicien mais à partir d'une information de pulsation envoyée depuis l'environnement extérieur (pédale, détecteur de beat). Cette information est récupérée via les mises à jour de la variable `$var_sync`. De plus, l'accompagnement correspond ici à une réinjection de la performance du musicien enregistré en temps réel. Comme précédemment, une curve envoie toutes les 0.05 seconde la position du fichier qu'il faut lire au vocodeur de phase. Cet accompagnement démarre avec 3 pulsations de retard par rapport au temps-réel et rattrape le temps-réel dans les périodes de déphasage. Le déphasage est réalisé en diminuant la valeur de `$puls`, accélérant ainsi la lecture d'une pulsation.

```
$compteur:=0
@tempovar $var_sync(100,1)
$tab := []
$ind_ref := -1
$ind_dec := - 1

WHENEVER($var_sync)
{
   $compteur := $compteur + 1
   if($compteur = 1)
   {
        $start := $NOW
                start_recording 1
   }
   $ind_ref := ($ind_ref +1) % 6

   $tab[$ind_ref] := $NOW - $start
```

```
    if($compteur = 3)
    {
         loop l $puls @target [4s] @sync $var_sync
         {

            $ind_dec := ($ind_ref +1) % 6
            if(($compteur%24) = 12 )
            {
                 $puls := ((1/4)/4)/3
            }
            if(($compteur%24) = 0 )
            {
                 $puls := 1
            }
            abort c
            curve c @Grain := 0.05 s , @Action := ScrubPos ($x *
                1000) (0.05 * 1000)
            {
                 $x
                 {
                                 { ($tab[$ind_dec])}
                         $puls { ($tab[($ind_dec+1)%6]) }
                 }
            }
         }
  }
}
```