Visual Programming and Music Score Generation with OpenMusic

Jean Bresson and Carlos Agon
STMS: IRCAM-CNRS-UPMC
Paris, France
{bresson,agon}@ircam.fr

Abstract—We present score programming features in the visual programming and computer-aided composition environment OpenMusic. The sheet object allows to build complex scores and fill or modify their contents algorithmically using visual programs.

I. INTRODUCTION

OpenMusic is a visual programming environment dedicated to computer-aided composition [1], [2]. The basic purpose of this environment is to provide music composers with means to design processes and develop musical projects using computers with the expressive power of a programming language.

Initially, OpenMusic visual programming tools are used to generate or process musical objects such as scores or other symbolic data. Our recent developments extended this approach by allowing building scores containing their own internal functional or algorithmic structure under the form of in-built visual programs.

II. A VISUAL PROGRAMMING LANGUAGE FOR MUSIC

The main working environment for OpenMusic users is called a “patch”. A patch is a visual program where functional components represented by boxes are connected together to form a graph, eventually evaluated at some points in order to trigger calculations and output musical structures or other data (see Fig.1).

Scores are of a major importance in compositional processes and are critical components in computer-aided composition frameworks. Several types of objects allow to represent musical data under the form of score in OpenMusic, using different temporal or notational conventions. These objects, used or generated in visual programs, can be manipulated using graphical editors where more standard musical software features are available (mouse selection and editing, playback, export, etc.—see Fig.2)

III. SHEET PROGRAMMING

The sheet is a special polyphonic score editor developed recently in OpenMusic, which introduces two important concepts [3]. First, it is meant to gather heterogeneous types of musical objects in its different voices, which involves specific and non-trivial handling of mixed time representations such as rhythmic notation (traditional scores), linear time notation (e.g. MIDI-like or “piano-roll” notation), or continuous representations (e.g. sound files or control curves). Fig.3 shows a sheet object created in an OpenMusic patch starting from a set of other musical objects. The sheet editor is visible at the bottom of the figure.

In the editor each object can be sized or positioned in time. Objects can also be added or moved between the different voices of the score. The score display algorithm ensures strict alignment of simultaneous events or objects depending on score spacing and other graphical constraints (see [3]).
The second important feature in the sheet editor deals with programming musical objects: A sheet includes a set of internal programs (or patches) which allow generating the objects in the score algorithmically or to relate them to one another. The sheet patches can be shown or hidden from the editor. When visible (see Fig.4), they can embed some special boxes called “sheet-access” representing their container sheet object. A sheet-access box is initialized with a number corresponding to unique IDs assigned to the musical objects in the sheet (the selected object then appears highlighted on the sheet-access box display). The other inlets/outlets of the sheet-access allow to connect and read or write the contents or temporal attributes of this object in the patch.

An arbitrary number of patches can be attached to a sheet object, and evaluated on request in order to compute or update the score contents.

IV. Conclusion

OpenMusic can be a powerful programming framework for various types of musical (or extra-musical) purposes [2]. It has been used for the creation of numerous contemporary music pieces by composers from varied aesthetic and geographical origins (user experiences in real-size projects and works have been reported in [4]).

The extensions presented here with the sheet and the general idea of score programming are examples of how computation and programming can tightly integrate compositional processes. The functional relations set between the heterogeneous components of such extended scores will hopefully enhance compositional processes carried out with computer systems and open new ways of representing and analyzing music.

REFERENCES