

Vers la Modélisation des Pensées Musicales : Le cas du « Tintinabuli » d'Arvo Pärt

DE PAIVA SANTANA, Charles¹
BRESSION, Jean¹

¹ UMR 9912 Sciences et technologies de la musique et du son, IRCAM-CNRS-UPMC

Introduction

Les compositions musicales peuvent être considérées comme des systèmes complexes et autonomes, et ainsi faire l'objet de modélisations informatiques avancées. Nous proposons de développer cette idée en réalisant des modèles formels permettant l'étude de pièces du répertoire musical contemporain, et de concevoir les outils informatiques correspondants. Dans le cas de ce travail nous avons choisi le style « tintinabuli » du compositeur estonien Arvo Pärt (fig. 1) et plus précisément la pièce « *Spiegel im Spiegel* » comme point de départ. Nous nous appuyons notamment sur des notions inspirées de la programmation orientée objet afin de modéliser, produire et évaluer différentes instances et configurations de tels systèmes..

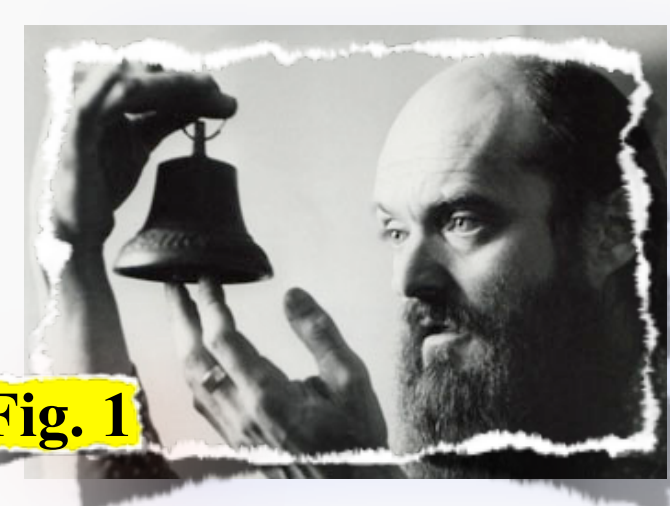


Fig. 1

Le style « tintinabuli »

Le style « tintinabuli » a été utilisé pour la première fois dans la pièce *Für Alina* en 1976. Il est basé principalement sur la notion de triade, c.-à-d. : un ensemble de trois notes séparés par des intervalles réguliers de l'échelle diatonique, par ex. : ut, mi, sol. Dans *Für Alina*, tandis qu'une voix monte et descend librement la plupart du temps par des petits intervalles, l'autre voix est limitée aux trois notes de la triade. Dans la figure 2 nous pouvons voir comment cette technique marche sur une partition. Dans la figure 3 on voit l'implémentation dans quelques lignes de code Lisp.



Fig. 2

```
(lambda (list triade dir pos)
  (let ((direction dir))
    (loop for note in list
      do (when (equal direction 'up)
          (collect (let* ((find-from-end (equal direction 'down))
                        (find-from-end (equal direction 'up))
                        (octave (+ 1200 (floor note 1200)))
                        (degrees (loop for n in triade collect (mod n 1200))
                                (degrees (+ octave (nth (nth-closest-degree-position (list closest-degree-position (1 pos))))))
                        (closest-degree (find (mod note 1200) degrees)
                                (loop find-from-end
                                  (closest-degree-position (position closest-degree degrees)
                                    (nth-closest-degree-position (apply octave-shift-up (list closest-degree-position (1 pos))))))
                        (print (list note degrees closest-degree))
                        (list note (+ octave (nth (nth-closest-degree-position degrees))))
                    ))
    ))
```

Fig. 3

La pièce « Spiegel im Spiegel »

La pièce *Spiegel im Spiegel*, pour instrument mélodique et piano, a été composée en 1978 et démontre un style de composition que malgré sa simplicité, il est rigoureux dans sa structure. Quelques éléments du style tintinabuli sont évidents comme la voix qui se déplace par degrés conjoints et l'utilisation prédominante de triades.

Formalisation

L'analyse de la pièce révèle l'utilisation d'un algorithme pour la construction des intervalles de la voix soliste en même temps qu'elle est accompagné par le piano. Dans la figure 4 nous voyons la progression des deux séries, l'une est le réflexe de l'autre et toutes les deux voix ont des parties symétriques, ce qui justifie le titre « miroir dans un miroir ».

Dans la figure 5 nous avons un résumé de la formalisation de l'accompagnement, chaque triade ayant trois positions ([1 5 6], [1 4 6] et [1 3 6]) qui sont choisies selon son degré dans l'échelle (premier, second, etc.)..

```
0
-1 0, +1 0,
-2 -1 0, +2 +1 0,
-1 -2 -3 0, +1 +2 +3 0,
-4 -3 -2 -1 0, +4 +3 +2 +1 0,
-1 -2 -3 -4 -5 0, +1 +2 +3 +4 +5 0,
-6 -5 -4 -3 -2 -1 0, +6 +5 +4 +3 +2 +1 0,
-1 -2 -3 -4 -5 -6 -7 0, +1 +2 +3 +4 +5 +6 +7 0,
-8 -7 -6 -5 -4 -3 -2 -1 0, +8 +7 +6 +5 +4 +3 +2 +1 0,
0.
if n = 0, then ldur = 5, else ldur = 2.
```

Fig. 4

```
(1 5 6) (1 4 6) (1 5 6) (1 4 6) (1 5 6) (1 4 6) (1 5 6) (1 4 6) (1 5 6) (1 4 6) (1 5 6) (1 4 6) (1 5 6) (1 4 6)
last run
("Eb3" ("F3" ("G3" ("Ab3" ("Bb3" ("C4" ("Db4" ("Eb4" ("F4" // ("G4" ("F4" ("Eb4" ("Db4" ("C4" ("Bb4" ("A4" ("G4" ("F4"))
((-8 -7 -6 -5 -4 -3 -2 -1 0 8 7 6 5 4 3 2 1 0)
(7 1 2 3 4 5 6 7 1 2 1 7 6 5 4 3 2 1)
(2 3 4 5 6 7 1 2 3 // 4 3 2 1 7 6 5 4 3)

1 (8
2 4 6 (9
3 5 7 (10
```

Fig. 5

Modèle

Le modèle a été construit en utilisant une structure hiérarchique où chaque instrument est représenté comme un objet, une classe qui hérite des attributs d'une classe supérieure commune à tous les objets (fig. 6). En même temps, il a fallu créer des méthodes qui donnent un traitement approprié à chaque attribut de chaque classe (fig. 7). Nous résumons ces attributs, qui peuvent être considérés comme des paramètres, à l'essentiel dans ce modèle. Chaque sous-classe a été attaché à un éditeur capable d'afficher des partitions musicales par le DSL « OpenMusic », développé et distribué par l'IRCAM (fig. 8 et 9).

```
(defclass! spiegel-class ()
  ((tonic :accessor tonic
    :initform (set-tonalite (make-instance 'chord :midic '(7400))
                                (make-instance 'tonalite :tonalite 7000 :tonnote 'si :tonalt 'benal))
    :initarg :tonic)
  (spiegel-i :accessor spiegel-i :initform '8 :initarg :spiegel-i)
  (time-signature :accessor time-signature :initform '6 4 :initarg :time-signature)
  (internal-voice :accessor internal-voice :initform nil))

(defclass! spiegel-melody (spiegel-class) ())
(defclass! spiegel-piano (spiegel-class) ())
```

Fig. 6

```
(defmethod spiegel-melody-to-voice ((self spiegel-melody))
  (spiegel-melody-score (tonic self) (spiegel-i self) (time-signature self)))

(defun spiegel-melody-score (fund spiegel-i time-signature)
  (let* ((series (in-spiegel-ser spiegel-i))
        (pitches (transp-by-step fund series))
        (tree (spiegel-melody-rhythm series time-signature)))
    (make-instance 'voice :tree tree :chords pitches)))
```

Fig. 7

Conclusions

A partir de ce modèle et par « l'instanciation » de classes particulières, il est possible d'obtenir différentes retranscriptions de la pièce, ou d'en modifier des attributs structurels. Il est important de souligner que ces manipulations ne sont nullement des détournements ni des transformations de la pièce au sens strict, mais avant tout des modifications de paramètres liés au système compositionnel sous-jacent.

En analysant les différentes « instances » d'une pièce sous la perspective de ce modèle, et en évaluant par là-même la puissance de son système compositionnel, nous avançons sur des terrains musicologiques jusqu'alors inexplorés. Nous proposons ainsi une analyse musicale qui prenne en considération non seulement les pièces musicales dans leur aspect « achevé », les théories sous-jacentes utilisées ou développées par les compositeurs, mais surtout qui les explore en tant que systèmes autonomes, ouverts et indépendants.

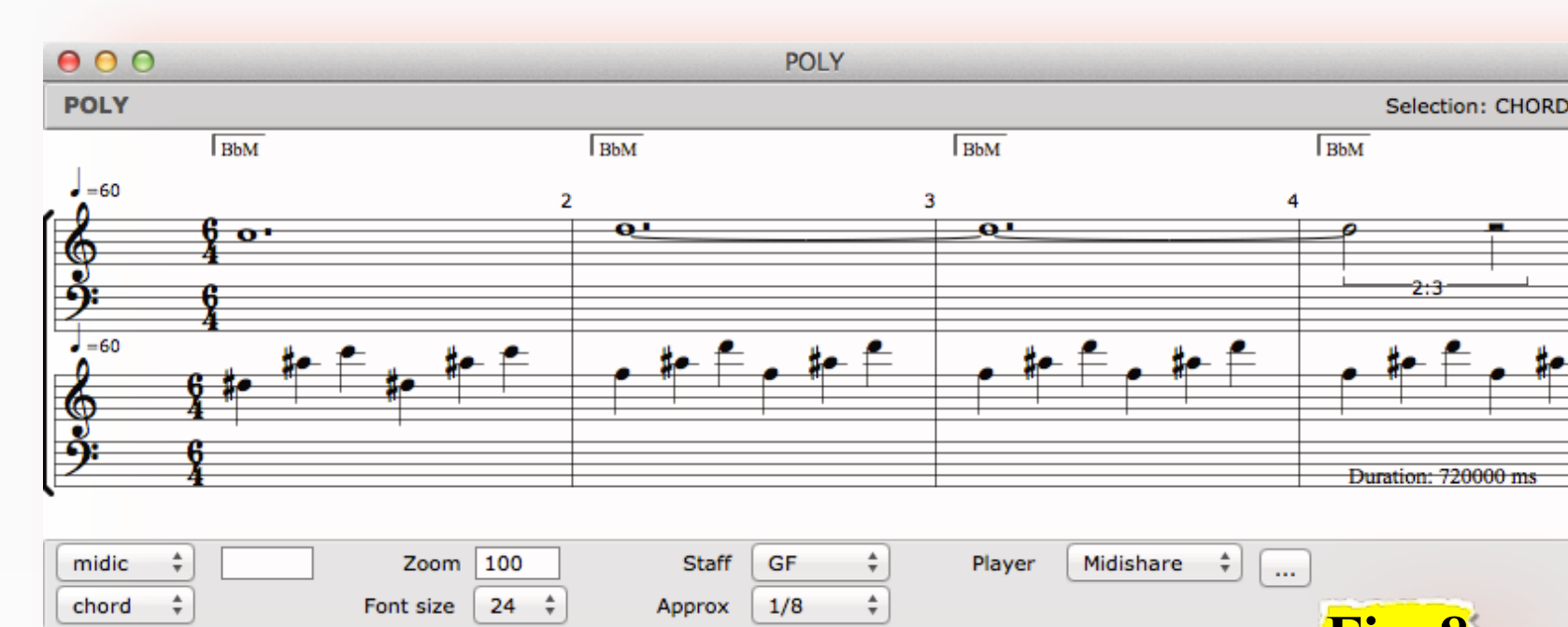


Fig. 8

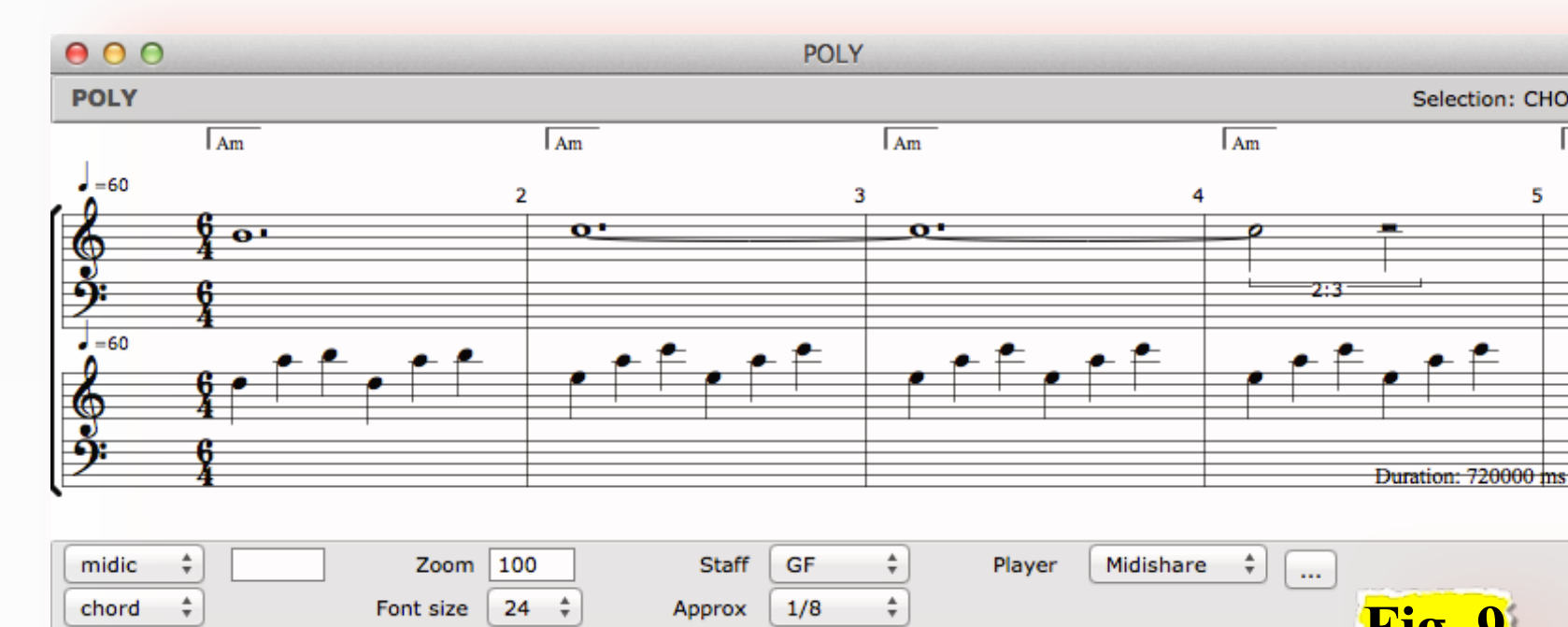


Fig. 9

Bibliographie

AGON, C. (1998). **OpenMusic : Un langage visuel pour la composition musicale assistée par ordinateur**. Thèse de doctorat, Université Pierre et Marie Curie (Paris 6), Paris, France.

HILLIER P. (1997), *Oxford Studies of Composers - Arvo Pärt*, Oxford University Press, New York

RIOTTE, A., MESNAGE, M. (2006). **Formalismes et modèles musicaux** (2 volumes), Editions Delatour,