A Modular Construction Set for Time-Domain Editors

Bernhard Feiten
Institut für Kommunikationswissenschaften
Technische Universität Berlin
feiten@e-lgv.TU-Berlin.DE

Markus Spitzer
Institut für Kommunikationswissenschaften
Technische Universität Berlin
msp@e-lgv.TU-Berlin.DE

Abstract

The paper describes a widget set, which is designed to facilitate the development of graphical editors for data based upon time. The groundwork is a basic editor, which provides basic functions like zooming and scrolling. Specific editors are derived from the basic editor in an object-oriented manner and allow manipulating varied data-structures like samples and sonograms. Several widgets for the vertical positioning of editors in tracks and a few auxiliary widgets like axes are available. A front-end for sound synthesis processes is presented, which uses the described widget set.

1. Introduction

The graphical user-interfaces for complex sound synthesis and sound manipulation processes are not yet sufficient realised. One reason is the lack of powerful development tools.

At the Institut für Kommunikationswissenschaft at the TU Berlin we are working on a project with the purpose of creating a set of tools that relieves rapid and easy developing of applications for editing time-based data. During the conception we were guided especially by the demand for portability and re-usability.

As a platform we chose Unix systems with the X Window System (X11 Release 5) and rely on the object-oriented widget concept of the X Toolkit (Xt and Xaw). In simple terms, a widget is an object with a visual representation on the display and a defined interaction behavior. The widget concept supports the re-usability of partial components by encapsulation of functions. X11 and Xt are widespread standards and therefore fulfill the demand on portability.

![Diagram of widget hierarchy](image)

**Fig. 1: Hierarchy of editor widgets. Arrow stands for the predicate "a derived from".**

2. Editor Hierarchy

We constructed a widget set, which essentially is composed of two parts. On the one hand we built up a multi-stage hierarchy of editor widgets, which roots in a basic editor widget. The basic editor provides all functions, which are recurrent in different graphical editors. Specific editors were derived from the basic editor. The hierarchy in figure 1 shows the actual state, but extensions are possible and desirable.

2.1 Basic Editor

The basic editor offers a uniform working surface to each derived editor. It provides a definable system of co-ordinates and a set of graphical basic functions working in that system. It offers the facilities to zoom and to scroll independently of the following levels of hierarchy. It provides an extensible popup menu, which allows to call fundamental methods such as load & save, print, select, cut & paste. As a rule, these methods are implemented by all derived editors.

![Sample editor image](image)

**Fig. 2: Sample editor**

2.2 Derived Editors

A new editor inherits all functions from the basic editor and adds specific functions like its own data structures, their input, output, print and redraw algorithms. Up to now we have derived a sample editor, an editor for additive synthesis, an envelope editor and a sonogram editor. Figure 2 shows the sample editor, figure
3. Positioning of Editors

The widget sets second part is a group of widgets, that enables an application to position arbitrary widgets in vertical direction in area, similar to channels or tracks. The track widgets are provided with an arbitrary definable abscissa and allows zooming and scrolling.

4. Score-Editor

As a second step of our project we develop a score editor based upon the described widget set, that makes it possible to arrange musical phrases in an arbitrary number of tracks, to modify them, to calculate and to play them. In this connection a phrase is an instance of an abstract instrument definition whose parameters are controlled by an arbitrary number of changeable envelope instruments. The instrument is a sample instrument, which is controlled by amplitude and frequency envelope and a FM- or physical-model instrument whose parameters are controlled by envelope. Finally the instrument definition by a Csound-like programming language with graphical front-end should be possible. A score editor can also be used as front-end for different synthesis processors, e.g., controlling MIDI-devices.

5. Summary

The two parts of the widget library, the editor hierarchy and the widget set editor arrangement, together result in a modular construction set for implementing complex graphical editors in short time. Various applications are imaginable, which use different levels of the library. An application for example could use only the zoom, scroll and drawing features of the basic editor but manage its own data structures independently. As the score editor, another application could use the leads of the editor hierarchy. These are almost autonomous editors. Moreover, widget programmers can add new editors to the hierarchy by deriving them from existing ones.

Using X11 and Xfree86 leads to the desired portability. Hibbert the widget set is running under the operating systems Irix, Ultrix, SunOS and Linx. 