A Signal Editor for the Ircam Musical Workstation

Gerhard Eckel

A Signal Editor for the IRCAM Musical Workstation

Gerhard Eckel

ICMC 90, Glasgow (Ecosse) 1990

Abstract

The SignalEditor is an extensible set of general purpose tools for viewing, analyzing, editing, and processing sampled signals on the IRCAM Musical Workstation (IMW). Its main intention is to provide the composer and the scientist with a sophisticated interface to handle digital audio signals naturally. This involves a high quality graphics interface, a flexible signal analysis system, an efficient sound file system, and a powerful signal processing environment. In this paper we discuss a prototype implementation of the spectrogram editing facilities, some aspects of the editor's extension language and the lowest layer of the sound file system.

Introduction

The SignalEditor is an application running on the NeXT computer system under NextStep and MACH. A subset of the SignalEditor, without graphics, runs on several UNIX platforms. The SignalEditor is designed with Interface Builder using a set of Objective-C classes built for signal management, processing, viewing and editing. These classes are supplied in the form of the SignalEditor Kit, which is well integrated into the NeXT Application Kit. Other applications of the IMW (like ANIMAL [1] or the Universal Recorder [2]) will use the SignalEditor Kit directly or use the SignalEditor application via MACH messages. The SignalEditor comprises an extension language based on Scheme [3] which provides a generic interface to the facilities of the IMW's i860 coprocessors via CPOS and FTS [4].

This paper describes the ideas which led to the design of the SignalEditor and it discusses some features which have already been realized. A general survey of the SignalEditor's features will be given followed by the description of its extension language. In particular we will discuss a prototype implementation of the spectrogram editing feature that has been realized in the form of the program SpecDraw, which will be presented in a demonstration session of this conference.

The SignalEditor

The SignalEditor is a tool for viewing and editing signals in various graphical representations. These are time domain (e.g. waveform) and frequency domain (e.g. amplitude spectrum) representations as well as combinations of the two (e.g. spectrogram). The SignalEditor eases the auditive and analytical evaluation and comparison of signal segments. This is accomplished by efficient and flexible signal access both on the level of graphical representation and sound playback. Therefore it allows selection and management of signal segments and related information. Signal segments are defined in the time domain, the frequency domain, or as a region of arbitrary shape in a time-frequency representation (e.g. a formant region). Segments can be named and referred to by name in subsequent operations. A set of segment management operations allows to deal efficiently with the segmented sound material.

In all signal representations one can select, copy, cut, and paste signal segments in order to change the

sequence of samples in the signal. Editing of sample magnitudes in the time domain (e.g. envelope editing) as well as filtering operations based on frequency domain representation (e.g. spectral envelope editing) are supported. Another aspect of the SignalEditor is its ability to extract features from signals using advanced signal analysis methods. The extracted features might guide segmentation, editing, or any other operation. Furthermore the SignalEditor supports a generic interface to signal processing facilities of the IMW. This allows different production styles to be realized in the same environment. The SignalEditor can be seen as the uniform interface for any signal used within the context of the IRCAM Musical Workstation.

The main intention of the design of the SignalEditor was to integrate known techniques of signal representation and editing into a tool which is flexible enough to allow new methods to be integrated easily. Several signal representations showing different aspects of a signal are visible simultaneously, allowing the observation of the consequences of a change made in one representation.

The Extension Language

In order to make the SignalEditor an extensible and flexible tool it needs to be programmable. This allows the user to program when he needs and wants to but does not oblige the use of a programming language to run the SignalEditor. That feature is well known from the text editor Emacs [5] which is one of the best examples of an apphcation that can be used as easily by a novice user as by an expert user. In addition, such kinds of applications allow smooth transition from the novice to the experienced user. We believe that this strategy can also be used with success for applications based on a graphics user interface like the SignalEditor.

Since the use of extension languages for non-trivial applications becomes more and morc common in software development the number of language implementations suitable for that purpose is increasing. The choice of the extension language for the SignalEditor was guided by three major constraints: the language used should be a standardized, interpreted, high level language; furthermore it should comprise a small but powerful set of concepts that allow different programming styles and it should be easily available in form of C implementations. The language Scheme fits the first two constraints ideally. After having compared different Scheme interpreters we found the very careful C implementation in the form of ELK 61 to be a good basis for the extensions language of the SignalEdilor. ELK allows the easy inclusion of Scheme primitives written in C via its dynamic loading mechanism This feature is used to add the functionalities needed by the SignalEditor to ELK. We will now briefly discuss three sets of primitives added so far: the Objective-C interface, the signal processing interface and the Yirtual file interface.

The Objective-C interface TELL allows access to all classes known by the SignalEditor. These are the classes defined in the various class libraries coming with the NeXT system and the classes which form the SignalEditor Kit. TELL defines a data type to handle class and object pointers and it supplies a function to send messages. This enables the extension language to intervene on all levels of the application. Thus TELL guarantees the desired flexibility of the extension language which has already been proven to be very useful for debugging.

The signal processing interface SPEX [7] is based on the UDI library [8] developed a IRCAM. UDI comprises a large number of signal processing routines and it runs on many different platforms. SPEX introduces data types to ELK which allow storage, I/O and processing of signals using UDI routines, which are made available in the form of Scheme primitives. SPEX currently runs on the NeXT, Sun3, and DEC3100 workstations. When used by the SignalEditor on the NeXT the SPEX functions will run on the i860 of the IMW via CPOS and FTS.

The virtual file interface VFILE is based on a library of C functions which constitute a virtual file scheme. These functions duplicate the functionality of the UNIX standard buffered input/output package (stdio). Additionally they allow one to insert, move, and delete data at arbitrary positions in the file. This feature, which has been implemented very efficiently, was desired for editing large sound files. A virtual file is

represented on disk as a directory containing a chain of fragment files which make up the virtual file. The order of the fragment files is established by the lexicographical sequence of their names. The file names also encode other information like the byte offset and size of the fragment. This scheme appears to be the most robust in keeping virtual files consistent because standard UNIX file system features such as directories and hard links are used. Since real time access of virtual files can be guaranteed the design of the new sound file system will be based on this layer.

SpecDraw

SpecDraw is a prototype implementation of the spectrogram editing feature of the SignalEditor. It represents the signal in the form of a spectrogram in which arbitrary regions may be selected and used in further processing. Two modes of processing are currently available. Either the selected regions are erased from the signal or all other parts but the selected regions are erased. The final implementation will generalize this approach and will allow any desired operation on the selected regions. This kind of signal interface allows relating characteristics readable in the signal representation to their auditive appearance, which is one of the central tasks we are confronted with when editing and processing sound signals. It is essential that the signal representation is adapted to the charactenstics of auditive perception. Therefore the SignalEditor will support perception-relevant scales like the Bark scale and it will take into account certain phenomena of auditive perception such as the masking effect [9].

The editing of time-frequency distribution based signal representations is the main novelty of the SignalEditor. SpecDraw serves mainly for validation of the ideas concerning this new type of interface. The program has been presented to a number of composers and scientists and the positive feedback it provoked brought many new ideas which will be taken into account in the final implementation of the SignalEditor.

Reference

[l] E. Lindeman, "ANIMAL: A Rapid Prototyping Environment for Computer Music Systems ", Proceedings of the ICMC, September 1990.

[2] B. Smith, "A Universal Recorder for the IRCAM Musical Workstation ", Proceedings of the ICMC, September 1990.

[3] J. Rees & W. Clinger (ed.), "Revised3 Report on the Algorithmic Language Scheme ", AI Memo 848a, MIT, September 1986.

[4] E. Viara & M. Puckette, "A real-time operating system for computer music", Proceedings of the ICMC, September 1990.

[5] R. Stallman, "GNU Emacs Manual", Sixth Edition, Emacs Version 18, March 1987.

[6] O. Laumann, "Reference Manual for the Elk Extension Language Interpreter", Technical University Berlin, Communications and Operating Systems Research Group, May 1990.

[7] G. Eckel, "SPEX 0.9, Scheme Signal Processing Extensions, Reference Manual", internal document, IRCAM, June 1990.

[8] P. Depalle & X. Rodet, "U.D.I.: A Unified D.S.P. Interface for sound signal analysis and synthesis ", Proceedings of the ICMC, September 1990.

[9] G. Eckel, "Ein Modell der Mehrfachverdeckung für die Analyse musikalischer Schallsignale", Ph.D. thesis, University of Vienna, June 1989.