A Compositional Environment based on PrefORM II, PatchWork and Esquisse

Jacques Duthen
IRCAM, Department of Musical Research,
31 rue Saint-Merri, F-75004 Paris, France.
e-mail: duthen@ircam.fr, jacques@ircam.duthen.

Mika Laurson
Prof. at Sibelius Akademia
Hollantievanpee 1A2
00330 Helsinki 53 Finland

ABSTRACT: We present an integrated compositional environment written in Le_Lisp on the Macintosh II and based on some of our previous research done in IRCAM and related to computer-aided composition. PrefORM is an object-oriented system which includes a MIDI driver, a scheduler, a graphical toolbox, a library of graphical controls and a multi-application environment, containing various editors, a graphical functional programming language named PatchWork, and a library of compositional functions named Esquisse.

Introduction

The aim of this project is to build a unique environment and open computer-aided composition system that offers a number of basic functions and editors used by the composer to design structures, manipulate and edit them under various representations, as well as adding new functions and new editors according to the compositional needs. This system can be used for creating a whole piece from sketch to realization, as well as controlling synthesizers (hardware like DX7, software like Clavis, or any synthesizer or processing program). The basic approach of this compositional environment is to sketch compositional structures by generating, filtering, and editing data.

PrefORM is used as a platform on which various event-driven applications can run simultaneously. The Le_Lisp package already contains the toplevel and a text editor for Lisp source files. The currently available applications, written in PrefORM, are:
- a music notation editor
- a break-point function editor (fig. 1)
- a continuous function editor
- a series of musical tools related to Alan Fote's Sent Theory
- PatchWork, a graphical programming language, using functional modules connected in patches
- Esquisse, a library of compositional functions implemented as PatchWork modules.

Each of the editors can be miniaturized in a (non-functional) PatchWork module. Each application has its own menu bar, its own on-line documentation and its own list of windows. The user can switch from one application to another by selecting it in the "Apps" menu, or by clicking on one of its windows, or from PatchWork, by opening one of its miniaturized images (see fig. 2).

This compositional environment is based on the combining of those different editors inside PatchWork and the addition of a series of new functions oriented towards editing musical structures for producing a traditional instrumental score or controlling sound synthesis and/or processing.


We strongly believe that a compositional environment must be designed as an integrated but open system that allows for taking into account all aspects of composition from the mere conceptual structures to the details of the realization of sound synthesis, when this happens to be considered compositionally relevant. In terms of programming, this means to allow applications (i.e. complex functions) to communicate by sharing and exchanging data through messages passing.

In order to build such a compositional environment, we decided to take advantage of the experiences accumulated with PrefORM, PatchWork and Esquisse.

After several years of experience, the original design of PrefORM has been almost completely transformed and rewritten by J. Duthen. This new PrefORM called PrefORM II, is the foundation on which this new environment, provisionally called Sketch-Pad, has been designed, from the merging of PatchWork and Esquisse.

Esquisse was originally by itself a compositional environment consisting of various libraries of compositional rules for generating and manipulating material. The connection with PatchWork allows the composer to patch together in a very flexible way generating and filtering functions, as well as to connect editors at various levels of the patch.

ICMC GLASGOW 1990 PROCEEDINGS

228
PatchWork is an integrator of graphical applications written in PreFORM II. It is designed to connect into a patch some generator and receivers through transformers. Every patch can be "abstracted" into a new object available in the system and duplicable. The objects can be simple generator objects like a variable, or random generator, or numeric input controller (like a scroll-bar). They can be "abstracted" patches designed by the user, or any custom-made object (whose functional properties can be easily designed by any programmer). They can even be whole applications which appear as "minimized" objects in the patch. Every object receives its data from its inputs in a more formal way than the simple "cut/copy/paste" paradigm (which can also be used). Each functional transformer treats the data algorithmically.

The data structures can be shared by different applications. Modules can talk to applications, and applications can talk to each other.

In the case of editors, the user can act directly and graphically on data as well as select it and create a minimized version of it to PatchWork modules to process it with Express-like functions.

II - Creating a notation editor which includes control over synthesis

The way the user interacts with data is defined in an object "editor", which contains a series of controllers (cf. the "Model-View-Controller" paradigm of Smalltalk). These controllers are event-driven; they use the standard Macintosh events (i.e. key, mouse) as well as the X-Window-like events (move-event, drag-event, enter-window, leave-window) implemented in a new event-manager. The use of these events, the model-editor splitting, the structuring of the editor into several small elementary controllers each one with its specific cursor, and a new mode of expressivity (selection by proximity instead of clicking) allow quicker interaction than usual Macintosh applications and a great flexibility in customizing the interaction part of the editors.

Using this technique, a traditional notation editor has been designed and realized, no aimed at publishing quality but aimed rather at allowing for compositional manipulations on musical structures, including not only traditional parameters (pitches, durations, and dynamics) but also the control of timbral characteristics for synthesis.

Any pitch structure (even quarter tones or non-tempered scales) or irrational rhythms can be represented and edited.

Break-point functions can be drawn and manipulated with a specialized editor and then associated with a specific parameter of an event (or note) to any group of events. These continuous controls can also be specified in the form of rules described by the composer at any level of structural granularity (i.e. the level of the note or the different arbitrary levels of groupings).

The composer can also define any arbitrary mapping between parameters internal to the editor and synthesis parameters of a specific synthesizer.

Conclusion

This compositional environment is now being used by composers M.A. Dalbavie and M. Lindberg for the realization of two new pieces at IRCAM.

We are aiming at connecting this environment to the new IRCAM workstation, by transporting it to the NeXT workstation.

We are also working on connecting this compositional environment to a DSP card like the Sound Accelerator by Digidesign.

Finally we are planning on transferring the whole environment to Common Lisp so as to make it more widely available.
Figure 1: The break-point function editor

Figure 2: a simple patch for the interpolation of two chords and its result in the music notation editor