HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering

Martti Rahkila

A COMPUTER BASED EDUCATION SYSTEM FOR SIGNAL PROCESSING

This Master’s Thesis has been submitted for official examination for the degree of Master of Science in Espoo on April 19, 1996.

Supervisor and instructor of the Thesis

Professor Matti Karjalainen

HELSINKI UNIVERSITY OF TECHNOLOGY — ABSTRACT OF THE MASTER'S THESIS

Author:	Martti Rahkila	
Name of the Thesis:	A Computer based education system for signal processing	
Date:	April 19, 1996	Number of pages: 47
Faculty:	Electrical Engineering	
Professorship:	S-89 Acoustics and Audio Signal Processing	
Supervisor:	Professor Matti Karjalainen	

In this work, computer based education (CBE) methods are applied to signal processing. The work consists of developing a CBE environment for signal processing education and as a case study, building a CBE application "Introduction to Signal Processing" with that environment. The application was used as training material within the course "Fundamentals of Acoustics I" and based on that, a user feedback analysis was performed.

An overview of computer based education theory and principles is presented. Also, two CBE design methods, the Lifländer method and the Lessonware method along with several authoring tools are discussed.

Signal processing education has been evaluated in this thesis. The suitability of CBE methods in signal processing has been proven. A number of signal processing software tools and education related examples are presented.

The signal processing software environment QuickSig based on Common Lisp language is introduced and new hypermedia features including "HotAreas", graphical user interface (GUI), and navigation tool needed for CBE developing are discussed. A log-system with appropriate login and logout dialogs for later usage analysis is also presented.

A CBE application "Introduction to Signal Processing" is examined. The educational contents of the application is based on a query performed with the Helsinki University of Technology signal processing education staff and on literature of the field. Interactive excercises are presented with examples. The target group and goals for the application are explained. The user feedback analysis results are discussed.

Finally, the work is summarized and future prospects are examined.

Keywords: computer based education (CBE), signal processing, digital signal processing (DSP), hypermedia

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Martti Rahkila	
Työn nimi:	Tietokoneavusteinen opetusympäristö signaalinkäsittelyyn	
Päivämäärä:	19.4.1996	Sivumäärä: 47
Osasto:	Sähkötekniikka	
Professuuri:	S-89 Akustiikka ja äänenkäsittelytekniikka	
Työn valvoja:	Professori Matti Karjalainen	

Tässä työssä tutkitaan tietokoneavusteista opetusta (TAO) signaalinkäsittelyssä. Työ koostuu signaalinkäsittelyyn soveltuvan TAO-ympäristön kehittämisestä ja esimerkkisovelluksen “Introduction to Signal Processing” rakentamisesta. TAO-sovellus oli käytössä “Akustiikan perusteet I”-opintojaksolla lisämateriaalina ja tämän perusteella suoritettiin käyttöanalyysi.

Aluksi tehdään katsaus tietokoneavusteisen opetuksen teoriaan ja periaatteisiin. Kaksi TAO-suunnitelumenetelmää: Lifländerin menetelmä ja Lessonware malli, sekä useita TAO-kehitystyökaluja käydään erityisesti läpi.

Signaalinkäsittelyn opetusta arvioidaan laajalti. TAO-menetelmien soveltuvuus signaalinkäsittelyn opetukseen osoitetaan. Joukko signaalinkäsittelyn työkaluja ja esimerkkejä opetuskäytöstä esitellään.

Common Lisp -kieleen pohjautuva signaalinkäsittely-ympäristö QuickSig esitellään ja uudet hypermediapiirteet kuten kuumat alueet, graafinen käyttöliitymä sekä navigointityökalu käydään yksityiskohtaisesti läpi. Järjestelmään rakennettu lokisysteemi login- ja logout-dialogeineen on esitelty.

TAO-sovellusta “Introduction to Signal Processing” tarkastellaan yksityiskohtaisesti. Sovelluksen opetussisältö perustuu Teknillisen Korkeakoulun signaalinkäsittelyn opetushenkilökunnan keskuudessa suoritettuun kyselyyn sekä alan kirjallisuuteen. Sovelluksen kohderyhmä sekä tavoitteet ovat niinikään selvitetty. Käyttöanalyysin tulokset esitellään.

Lopuksi esitetään yhteenveto ja arvioidaan tulevaisudennäkymiä.

Avainsanat: tietokoneavusteinen opetus (TAO), signaalinkäsittely, digitaalinen signaalinkäsittely, hypermedia

PREFACE

This work has been carried out at the Laboratory of Acoustics and Audio Signal Processing, the Helsinki University of Technology, during the years 1994–1995. The work has been part of a Computer Based Education development project financed by the Helsinki University of Technology. I am very grateful to the supervisor and instructor of my work, Professor Matti Karjalainen, for inspiring, guiding, assisting and supporting me throughout the work, even while being at CCRMA, Stanford University, USA, most of the time. I would also like to thank professor Karjalainen and Lic. Tech. Toomas Altosaar for their expertise and help in LISP programming. Dr Unto Laine is acknowledged for very interesting discussions and guidance on DSP issues. My very special thanks I would like to express to my friends Dr Vesa Välimäki and M.Sc. Jyri Huopaniemi for being enourmously encouraging and helping me with numerous concerns. Also the rest of the Acoustics lab team deserves my sincere gratitude for inspiration and support.

Furthermore, I would like to thank professors Iiro Hartimo and Olli Simula for their interest in my work, all the other people that gave me their opinions on the basic concepts of signal processing, Mr Veli-Pekka Lifländer for his excellent advice concerning CBE and my beta test crew Dr Vesa Välimäki, M.Sc. Jyri Huopaniemi, and Mr Aki Härmä. Special thanks are due to all the students that used the program in autumn 1995; you made my work worth while. I also thank CARTES (Computer Arts Centre of Espoo) for kindly providing me working facilities during summer 1994.

This work would not have been possible without the support from my friends and relatives. Therefore I thank my parents, Tuula and Matti Rahkila and my sister Aija Rahkila. Very special thanks I give to my fiancée Outi Kosonen for her love and support throughout my studies.

Munkkiniemi, Helsinki, April 19, 1996

Martti Rahkila

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CAL	computer assisted learning
CBE	computer based education
CLOS	common lisp object system
DSP	digital signal processing
FFT	fast Fourier transform
GUI	graphical user interface
ICASSP	international conference on acoustics, speech and signal processing
MCL	Macintosh common lisp
TMS320C30	a signal processor from Texas Instruments

INTRODUCTION

"The human mind... operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain... the speed of action, the intricacy of trails, the detail of mental pictures, is awe-inspiring beyond all else in nature. Man can not hope fully duplicate this mental process artificially, but he certainly ought to be able to learn from it." (Bush, 1945)

Computer Based Education (CBE) in signal processing is the topic of this work. The aim has been to create a system for developing signal processing CBE and to produce a tutorial CBE application presenting the basic concepts of signal processing.

Modern CBE typically applies hypermedia to achieve it's goals. The reason for that is summarized nicely in the citate above by Vannevar Bush, the pioneer of hypermedia. This way, a completely new level of interacitivity can be accomplished in teaching. But it isn't free: flexible, easy to use systems and environments are needed.

Signal processing is a field of special interest in CBE. In Digital Signal Processing (DSP) computers are the tools, which makes them an ideal topic for CBE. However, a lot of computational power is needed for signal processing. Luckyly, with new multimedia computers, it is possible to compromise the requirements of both signal processing and CBE in a satisfactory manner.

QuickSig is an object-oriented signal processing environment. It provides both flexibility and power, which makes it a good base for signal processing CBE. Within this work, it has been extended to support CBE purposes. A variety of hypermedia functionality has been added. Particular interest in CBE development and maintenance has been paid.

A CBE-application, “Introduction to Signal Processing” has been created. It is a tutorial of the fundamental concepts of signal processing especially from the viewpoint of audio and sound. The application was immatediately taken into use within the Acoustics laboratory’s teaching. The usage was also analyzed.

This thesis is divided into six major sections. CBE principles are discussed in chapter 1 and signal processing specific questions in chapter 2. The QuickSig environment and the new hypermedia and CBE features are presented in chapter 3. Chapter 4 overviews the CBE-application “Introduction to Signal Processing” along with it’s contents and design issues. The analysis of using the application in autumn 1995 is presented in chapter 5. Chapter 6 draws the conclusions and discusses future development.

1 COMPUTER BASED EDUCATION PRINCIPLES

1.1 Introduction

In a broad sense Computer Based Education (CBE) means using computers in education for all kinds of purposes. This includes both the learning process itself and the educational administration. Typically, CBE applies hypermedia tools to accomplish these tasks. The interest in CBE has grown in the past few years all over the world.

The terminology of Computer Based Education is quite colourful depending on historical, geographical and other differences in development history and how generally the term should describe the matter (Laitinen, 1988). Beside CBE, a set of other terms meaning the same, are often used. The following table presents the most common terms and their interpretations (table 1):

CBE terminology	
Computer Based Education (CBE)	General term
Computer Assisted Instructions (CAI)	General term
Computer Aided Learning (CAL)	Emphasis on the student
Computer Based Instructions (CBI)	
Computer Based Training (CBT)	
Computer Managed Instructions (CMI)	Both teaching and student guiding

Table 1: CBE terminology

In the author’s opinion, the general nature of the term CBE best describes the matter in this case.

1.2 Design principles

Even though CBE can be applied in almost any education, it is not always meaningful. Using CBE must be in some way more useful than ordinary teaching methods. That is why the starting point for every CBE-project is to think if CBE can give something "extra" or perhaps a "better way" in the means of education compared to ordinary methods. However, CBE can easily provide some elements in education, for example interactivity, that are difficult to achieve using standard methods (Laitinen, 1988). With signal processing, and especially with Digital Signal Processing (DSP), CBE can be very fruitful.

Another important aspect is the quality of CBE, which includes both the quality of the contents and the quality of the application, namely its user interface. Getting high quality requires careful planning. Luckyly, a number of design methods for CBE exist (see below). These methods help defining the contents and the logical structure as well as the interface itself. The key idea is that the emphasis of a CBE-application should always be on the subject, not on the program itself! Both the structure and the interface should support that principle.

1.3 Design methods

There are several design methods for CBE. The methods emphasize different aspects and one should choose whatever fits one’s purposes best. Studying and using these methods will certainly pay back, because the time spent on planning can be reduced and the quality of the final result, the CBE application, will be improved.

One division of the methods is (Likländer, 1988):

- behaviouristic methods
- cognitive methods
- methods for learning environments
- designing intelligent applications

In our project we have used the Lifländer design method (Lifländer, 1988, 1989) that is based on the studies of learning by Engeström (1987) and also some ideas of the Lessonware (Crossley & Green, 1985). They both lie in the category of cognitive methods, but also represent ideas of learning environments. An overview of existing design methods can be found in (Laitinen, 1988) and (Lifländer, 1989).

1.3.1 Lifländer design method

The pedagogic background of the Lifländer design method are the studies of learning by Yrjö Engeström (1987). The basis is Engeström's 6-phase model of a learning process:

- motivation
- orientation
- comprehending
- applying
- evaluating concepts
- controlling learning results

Another key concept is an "open, controlled learning environment" also based on Engeström's studies. The term stands for a system that contains a controlled learning process, but it can also be easily expanded by both the student and the teacher. The emphasis of the Lifländer method is in the teaching tasks: small tasks that activate student and thus power up learning.

A summary of the Lifländer design method (Lifländer, 1994) is presented in figure 1.

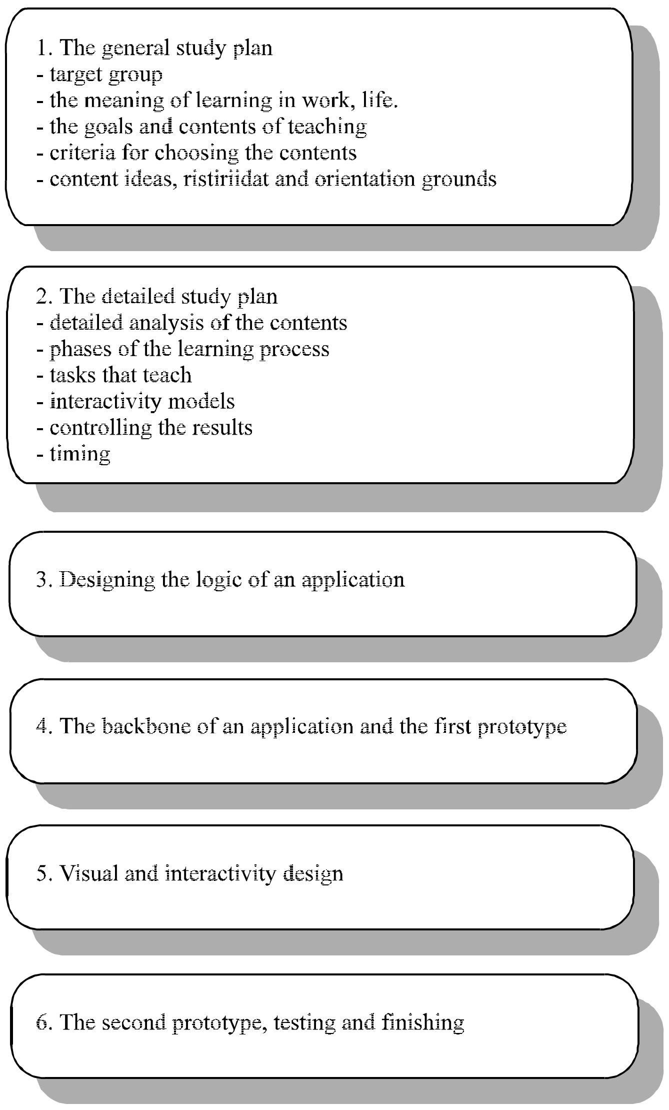

1. The general study plan
- target group
- the meaning of learning in work, life.
- the goals and contents of teaching
- criteria for choosing the contents
- content ideas, ristiriidat and orientation grounds
2. The detailed study plan
- detailed analysis of the contents
- phases of the learning process
- tasks that teach
- interactivity models
- controlling the results
- timing
3. Designing the logic of an application
4. The backbone of an application and the first prototype
5. Visual and interactivity design
6. The second prototype, testing and finishing

Figure 1: The Lifländer design method

Graphical presentations of the general and the detailed part of a teaching plan for the application “Introduction to Signal Processing” are presented as Appendix A and B.

1.3.2 Lessonware

Crossley’s and Green’s Lessonware method also provides an open learning environment. The key idea is not to plan student’s learning “route”, instead a model of a market place, where users can freely move and experiment, is presented. The Lessonware method also emphasize the didactic and pedagocic aspects, technical issues are discussed only at the end of the design phase. Figure 2 presents the Lessonware method in a nut shell (Laitinen, 1988).

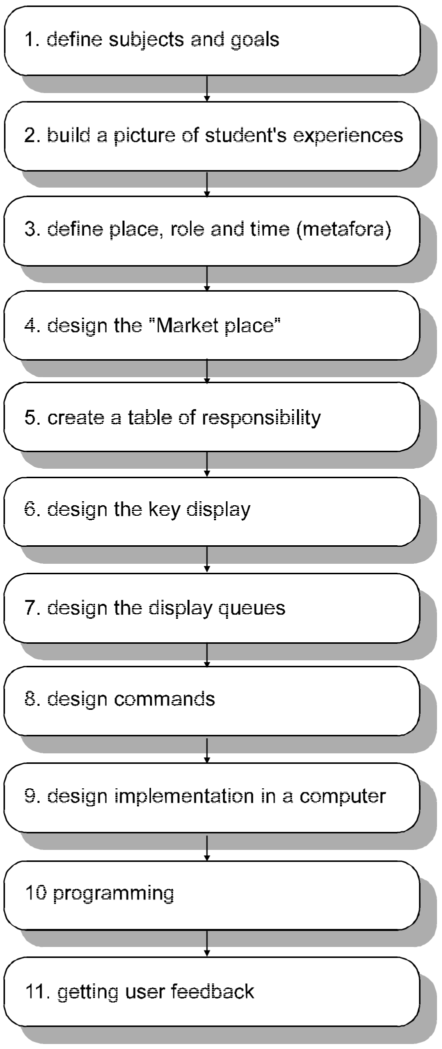

1. define subjects and goals
2. build a picture of student's experiences
3. define place, role and time (metafora)
4. design the "Market place"
5. create a table of responsibility
6. design the key display
7. design the display queues
8. design commands
9. design implementation in a computer
10 programming
11. getting user feedback

Figure 2: The Lessonware method

1.4 Costs of CBE

The costs of CBE have become lower and lower due to the heavy competition and technological progress in the computer hardware and software markets. The most expensive part of a CBE-project is the design phase, but also the desired equipment, especially in DSP, can be quite expensive. Applying design methods can dramatically lower the costs and improve the results and is, therefore, highly recommended. The costs can also be lowered using "general-purpose" authoring and hypermedia tools, but these are not necessarily suitable for all education. However, the total price for CBE is in many cases competitive with ordinary education costs, especially when the number of students is high. A more thorough discussion of the costs of CBE is presented by Lifländer (1988).

1.5 Choosing the system

Naturally choosing the system for implementation is one of the key questions in CBE. At least the following things must be taken into account:

- the suitability of a system for the area of teaching
- the desired features in terms of using and distributing the application
- special demands of environment
- further development

Several general-purpose authoring tools exist, but they are not necessarily suitable for all education. For example, in DSP, we often need as much computational power as we can get. Therefore easy to use, but heavy to run authoring tools are out of the question. Authoring tools also lack many desired features in developing DSP algorithms. This was the main reason and innovation for creating a new CBE system especially for DSP education.

1.5.1 Authoring tools

HyperCard is Apple’s own hypermedia tool for Macintosh. Over the years, it has proved to be a flexible and good environment not only for CBE but also for other applications. Earlier it was bundled in the Macintosh System, but the latest versions have been sold separately. Nevertheless, HyperCard is a widespread tool and at least thousands of applications for it exist. Nowadays the program itself is not necessary for running HyperCard stacks, instead a HyperPlayer can be used.

Toolbook is a Windows-based hypermedia tool from Asymetrix. Like Hyper-Card, it has been used also for other kind of applications, for instance multimedia and Internet, as well. Toolbook’s main distribution philosophy has always been that a free runtime-version of the program kernel can be distibuted with the applications thus making them stand-alone. The latest version of Toolbook presents many advanced multimedia features. There are also special university and educational licences.

Authorware Professional is a specialized tool for CBE development. From educa-tional point of view it has several advantages over the previously mentioned ones. It is available at least for Macintosh and Windows-environments. However, it is not so widespread as HyperCard or Toolbook, possibly due to being more expensive.

SuperCard and MetaCard are HyperCard-like programs with some enhancements. SuperCard is available for Macintosh and MetaCard for Unix X-windows environ-ment. Because Unix systems offer several advantages over microcomputer environ-ments, MetaCard is a good alternative.

MacroMedia Director is a professional multimedia developing tool that has been used more for other kind of multimedia products than CBE. It has very advanced features, thus being also quite expensive. Recently (autumn 1995) MacroMedia made a contract with World Wide Web-software developer, Netscape, which could make this product the most attractive alternative for CBE developers as well.

2 CBE AND SIGNAL PROCESSING

2.1 Signal processing

Signal processing means that we use systems to operate on signals. In practice, all measurable quantities can be understood as signals and everything that we do with them is signal processing (Proakis et al., 1992). Since this is a very convenient way to understand many kinds of problems, signal processing is applied in a number of fields of interest: communications, audio and image technology, measurement technology, economics etc. Such common devices as TV, radio and telephone are based on signal processing. Therefore it is necessary to teach signal processing at least in technical universities.

An important division of signal processing is between analog and digital signal processing (DSP). Making it short, the difference is what kind of technology is used: analog or digital. They both have their advantages and drawbacks, but in the last 10-15 years digital signal processing has become more and more important. The following analysis of signal processing education is thus somewhat DSP oriented.

2.2 Suitability for CBE

Signal processing is a special area of interest in CBE. In Digital Signal Processing (DSP) all practical work is done with computers which makes it a very natural topic for CBE as well. As a matter of fact, computer experiments are considered necessary for DSP education throughout the world. This can be seen for example from the ICASSP proceedings signal processing educations sessions that first appeared 1992. Thus, it fulfills the CBE principle number one: CBE methods definitely are suitable for signal processing education (Rahkila & Karjalainen, 1995).

Also, the fast progress in computer technology in the last few years supports the concept: advanced multi and hypermedia features of new computers are particularly well suited to support the education of signal processing concepts. Non-realtime and even realtime processing of audio and speech signals and images is already made possible without any or with minimal extra hardware. CD-quality (16-bit) sound input and output are standard features in many multimedia computers. High-quality graphical interface, including video and animation, adds to the means of user interfacing.

2.3 Special demands

Signal processing applications demand a lot of computer capacity (and often some special hardware). Many traditional DSP tools have therefore been designed to be efficient but they are not necessarily very flexible. On the other hand, CBE-applications demand great flexibility of the environment and good tools for interface design. The authoring and hypermedia tools, such as HyperCard and ToolBook, contain that but they usually do not support special requirements of individual teaching areas like DSP. Therefore they are not necessarily suitable for DSP purposes.

In CBE, also real-time processing or at least reasonably short response times are needed in order not to frustrate the user and make the application unusable. This states extra demands on the software: It should provide maximum support for signal processing tasks and maximum flexibility and efficiency on the user interface. However, in DSP education it is also very important to understand the limitations of the system used and therefore waiting for a DSP task to finish can be tutorial as well. The processing must be done in a reasonably short waiting time though.

What is needed is a flexible yet efficient environment for both signal processing and CBE. For example QuickSig (Karjalainen, 1990) fulfills these requirements.

2.4 DSP tools

There are dozens of DSP software available in the world. A couple of them that have been used in signal processing education are introduced here.

2.4.1 Matlab

Matlab (from Mathworks Inc.) was at first created for numerical matrix calculations (Matlab = Matrix Laboratory), but it soon became the leading numerical mathematics software. Matlab is available al least for Unix, DOS, Windows and Macintosh environments. Since many engineering areas are based on numerical mathematics, a set of toolboxes for engineering applications like DSP, neural networks and control, were added. These toolboxes contain a set of pre-defined Matlab routines for extra comfortability.

The programming language of Matlab is quite simple and easy to learn. It contains the basic control structures and makes use of existing and user-created libraries. It is also possible to use compiled C-routines when faster processing is needed. The latest versions of Matlab contain also development tools for graphical interface, but they are perhaps more suitable for smaller individual training experiments and demonstrations rather than a full CBE program. However, for example in the Macintosh environment, Matlab supports AppleEvents, which makes it possible to link Matlab to other software. An example of this can be found in (Pohjolainen et al, 1994). Also DSP education books with Matlab exercises are available, for instance (Burrus et al, 1994).

2.4.2 Mathematica

Where Matlab was specialized on numerical mathematics, Mathematica’s (Wolfram Inc.) key area has always been symbolic mathematics. From DSP viewpoint, symbolic mathematics gives extra means to teaching. Like Matlab, Mathematica is available for all the common environments. For a DSP engineer, Mathematica is a great development and theory testing tool. It is programmable but unfortunately has very limited capabilities for interface design. On the other hand, Mathematica documents can be presented as NoteBooks, hypermedia documents that can contain text, formulas, sounds, images and even animation (Laukkanen, 1991). These NoteBooks can be easily viewed with MathReader, a freely distributable runtime kernel. This way, stand-alone programs can be created. In Macintosh environment Mathematica also supports AppleEvents. At it's best, it can be used for astonishning demonstrations and excercises, that turn theory into practice. An example of using Mathematica for signal processing education can be found in (Evans & McLellan, 1993).

2.4.3. QuickSig

QuickSig (Karjalainen, 1990) is an object-oriented signal processing and algorithm development environment. In this work, a variety of CBE-functionality has been added into it. A more detailed discussion of QuickSig is presented in chapter 3.

2.4.4. Other DSP software

Other software that have been used for signal processing education are for example Ptolemy, a graphical block diagram programing environment from Berkeley University, and SPW - a commercial signal processing software (see Appendix F for additional information). Both of these are available for Unix-platforms. An example educational usage can be found for instance in (Lee, 1992).

2.5 Existing applications

At this point, commercial and stand-alone DSP-related CBE-applications seem to be missing. However, several books with CBE excercise and demostration sets do exist, some with bundled software, some relying on common DSP software. There are also some experimental software related to universities' signal processing education. Even a couple of signal processing education environments, for example SP Education laboratory (Wakefield et al, 1992) and VIP environment (McLean et al, 1992), have been developed.

2.5.1 Books with excercises and demonstrations software

Kamas and Lee (1988) have written a signal processing book especially for computer laboratory experiments in mind. The book includes a software “DSPlay” that students can use in their laboratory assignments. The book itself consists of several laboratory excercises and guidance. It is used as a course book at the Helsinki University of Technology. Another similar example is Mersereau and Smiths's computer laboratory textbooks (1992, 1994) also including special software for self-studying.

Another approach is Burrus et al's (1994) signal processing excercise book. It presents a wide variety of excercises to be performed with Matlab. The book was designed to be a teacher's aid in DSP education: excercices can easily be fit into courses to support textbooks and lectures. Since Matlab is available for several platforms, arranging the excercises can be done in a manner taking local circumstances into account.

2.5.2 Special signal processing education software

Naturally, there are several individual software used for signal processing education. However, there aren't too many of those with multipurpose or education environment nature. Perhaps worth mentioning are the tutorial visualization software in Santa Clara University (Wood, 1992), the SP Education laboratory in the University of Michigan (Wakefield et al, 1992) and the VIP environment especially for teaching image processing (McLean et al, 1992).

There is also a signal processing education package for Matlab called “SPCTools”. It is written by Dennis W. Brown from the Naval Postgraduate School. The package is available from Mathworks ftp site (see Appendix F).

2.6 Future

To the author, the future of signal processing education seems more and more computerized. The fast development of both computer hardware and software technology will certainly affect all education, but especially signal processing education. There are at least two good reasons for that: 1) the significance of signal processing as a field of science will increase due to development and demand in communications technology, multimedia and so on. 2) teaching signal processing with computers will become easier, more efficient and cheaper than it is now.

Due to increasing multimedia technology, there is no doubt that also commercial, self-study signal processing education software wouldn’t appear sooner or later. A big question is also the development of computer networks and especially Internet. This issue is discussed more thorougly in chapter 6. Already now the World Wide Web is used as a course information and organizing tool throughout the world. See for example (Orsak & Etter, 1995) and Appendix F.

3 THE QUICKSIG ENVIRONMENT

3.1 QuickSig in a nutshell

QuickSig is an object-oriented signal processing and algorithm development environment (Karjalainen, 1990). It provides signal processing tools for many application domains but it is also easily extendible to meet requirements of CBE since it runs on top of the Lisp language (Steele, 1990) and CLOS (Common Lisp Object System) (Keene, 1990).

Signals and related concepts are represented as objects and operations are typically implemented by method functions. A wide variety of DSP functionality, such as filtering, transforms (FFT), etc., are built-in features in QuickSig. An optional real-time DSP extension QuickC30 (Karjalainen, 1992), based on the use of the TMS320C30 processor, is also available. The full QuickSig software environment runs on Apple Macintosh computers. CD-quality (16-bit) stereo sound input and output are available on some models. Currently QuickSig supports four audio file formats: AIFF, binary, ascii (text) and matlab. The structure of the environment is presented in figure 3.

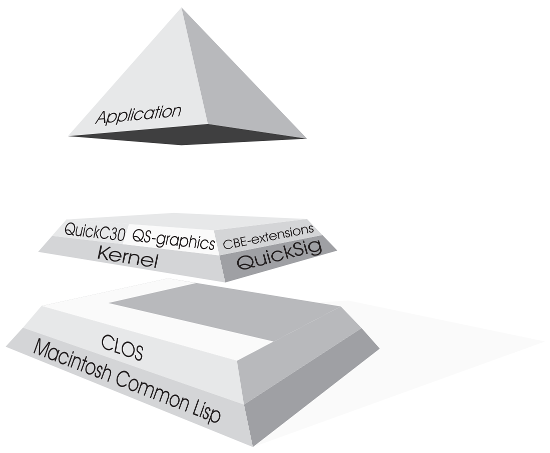


Figure 3: The QuickSig architecture (original layout (Helle, 1992))

3.2 QuickSig DSP features

The QuickSig is an object-oriented DSP environment, where signals are presented as class instances and DSP operations are typically implemented by method functions. The object formalism exhibits the clarity of abstract concepts in signal processing and symbolic manipulation of signals (Karjalainen et al., 1988). It is also the base for the QuickSig graphical user interface (GUI) (Helle, 1992). Nowadays object-oriented programming is indeed widely accepted as a standard way of creating large programs, especially when graphical user interface (GUI) is needed. The class hierarchy for a basic signal is presented in figure 4.

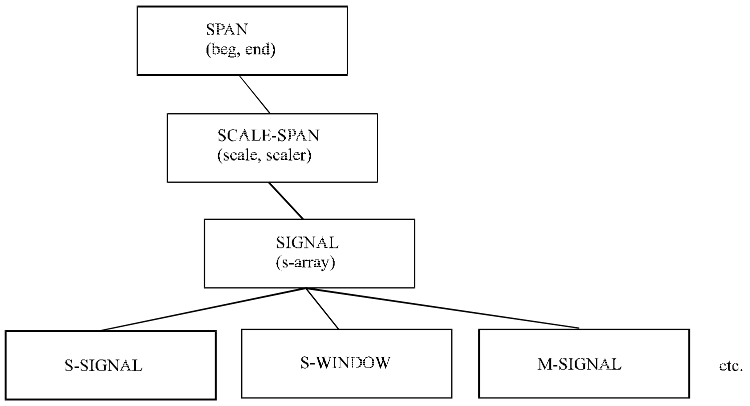


Figure 4: The class inheritance of signals

Basic DSP operations are realized by generic functions (globally named functions) that call the local method functions for that particular class. This retains the simple syntax of Lisp. For example, the function `add` meaning additive sample by sample mixing of signals, can be defined as:

```
(add sig1 sig2) => sig3
```

Figure 5 presents a summary of the signal object structure and DSP method functions.

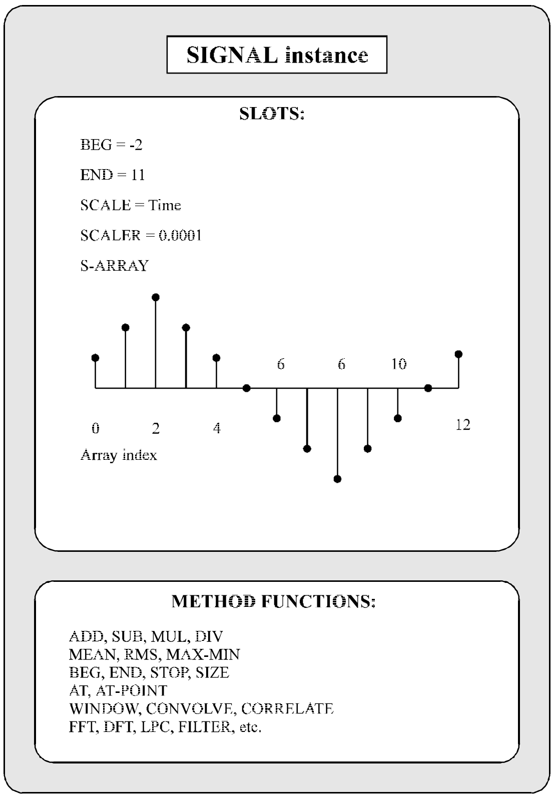


Figure 5: The structure of a signal instance (object) (original layout (Karjalainen, 1990))

3.2.1 C30 signal processor support

QuickC30 is an extension to the QuickSig basic system which provides support for the TMS320C30/C40 signal processors. This extension, quite unique of it’s kind, implements the full assembly language to those processors, emulates assembler on the fly and even provides object-oriented assembler programming. QuickC30 allows realtime algorithms to be developed and used within standard QuickSig operation (Karjalainen, 1992). For CBE purposes the QuickC30 extension could be used for example in the form of block diagram compilation and graphical editing of DSP algorithms (Karjalainen & Helle, 1988).

3.3 QuickSig graphical user interface (GUI)

QuickSig Graphics, the QuickSig graphical user interface (Helle, 1992), is a package including graphical presentations for signals and related objects as well as direct manipulation of them by mouse and keyboard actions. All the standard GUI functionality like easy creating and handling of windows, dialogs, buttons etc. is already present on the Lisp implementation used, MCL 2.0.1 and 3.0 (Macintosh Common Lisp), but QuickSig Graphics has these and a lot of other elements specially designed to support DSP development.

The basic object for graphically presenting signals, spectra, etc., is called a presentation (figure 6). It includes a lot of functionality of it’s own like zooming signals, selecting signal areas and so on. Other graphical elements such as pop-up menus, can be attached to presentations allowing object-specific behaviour.

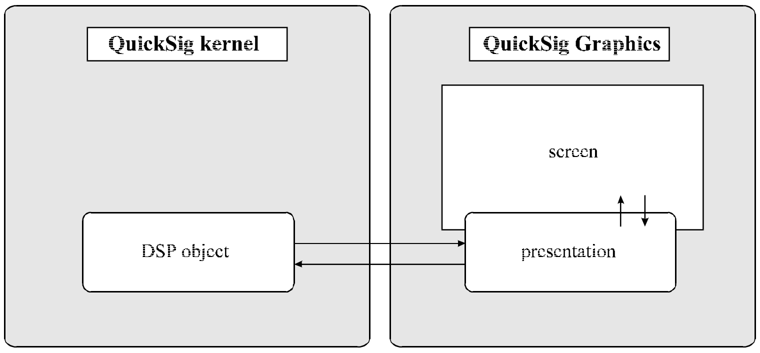


Figure 6: DSP objects and their presentations. The DSP objects may have several presentations simultaneously (original layout (Helle, 1992)).

A list of typical graphical elements is presented in table 2.

GUI elements	
window	A basic object that contains other elements except menubar and dialogs. By default, window is scalable and includes a close-box.
view	A window is divided into subelements called views.
dialog	Another basic object that is typically used in queries and error messages.
menubar	A horizontal list of menus on top of the screen
menu	A vertical list of items that may also be sub-menus
pop-up menu	A menu that appears when mouse is clicked on top of an object
scrollbar	Used for scrolling objects when they don’t fit into the visible area
presentation	A graphical presentation of an object like signal, spectrum etc.
button	A basic control that, when clicked, performs an action. Typically, different kind of buttons are used for different situations.
slider	Another type of control that can be used for example to pick up a value in a scale
help view	A text field on the bottom of windows showing help texts
page	A CBE or documentation window
HotArea	An area of a window that, when clicked, performs an action

Table 2: Typical GUI elements and their descriptions

3.4 QuickSig as an authoring tool

Within this project, QuickSig has been extended with CBE-oriented features. These features are called QuickCBE for convinience and to maintain the naming policy. QuickCBE could be understood as an authoring tool like HyperCard, ToolBook, etc., but especially designed for signal processing education. The system is a flexible programming environment for signal processing related CBE-applications. Since all basic signal processing operations can be easily used and new algorithms easily developed, and yet the graphical user interface (GUI) is totally under developer’s control, QuickCBE makes QuickSig a powerful CBE-tool as well. With MCL 3.0 it is also possible to create stand-alone applications independent of the Lisp itself, which of course means easily distributable applications.

There are a few drawbacks though: because QuickSig is under constant development, the documentation is only partial and the system is under limited distribution. These restrictions affect only developers though, final CBE-applications do not suffer from them.

3.5 Hypermedia features

3.5.1 The definition of hypermedia

Before getting to the hypermedia features it would be a good idea to first explain what is understood as hypermedia, hypertext and multimedia (Heimbürger, 1990). Especially the difference between hyper- and multimedia is crucial, even though the terms are widely used as synonyms.

Multimedia is a general term that relates to presenting information on a computer. Multimedia stands for computer documents that combine different kinds of information. The information can be any kind of information that can be presented with a computer: text, images, sound, video etc.

Hypertext is an older term describing computer documents where textual infomation can contain links to other textual information. These links can be easily followed by clicking a mouse, for example, and the reading process thus becames nonlinear. That is why it is sometimes called nonlinear writing. Actual hypertext documents are nowadays seldomly seen, hypermedia has replaced it.

Finally, hypermedia means simply multimedia containing an associative data structure similar to hypertext. That is, hypermedia stands for computer documents that combine different kinds of information and contain links to other information. Thus hypermedia is multimedia, but multimedia is not necessarily hypermedia.

In CBE, the commonly used term is hypermedia even though it seems not to be fashionable at the time. The reason is that the associative data structure describes the way we think and moreover, the way we learn! Since multimedia does not necessarily contain that structure, the term hypermedia is more precise. However, the use of different kinds of information in the structure is of course important as well and, on the other hand, the associative data structure is not present in every CBE-application.

Lately, the term multimedia has been freely used to mean all of the above.

3.5.2 GUI extensions

QuickSig Graphics, the graphical user interface package in QuickSig, has been extended to support also CBE applications. A display window may easily be composed of signal presentations, various controls (such as buttons and sliders), texts and figures by using a layout script syntax. A code example for the “Home Page” presented in figure 11 has been included as a reference to illustrate the syntax (Appendix C). The behaviour of graphical and other GUI elements is stricktly under developer’s control. For instance, use of keyboard and mouse to produce a pop-up menu in a signal presentation can easily be enabled or disabled. Together with

HotAreas (areas on the screen that, when clicked, perform actions, for example open a new window) this forms the hypermedia base.

3.5.3 HotAreas

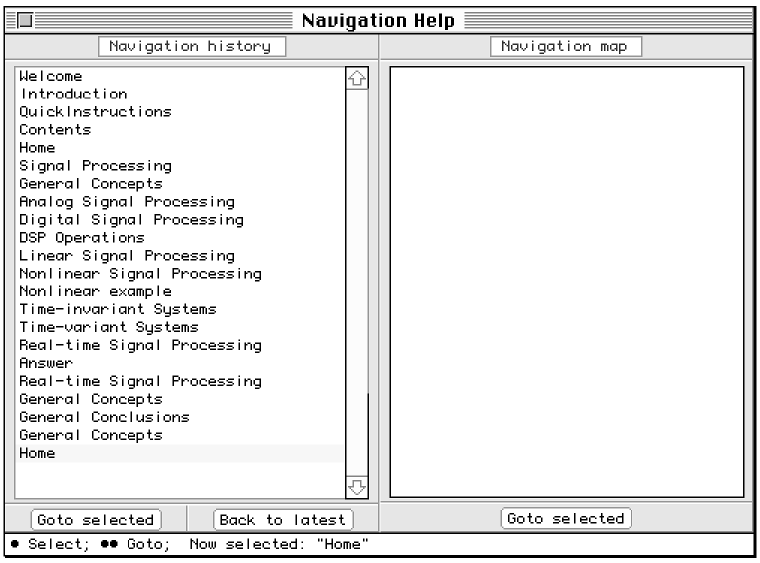
HotAreas are areas on the screen that, when clicked, start an event. This event can be practically anything: opening a new window, filtering a signal and displaying the result, a file operation etc. The basic use of these areas is that when the cursor is moved onto a HotArea, a help text will appear and when clicked, a new window is opened and the previous one closed. The action is similar to using WWW-browsers. There are no technical reasons, however, for not creating any other kind of action. The implementation of HotAreas restricts to use them only with certain kinds of graphical presentations but, in practice, this should be considered as a desired feature rather than a limitation. The creation and handling of HotAreas is very “developer-friendly” as can be seen from the code example of the “Home Page” in figure 11 (see chapter 4, Appendix C).

Perhaps more common term in the literature is “HotWord”, which means the same. However, from developer’s viewpoint, the term “HotArea” describes the nature of the element more precisely.

3.5.4 Navigating in hyperspace

Navigating in hypermedia is an extremely difficult thing (Rivlin et al. 1994). An associative structure, that can also be dynamic(!), is very difficult or even impossible to visualize or describe. For a predefined set of visible or hidden nodes, the structure can be an n-dimensional hypercube. If the structure is, however, allowed to change in time, i.e., existing nodes to vanish and new nodes to appear, the visualization can only, if at all possible, be done at certain point of time. Still, when thinking of CBE, navigating is important: the user must know where he or she is going to and how to go forward or backward. If learning would only be constant digging up of new things in cyberspace, there would be absolutely no way of controlling the learning process. And for CBE, we definitely do not want that.

At this point, a simple navigation tool (Figure 7) has been added to the system. The tool provides a navigation history for the user, pretty much similar to the history functions in WWW-browsers. It allows the user see where he or she has already been and to go back wherever desired. In the future, there may even be a graphical map that could be based for instance on dividing the CBE-hyperspace into levels or areas, but it is not possible to create a graphical map for all applications.



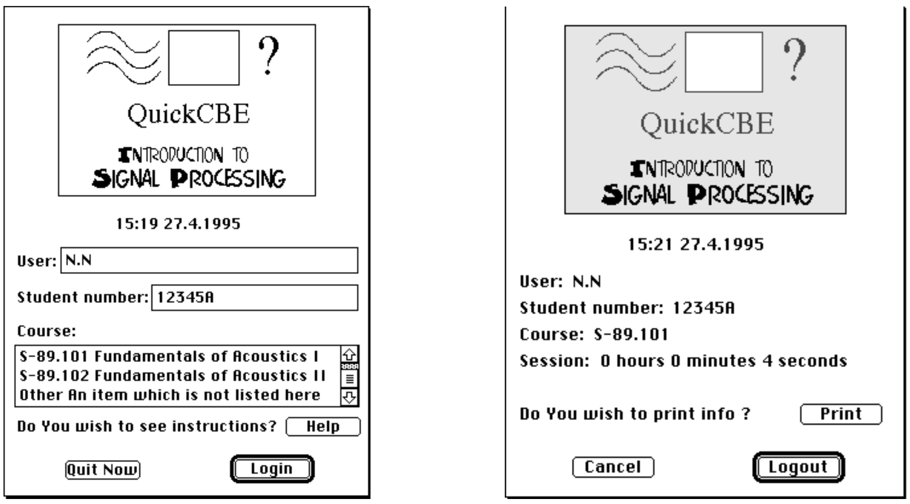
```
Welcome
Introduction
QuickInstructions
Contents
Home
Signal Processing
General Concepts
Analog Signal Processing
Digital Signal Processing
DSP Operations
Linear Signal Processing
Nonlinear Signal Processing
Nonlinear example
Time-invariant Systems
Time-variant Systems
Real-time Signal Processing
Answer
Real-time Signal Processing
General Concepts
General Conclusions
General Concepts
Home
```

```
• Select; •• Goto;  Now selected: "Home"
```

Figure 7: Navigation tool

3.5.5 User Surveillance

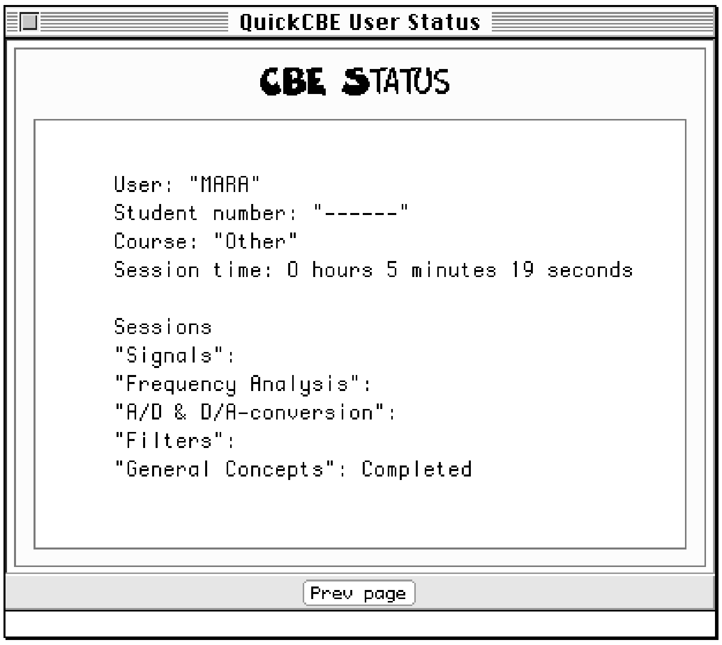
A good CBE or hypermedia application keeps track of user’s actions and changes it’s behaviour according to this information. QuickCBE observes the user on two levels: it keeps track of pages and sessions (groups of pages). Both the information of the pages and the sessions opened and closed is used for navigation purposes and is also written into a log file. For the surveillance, it is important to identify users: this is done with special login and logout dialogs (figure 8). This allows also later analysis of the behaviour of users and programs. The sessions are implemented using special class-slots thus being nice and easy for a developer to control. The user can also check his or her session status any time (figure 9)



15:19 27.4.1995
User: N.N
Student number: 12345A
Course:
S-89.101 Fundamentals of Acoustics I
S-89.102 Fundamentals of Acoustics II
Other An item which is not listed here
Do You wish to see instructions?

15:21 27.4.1995
User: N.N
Student number: 12345A
Course: S-89.101
Session: 0 hours 0 minutes 4 seconds
Do You wish to print info ?

Figure 8: Example login and logout dialogs.



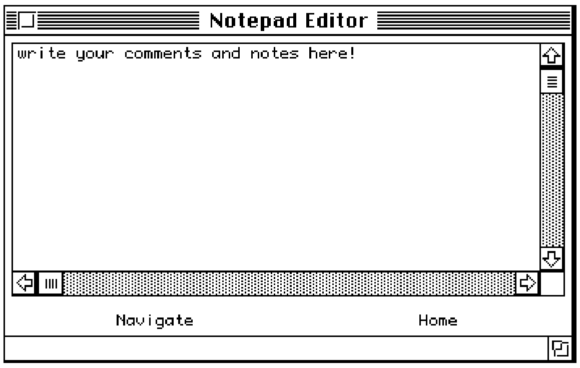
```
User: "MARA"
Student number: "------"
Course: "Other"
Session time: 0 hours 5 minutes 19 seconds

Sessions
"Signals":
"Frequency Analysis":
"A/D & D/A-conversion":
"Filters":
"General Concepts": Completed
```

Figure 9: A status window..

3.6 Other CBE features

A notepad editor gives students the possibility of making their own comments while using an application (figure 10). These new features not only make QuickSig a good authoring environment, but can also be used for documentation and other applications as well.



write your comments and notes here!

Figure 10: A NotePad Editor gives a student a possibility of making his/her own comments while experimenting with the CBE-application.

3.7 QuickSig application areas

QuickSig has been succesfully used with many DSP-related problem domains like psychoacoustics, speech processing, physical modelling of musical instruments, and auralization. Many of them could be applied for educational purposes as well. For instance, a student can measure his or her frequency and time-domain masking curves (Karjalainen & Rahkila, 1995). An articulatory speech synthesizer shows the vocal tract profile where the user can move the articulators and hear the generated voice (Välimäki et al., 1994), (Välimäki et al., 1994). An object-oriented database (Karjalainen & Altosaar, 1993) gives an intuitive graphical access to transcribed and stored speech signals. Model-based sound synthesis of musical instruments (Karjalainen & Välimäki, 1993), sound effects used in effect processors, and real-time simulation of room acoustics combined with headphone auralization (Huopaniemi et al., 1994) are other closely related topics.

Besides education many of these new features can also be applied to other purposes as well. For example the “HotAreas” can be used for improving the graphical user interface or for building a help system into a specific signal processing application. As an ongoing, long term project, an on-line documentation of the QuickSig system itself is constructed using these new hypermedia features.

4 CBE-APPLICATION "INTRODUCTION TO SIGNAL PROCESSING"

"Introduction to Signal Processing" is a CBE-application that is designed to give 2nd-3rd year university students a practical and yet general enough view on signal processing especially from the point of view of audio and speech signals. The approach is practical, emphasizing applications. A similar approach has been used in undergraduate education for example by Allebach et al. (1994), with success. Some "heavy" mathematics is included, but only as extra material. In this tutorial step it is more important to explain concepts with examples and exercises. The "inside theory" of signal processing is left for more specific courses.

4.1 Target group and goals

In the last few years signal processing education has become a more and more important area of teaching at the Helsinki University of Technology. The number of students specifying in this area has steadily grown. Signal processing education also has a wider interest as being used in many other fields of science. In teaching acoustics, it became clear that signal processing education has to be included in the courses starting from the very first course: Fundamentals of Acoustics I. This course is followed by 2nd and 3rd year electrical engineering and computer science students along with students from other departments and even other universities. The course has a tutorial nature, so including a tutorial in signal processing within a tutorial in acoustics would fit in smoothly. A typical number of students taking this course is 130 students per year. Since the amount is quit large, it is important to use efficient teaching methods in order to maintain the quality of education.

The goal for this CBE-application was therefore to give these students a tutorial in signal processing in a practical way. The focus was to explain the very basic concepts of signal processing using examples and excercises and leave the theoretical side for later courses. That way, the students could form a general view on signal processing that would certainly help them in understanding the theory. Also, audio signals are a very good choice for examples, because the results and changes can be easily heard. However, the topics of this application are of general nature and therefore hold for other signal processing as well. As an author’s opinion, audio signal processing can definitely bring life into theory.

4.2 Contents

4.2.1 Contents decisions

“What concepts should be included?” was the very first task in designing the application. At the Helsinki University of Technology there are three laboratories giving signal processing education: the Laboratory of Acoustics and Audio Signal Processing, the Laboratory of Digital Signal Processing and Computer Technology and the Laboratory of Information Science. In order to cover all laboratories mutual interests, a small query and survey was performed. Based on this information the decisions of the contents were made. As a conclusion, opinions in different laboratories were quite similar.

A set of signal processing tutorials was also examined in order to find help and assistance for designing the contents and excercises. Most of them were already mentioned in chapter 2. In addition to those, at least Smith’s (1989) article and Karu’s (1995) book deserve credits.

4.2.2 Final contents

The key to this CBE-application is the model illustrated in figure 11. It serves as a base for the application, being simultaneously a system model of (audio) DSP and the

"Main Menu". Each block is a Hot Area that, when clicked, leads to an independent session of that subtopic.

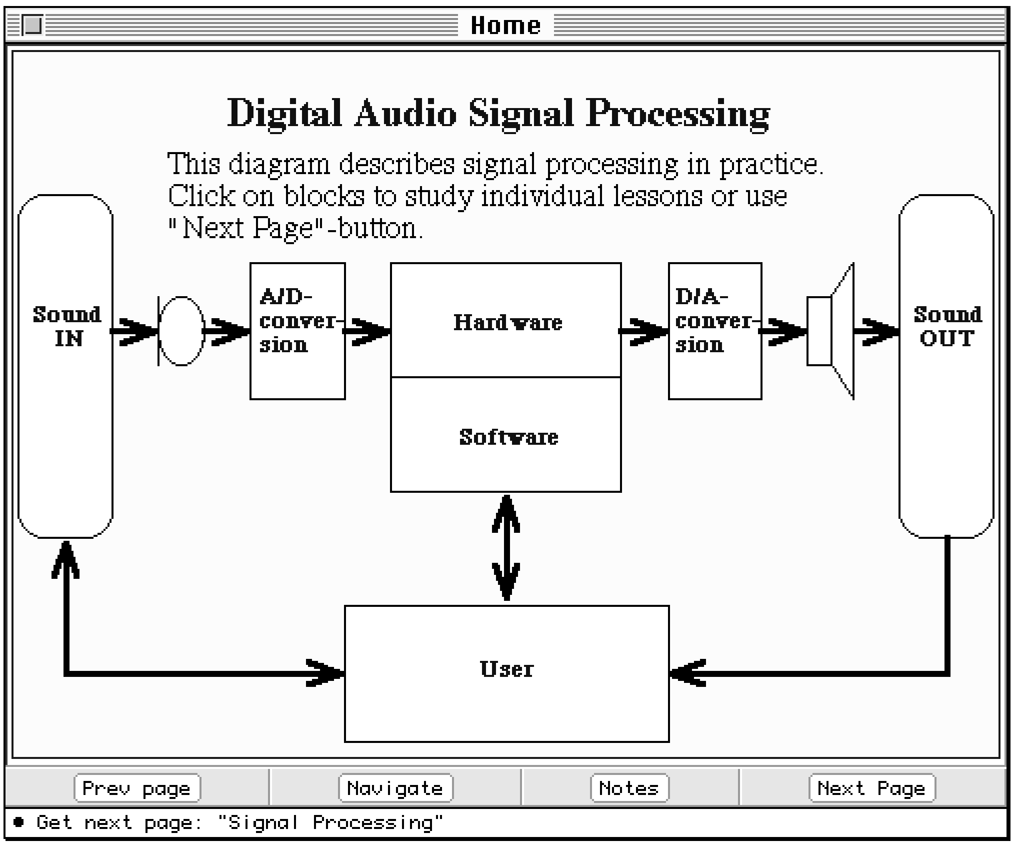

Home
Digital Audio Signal Processing
This diagram describes signal processing in practice. Click on blocks to study individual lessons or use "Next Page"-button.
Sound IN
A/D-conver-sion
Hardware
Software
D/A-conver-sion
Sound OUT
User
Prev page
Navigate
Notes
Next Page
● Get next page: "Signal Processing"

Figure 11: The system model for CBE-application "Introduction to Signal Processing", including the "Main Menu" view and navigation buttons.

The subtopics covered by the application are:

• Signals, their properties and representation (graphical presentations, numerical and function

representations).

• Spectrum Analysis: spectrum and its properties, calculating spectrum, and windowing.

• A/D and D/A-conversion: sampling, quantization, and coding.

• Filters: impulse response and transfer function, convolution, and digital filters.

• Signal Processing: general principles (linear/nonlinear, analog/digital etc).

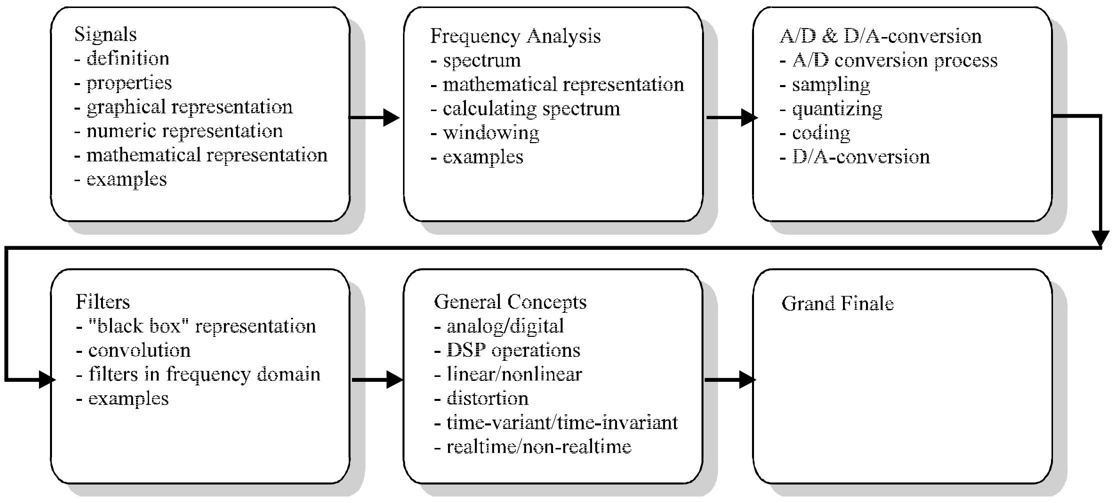


Figure 12: Application contents and default path

4.2.3 Default path

Each subtopic is presented by one or several pages of concept explanation, examples, and exercises. Students can study these subtopics in any order they wish and they can also use the navigation tool for determining individual study paths (figure 7). However, a default path, that is, an order for the topics, was also created (figure 12). This path could be followed by clicking the “Next page”-button on the lower right corners of the windows. The default path is similar to the order of the subtopics in the previous paragraph. The reason for this path was to make sure that the user never would be in a situation when he or she wouldn’t know where to continue. The program also checks that all topics will be covered, the user does not reach “the end” until so. This is done by placing a missing subsession start into the “Next Page” button at the “Home Page”.

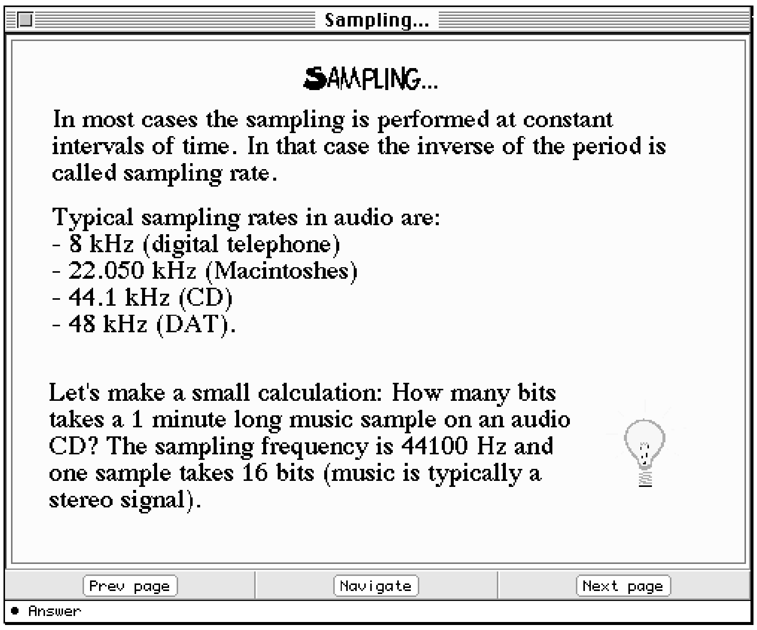
4.3 Graphical User Interface

The application was designed to be as easy to use as possible. Therefore it has a fully graphical interface with standard Macintosh-like operation. The GUI includes a standard menubar, dialogs, and buttons that are easy and intuitive to use. No extra help system is thus needed. Within certain material other control elements like sliders have also been used, but their usage is quite obvious (see for example figure 15).

The QuickSig system itself has a large set of special keyboard and mouse controls, but most of them have been disabled in this application. This was to make sure the users don’t end up in trouble because of undocumented “tricks”. However, some features are still available, for example it is possible to scale (zoom) the signals in their windows.

4.4 Interactive excercises

The excercises in this CBE-program are of two basic types: “question-like” tasks and “try it”-like tasks. “Question-like” tasks require only activity of the brain, that is, something to think about. An example of such is presented in figures 13 and 14. Typically a student is first given some information and then asked something based on that information. After having thought about it, the student can click and see the answer.



Sampling...

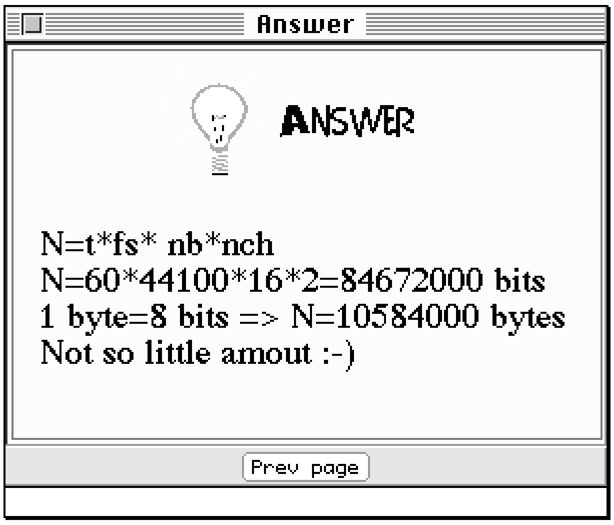
SAMPLING...

In most cases the sampling is performed at constant intervals of time. In that case the inverse of the period is called sampling rate.

Typical sampling rates in audio are:
- 8 kHz (digital telephone)
- 22.050 kHz (Macintoshes)
- 44.1 kHz (CD)
- 48 kHz (DAT).

Let's make a small calculation: How many bits takes a 1 minute long music sample on an audio CD? The sampling frequency is 44100 Hz and one sample takes 16 bits (music is typically a stereo signal).

Figure 13: A “question-like” excercise...



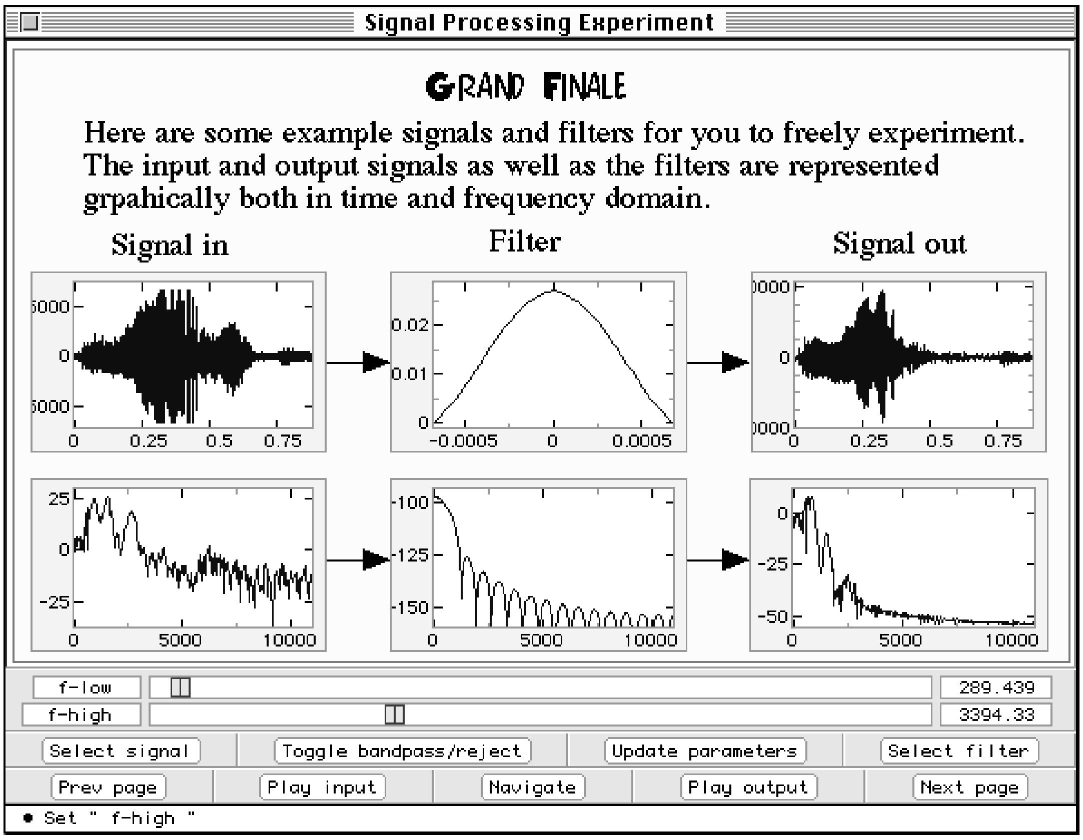
N=t*fs* nb*nch
N=60*44100*16*2=84672000 bits
1 byte=8 bits => N=10584000 bytes
Not so little amout :-)

Figure 14: ...and the answer.

Another type of excercise is a “try it”-like task where a student is encouraged to freely experiment with the concepts. Normally such a task would follow a larger amount of information. The purpose is to summarize the information and to give a possibility to apply the things just learned. As an example, in figure 15 there is an excercise that summarizes the whole contents of the program. The page “Grand Finale” activates student to experiment in practice with concepts like signals, filters etc. Code for this experiment is presented in Appendix D.

4.5 Log-system

The application uses a log-system to gather information on users and usage. Login and logout dialogs (see chapter 3) are used for identifying users. The log-file is then updated constantly while using the program. Currently the log-file keeps track of time of opening and closing windows with their attributes like session marks etc., which allows a full analysis of the usage to be performed, for instance individual study paths. The purpose of the log-file is to be teacher’s debugging aid: analyzing usage is of great help when evaluating the program, searching for errors and for further development. Chapter 5 presents an example analysis that was done when the application was used within a course for the first time.


GRAND FINALE

Here are some example signals and filters for you to freely experiment. The input and output signals as well as the filters are represented grpahically both in time and frequency domain.

Figure 15: An interactive excercise that summarizes signal processing concepts in a practical way.

4.6 Other target groups

The primary target group for this application was the Helsinki University of Technology students participating in the course “Fundamentals of Acoustics I”. Because of its general nature, the application has waken an interest also in other universities. For example, the Phonetics Department, University of Helsinki, has been using the program as teaching material for their students and researchers. The Sibelius Academy has also expressed their interest in using it. The program should not, however, be considered suitable for everyone, there is a certain technological way of thinking behind the material.

4.7 “Introduction to Signal Processing” as a product

The application is not ment to be a product in a commercial sence. It is a stand-alone program in a way that no other software is needed to run the program but it is an “image file”, a program that includes the MCL interpreter. This causes also licencing problems if distributing it freely. With MCL 3.0 it is possible to create “real” stand-alone programs, but within this work there has been no time for it. A free distribution would also require an installer in order to take the requirements of different environments into account.

Currently, “Introduction to Signal Processing” is only available for Macintosh platform. It takes about 5 Mb of hard disk space and a minimum of 16 Mb of RAM memory. Future development of the program is discussed in chapter 6.

5 USER FEEDBACK ANALYSIS

5.1 Context

The analysis presented is based on the statistical information gathered while the application was used in the course “Fundamentals of Acoustics I” in autumn 1995. The students participating in this course are typically 2nd or 3rd year university students majoring in electrical engineering or computer science. Typically, there is also a minority of students from other faculties of the Helsinki University of Technology or even from other universities such as the University of Helsinki, the Sibelius-Academy and the Helsinki University of Arts. Hence the majority of users were technologically oriented students with various amounts of previous experience of signal processing. Previous knowledge of signal processing was estimated to be quite small based on the survey among education staff and the fact that the first years in the Helsinki University of Technology are spent mainly with mathematics and physics.

The use of the application was organized in the following manner: one Macintosh FX computer equipped with good loudspeakers was placed in a quiet room for the time of the course period. The students reserved time, one hour at a time, for using the application. One hour was alleged to be sufficient, what it indeed was. The usage was on a strickly volunteer basis but users were promised some extra credit for their course evaluation. Students were also informed about the application during the first course lectures. After using the application, students were encouraged to fill a query form for analysis purposes. This, along with the information in the log files, was analyzed and the results were presented to the students.

The users were given only a little guidance before usage. Only the experiment background, the goals and very short instructions of program usage were presented beforehand. This arrangement was in order to evaluate the “easy to use”-nature of the program.

5.2 Analysis

Statistical analysis was performed after the first time the program was used within the course “Fundamentals of Acoustics I”. The analysis was done on both the log file information and the query forms. The idea was to measure the program and the context mainly for debugging and future developing in mind. The logged information would give a nice way of measuring the program itself and the query forms would supply information on how the using of the program was organized and what kind of thoughts the experiment has given to the students. This would surely help the educational staff next year when the course is held again.

Before the analysis, the log-file was checked for possible error situations. When found, the situation was analyzed and after that, removed from the file. That was to make sure that the statistical results would be as accurate as possible. Typical error situations were due to mistyping in the login dialog that forced the user to logout and login again. Some of the errors also related to unexpected program crashes.

5.2.1 Usage time

The histogram in figure 16 represents time versus the number of users. It clearly shows that the estimated using time was excactly right. The mimimum result 3 minutes tells that the user has not really operated on the program but instead just clicked his or her way through the program as fast as possible. The maximum result on the other hand shows an interest and ethusiasm towards the subject. The average, 30.31 minutes, falls right into the estimated usage time of 30-45 minutes.

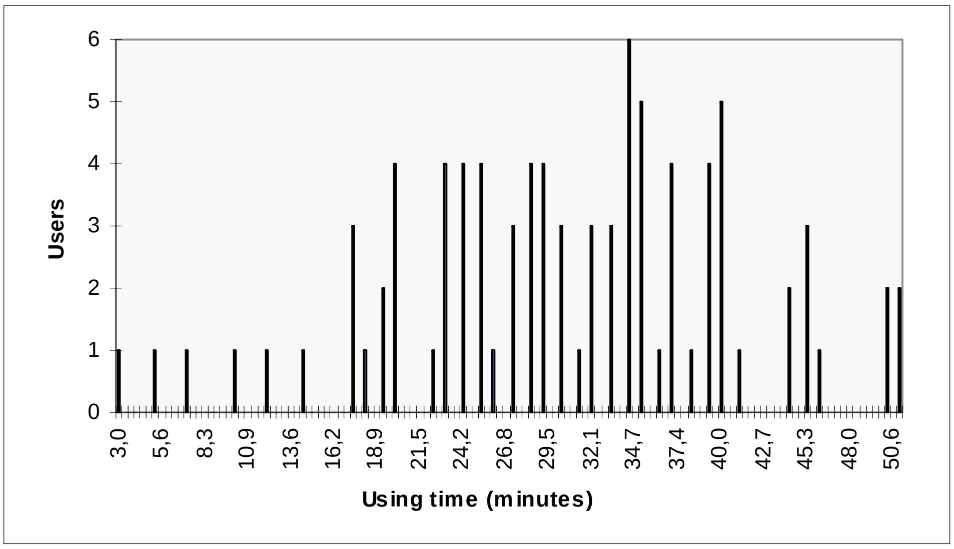


Figure 16: The distribution of time in application usage.

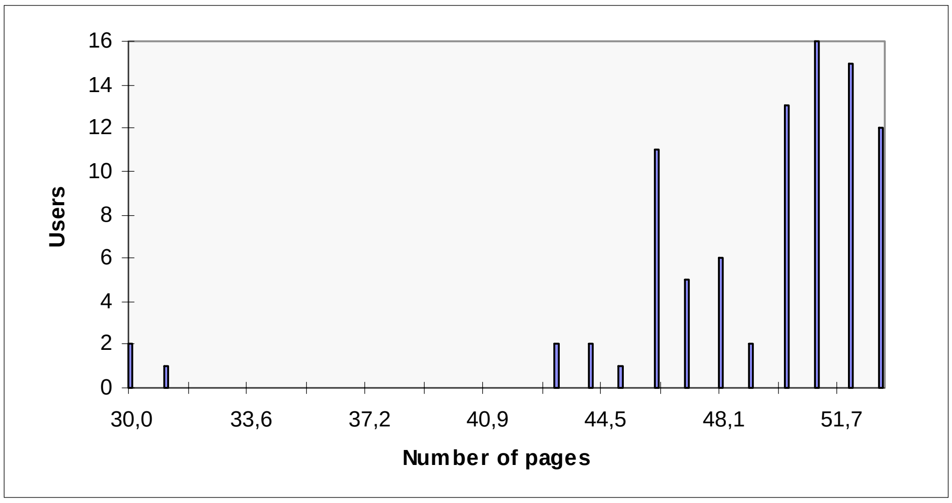


Figure 17: The number of pages per user.

5.2.2 Number of pages per user

Since the default path did not include all possible pages, it was interesting to find out how many pages the users would go through. It turned out that no-one "found" all pages, but most of them anyway. The minimum result is 30, which tells that the user has not gone through the whole application. The default path consisted of 52 pages. The average was 49,1, which relates to a known memory problem that would cause the program to crash within the last few pages and thus making some error to the log information. The histogram in figure 17 shows the results.

5.2.3 Time spent on a page

This analysis was done to see if there were pages that would take extraordinary long time to finish or pages that would be passed right away. It turned out that only one page took signicantly longer time than the others and that was designed so. The page “Grand Finale” (figure 15, chapter 4) contains an interactive demonstration of signals and filtering them and it serves as a summary for the application. The average time spent on one page was about 40 seconds and the shortest time, about 6 seconds, was spent on a page “Non-real-time signal processing”. From pedagogical viewpoint it is important that there are also “relaxing” pages, i.e. pages, that do not contain too much information. The page with shortest visiting time falls into that category. Within this application, there were also other “relaxing” pages. The histogram in figure 18 presents the results.

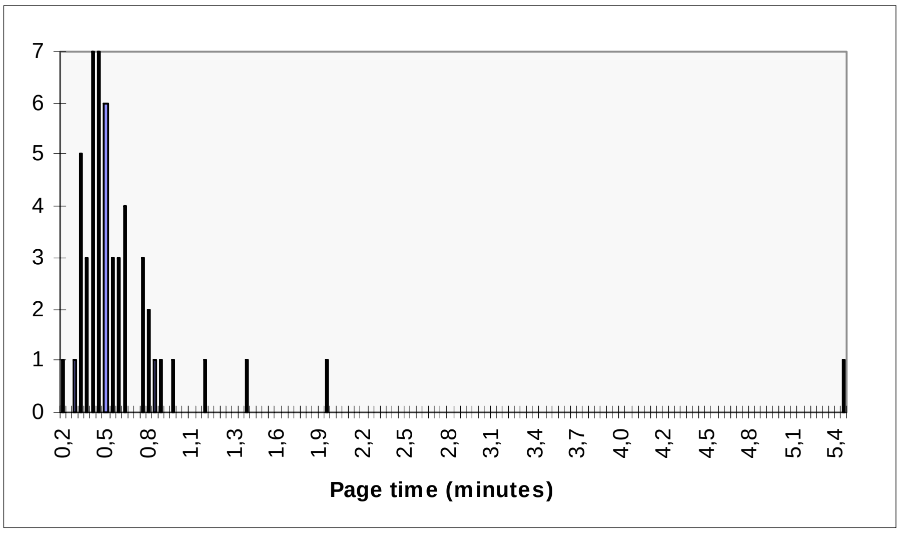


Figure 18: The time spent on a page.

5.2.4 The query form

The summary and the English translation of the query form is presented in appendix E. Interesting results are that the students had quite a good knowledge on the subject already and that majority would have used the program even without the credits. The query clearly showed, however, that every user had learned something while using the program. What they did learn depended on their previous knowledge and varied a lot, but for example filters opened up in a different way with this program. As a curiosity, many students learned that signal processing indeed requires computational power and the machine was considered quite slow. Many of them also liked the practicality of the program. This should be understood as a success. The students also felt that this kind of education gives a good alternative to ordinary methods and they were very excited about using the program. Surprisingly, the students paid only little attention to small bugs and error situations.

5.3 Problems

Based on this analysis, no major problems or bugs where found. A few minor ones were discovered though. First of all, the login dialog (see figure 8, chapter 3) was designed to have a “default-button”, that is a button that could be activated either by a mouse click or by pressing “Enter”. Unfortunately, many users typed in their name and then pressed “Enter”. This would cause the login dialog to finish and start the program, but lacking the student number. The students immidiately noticed that and logged out and in again. Several “false logins” appeared into the log-file because of that. They were removed from the file before analysis. The right way to design a login dialog would have been either without the default button or with an extra information check routine.

Secondly, the application suffered from crashes within the last few pages. This was due to memory problems and was known already before. The reason was basically bad code including several global variables that required memory managing. The problem was solved by installing a total of 32 Megabytes of RAM into the machine used.

Lastly, not a problem but rather misthinking was that the students had much more previous knowledge on the subject than expected. The contents of the application were chosen by assuming that the users would have no previous knowledge. Some of them didn't but majority did and thus the contents could have been expanded a little. However, the analysis results were discussed with the students and many of them regarded the application as a good retention. In that sense, the contents decisions were also successful.

5.4 Summary

As a conclusion to the user analysis I would say the results were positive. This analysis clearly showed that the effort to teach signal processing concepts in an alternative way has been succesful. The students used and liked the CBE-program "Introduction to Signal Processing" but the effect of the extra credits they were given remains a little bit unclear. Based on the query form, the extra credits would have been unnecessary but on the other hand, the queries were filled only after students already knew they were going to have extra credits for using the program. Next year, when the course is held again, there propably won't be any extra credits, only a strictly voluntary usage.

Also, the primary goal of this CBE-program, to teach signal processing concepts, was achieved. Based on the analysis presented here, the students did learn the topics and found the CBE way of learning as a good alternative to standard methods.

6 CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Computer based education is an area of wide interest throughout the world. CBE can be used in teaching almost anything, however, this is not always desirable or succesful. Thus the rule number one for CBE is that applying CBE methods for teaching must be, in some way, more useful that ordinary teaching methods. Another principle is that a CBE application must teach the subject, not the program itself. In order to fulfill these principles, using some CBE design method is strongly adviced. Within this work, two methods have been used: the Lifländer design method and the Lessonware model.

Signal processing education is a special case in CBE. Especially in DSP, all the work in practice is done with computers, so it is a natural topic for CBE. However, signal processing applications typically demand a lot of computational power, which of course limits CBE methods. However, modern multimedia computers have enough capacity for this kind of purposes as well and in the future computational capacity is no longer such a major limitation. With CBE methods, signal processing education can indeed be improved within the scope of quality, efficiency, and costs.

QuickSig is a multipurpose signal processing environment developed specially with audio signal processing in mind. The system has now been extended with multi- and hypermedia features thus making it a flexible signal processing CBE tool. Such tools are hard to find since the usual authoring tools do not provide ready-made signal processing functionality. The CBE extensions include improved graphical user interface with HotAreas and various controls, a navigation tool and a notepad editor for making applications usage more convenient, and a logsystem for providing information for authors. Naturally, these features can be used for other purposes as well, for instance for improving the user interface or creating on-line documentation and help for standard signal processing applications and demonstrations.

As a case example, a CBE application “Introduction to Signal Processing” was built. Designed to be a tutorial, the application explains the basic concepts of signal processing especially from the viewpoint of audio and acoustics. The application is a self study package including all the information, excercises and examples in one stand-alone program. The pedagogical approach is practical, emphasizing applications and “real-life” examples. The theory and “heavy mathematics” of signal processing are left for more spesific courses. Should theory come first, the big question in teaching signal processing, remains, but based on experiences with this work I would vote against it. Practical examples motivate students in a totally different way than plain theory and also activate them with their theoretical studies as well.

The application was used in autumn 1995 within a course “Fundamentals of Acoustics I” at the Helsinki Univesity of Technology. Nearly one hundred students used the program and feedback was indeed positive. Log-file from the usage was saved and a query form was given to students. A statistical analysis was then performed on that feedback information. The analysis results clearly show that the experiment was a success, students liked the program and no major problems were found. The benefits of using the application will evidently show in other courses.

6.2 Further development

With this work, the aim has been of tutorial kind. By a hands-on-like approach the student learns the functionality and operability of the basic concepts, resembling experimentation with ordinary (physical) equipment. The next step after the tutorial level should be to learn the role of a constructor (and straightforward designer). The best way, already widely utilized, is the block-diagram (or patch editor) approach, where the user has DSP operations as building blocks that may be wired into dataflow diagrams. An experimental patch editor based on QuickC30, a real-time execution environment for the TMS320C30 processor, has already been built with the QuickSig system. With the advent of fast RISC-based computers a single machine will be able to do the whole task in real time.

The final step in a signal processing CBE-program should be to integrate mathematics, already known from math courses, to the tutorial and constructional levels, in order to educate a master designer in signal processing. To support this level, integration of mathematics toolboxes, such as Mathematica and MATLAB, to QuickSig CBE is needed. Since all these programs can make use of AppleEvents, the task should not be too difficult. Still, at this point, the integration is left for the future.

Another aspect in further development is to move from the single machine usage to network. The ultimate goal for a CBE system would be to have it available on-line for everyone through, for example, the Internet. An outragious development is currently going on with WWW and related Internet technologies all over the world. Java, real-time video conferencing and network audio, the current hot topics of Internet, are likely to make truly interactive education possible. For example, an on-line CBE server would open totally new possibilities for signal processing education.

REFERENCES

Allebach J.P.; Bouman C.A.; Zoltowski M.D. 1994. Digital signal processing with applications: A new and successful approach to undergraduate DSP education. ICASSP '94, Adelaide, South Australia, April 19-22, 1994. IEEE. pp. 49-52.

Burrus, C.S.; McLellan, J.H.; Oppenheim, A.V.; Parks, T.W.; Schafer, R.W.; Schuessler, H.W. 1994. Computer-Based Excercises for Signal Processing using Matlab. Prentice-Hall. 404 p.

Bush, V. 1945. As we may think. Atlantic Monthly. 176, 1, pp. 101-108.

Crossley K.; Green L. 1985. Designing Computer Lessonware: A Practical Guide for Teachers. Crossley & Green.

Engeström Y. 1987. Learning by Expanding - An Activity-theoretical Approach to Developmental Research. Helsinki. Gummerus. 368 p. *In Finnish*

Evans, B.L.; McLellan, J.H. 1993. Investigating Signal Processing Theory with Mathematica. ICASSP'93, Minneapolis, Minnesota, USA, April 27-30, 1993. IEEE. pp. I-12, I-15.

Heimbürger, A.; Alkula R.; Kuhanen, T. 1990. Hyperteksti ja hypermedia. Valtion teknillinen tutkimuskeskus, Tiedotteita 1154. Espoo. 207 p. *In Finnish*

Helle, S. 1993. Graafinen käyttöliittymäohjelmisto digitaalista signaalinkäsittely-järjestelmää varten. Teknillinen korkeakoulu, sähkötekniikan osasto, akustiikan ja äänenkäsittelytekniikan laboratorio. 63 p. Lisensiaatintyö. *In Finnish*

Huopaniemi J.; Välimäki V.; Karjalainen M.; Huotilainen T. 1994. Virtual Instruments in Virtual Rooms – A Real-Time Binaural Room Simulation Environment for Physical Models of Musical Instruments. ICMC-94, Aarhus, Denmark, September 12-17, 1994. ICMA. pp. 455-462.

Kamas A.; Lee E.A. 1989. Digital Signal Processing Experiments. Prentice-Hall, 1989. 103 p.

Karjalainen M. 1990. DSP software integration by object-oriented programming: a case study of QuickSig. IEEE ASSP Magazine, 7, 2, pp. 21-31.

Karjalainen M. 1992. Object-oriented programming of DSP processors: a case study of QuickC30. ICASSP'92, San Francisco, CA, USA, March 23-26, 1992. IEEE. pp. 601-604.

Karjalainen, M.; Altosaar, T. 1993. An Object-Oriented Database for Speech Processing. EUROSPEECH-93, Berlin, Germany, September 21-23, 1993. ESCA. pp. 183-186.

Karjalainen, M.; Altosaar, T.; Alku, P. 1988. QuickSig - an object-oriented signal processing environment. ICASSP’88, New York, NY, USA. IEEE.

Karjalainen, M.; Helle, S. 1988. Block diagram compilation and graphical editing of DSP algorithms in the QuickSig system. ISCAS’88, Espoo, Finland.

Karjalainen, M.; Rahkila, M. 1995. Learning signal procesing concepts and psychoacoustics in the QuickSig DSP environment. ICASSP'95, Detroit, Michigan, USA, IEEE. pp. 1125-1128.

Karjalainen M.; Välimäki V. 1993. Model-Based AnalysisSynthesis of the Acoustic Guitar. SMAC-93, Stockholm, Sweden, July 28 - August 1,1993. The Royal Swedish Academy of Music. pp. 443-447.

Karu, Z.Z. 1995. Signals and Systems made ridiculously simple. ZiZiPress. 122 p.

Keene, S.E. 1989. Object-oriented Programming in Common Lisp. Addison-Wesley. 266 p.

Laitinen, A. 1988. Tietokoneavusteinen opetus. Valtion painatuskeskus. Helsinki. 82 p. *in Finnish*

Laukkanen, P. 1991. Johdatusta Mathematica-ohjelmiston käyttöön. Jyväskylä. Jyväskylän yliopisto, matematiikan laitos. Luentomoniste 19. 122 p. *in Finnish*

Lee, E. 1992. A Design Lab for Statistical Signal Processing. ICASSP’92, San Francisco, California, USA, March 23-26, 1992. IEEE. pp. 81-84.

Lifländer V-P. 1988. A Design Method for A Computer Based Learning Environment. *ADCIS 30th conference proceedings*, Philadelphia, Pennsylvania, pp. 259-264.

Lifländer V-P. 1989. Tietokoneavusteisen opetuksen kehittäminen. Helsingin kauppakorkeakoulun julkaisuja D-112. 233 p. *in Finnish*

Lifländer V-P. 1994. Tietokoneavusteisen opetuksen suunnittelu ja toteutus. Espoo, Finland. 18 p. *coursematerial, in Finnish*

McLean, G.; Jernigan, M. 1992. A Software Environment for Teaching Image Processing. ICASSP’92, San Francisco, California, USA, March 23-26, 1992. IEEE. pp. 101-104.

Mersereau R.M.; Smith M.J. 1992. Introduction to Digital Signal Processing: A Computer Laboratory Textbook. John Wiley & Sons. 191 p.

Mersereau R.M.; Smith M.J. 1994. Digital Filtering: A Computer Laboratory Textbook. John Wiley & Sons. 223 p.

Orsak, G.C; Etter, D.M. 1995. Collaborative SP Education Using the Internet and Matlab. IEEE Signal Processing Magazine, 12, 6. pp. 23-32.

Pohjolainen, S.; Multisilta, J.; Antschev, K.; Väljas, M. 1994. Hypermedia Learning Environment for Mathematics. Tampere University of Technology, Mathematics, Hypermedialaboratory, Mathematics Software Report 8. 67 p.

Proakis, J.G.; Manolakis, D.G. 1992. Digital Signal Processing: Principles, Algorithms and Applications. 2 ed. Macmillan. 969 p.

Rahkila, M; Karjalainen, M. 1995. Computer Based Education with the QuickSig DSP environment. FINSIG'95, Espoo, Finland, June 2, 1995. Espoo, Finland. Helsinki University of Technology. pp. 123-127.

Rivlin. E.; Botafogo, R.; Shneiderman, B. 1994. Navigating in hyperspace: designing a structure based toolbox. Communications of the ACM. 37, 2, pp. 87-96.

Smith, J.O. 1989. Fundamentals of digital filter theory. In: Roads, C. (ed.). The Music Machine. MIT Press. pp. 509-519.

Steele G.L. 1990. Common Lisp - The Language. 2nd ed. Digital Press. 1029 p.

Wakefield, G.H.; Feng, B.J.; Raghavan, K.R.; Blommer, M.A. 1993. SP Education Laboratory. ICASSP’93, Minneapolis, Minnesota, USA, April 27-30, 1993. IEEE. pp. V-626 - V-629.

Wood, S. 1992. Tutorial Visualization Software for Concept Reinforcement in DSP Education. ICASSP’92, San Francisco, CA, USA, March 23-26, 1992. IEEE. pp. 77-80.

Välimäki V.; Karjalainen M.; Kuisma T. 1994. Articulatory Speech Synthesis Based on Fractional Delay Waveguide Filters. ICASSP-94, Adelaide, Australia, April 19-22, 1994. IEEE. pp. 585–588.

Välimäki V.; Karjalainen M.; Kuisma T. 1994. Articulatory Control of a Vocal Tract Model Based on Fractional Delay Waveguide Filters. ISSIPNN-94, Hong Kong, April 13-16, 1994. IEEE. pp. 571-574.

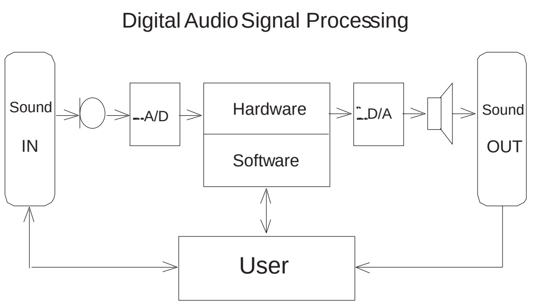
Author: Martti Rahkila
Subject: Introduction to Signal Processing

General study plan
Date: Summer 1994

Topics

* Introduction
* Signals and their properties
* Analog < > Digital
* Signal Processing:
- description
- spectrum analysis
- filters

First orientationground



Educational contents

Basic concepts in a practical way "familiarizing with signal processing in practice"

Target group

Students of acoustics and audio signal processing 2nd-3rd year

Practise

Interactive teaching with a computer

Level of knowledge or education

Present	Goal
Know how to use computers Familiar with mathematics (at least in principle) Knows the field only a little Cannot see the whole picture and does not know working methods	Clear picture of the field Knows the basic concepts Knows how sound sounds and what it looks like In future knows how to combine theory and practice

Working model or context

Present	Goal
There is a need: education emphasizes theory and is somewhat nonuniform Many different tools and applications are used in practice	Puts education and practice together Makes education more clear

Original design © Veli-Pekka Lifländer

DETAILED STUDY PLAN Course: Introduction to Signal Processing
Page: / Date: Autumn 1994

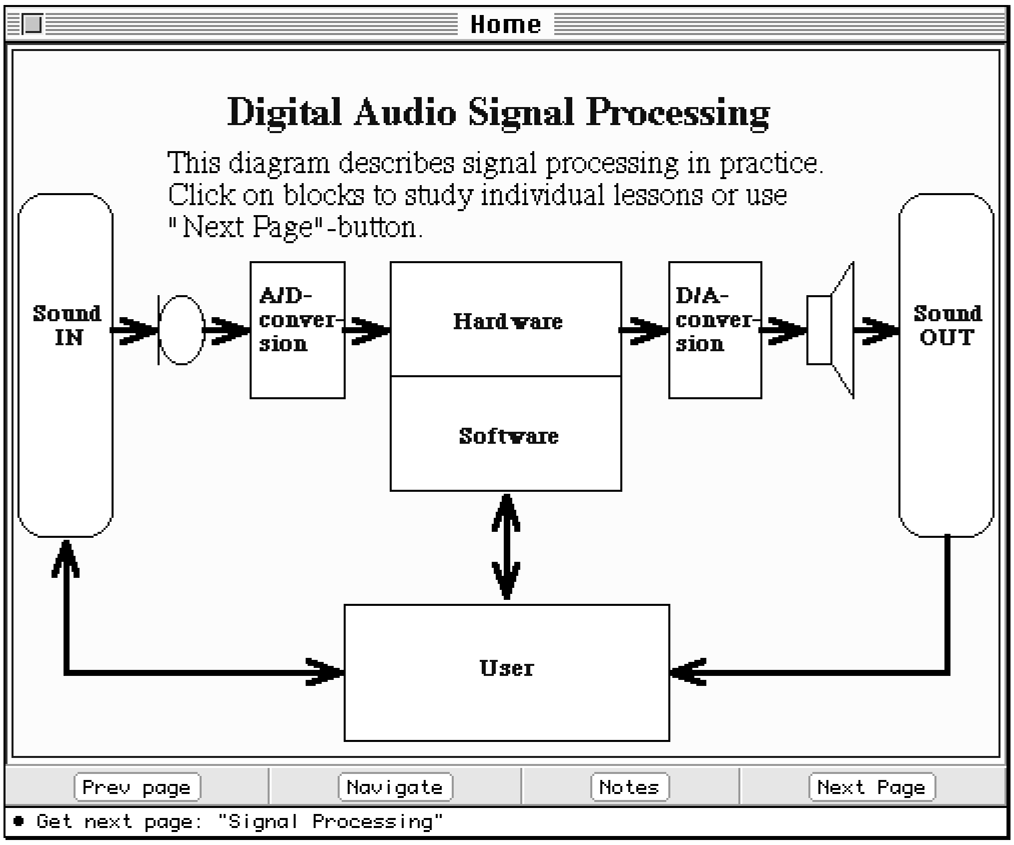
TOPIC AND THE DEVE-LOPMENT PHASE IN THE ORIENTATION GROUND	STUDYPERIODS KEY CONCEPTS AND SPECIAL ORIENTATION GROUNDS	STUDY PHASE (MOtivation ORientation UNderstanding APplying EValuation COntrol) AND ITS CON-TENTS	DURATION	STUDENT'S TASKS AND LEARNING PHASE (MO, OR, UN AP, EV, CO)	INTERACTION: (Group-student-tools-teacher) TEACHING METHOD AND DIVISION INTO GROUPS
General description	Digital Audio Signal Processing (figure) - Home Page	MO, OR		Get extra information by moving the cursor	
Signals	Verbal definition Graphic Presentation Numeric Presentation Mathematic Presentation Properties: amplitude, frequency, phase	OR, UN		See and listen example signals: Impulse Sine wave Noise Speech	
Home Page	Digital Audio Signal Processing (figure)	MO, OR, EV, CO		Repeat the orientation ground	
Spectrum Analysis	Verbal and graphic presentation Mathematical definition and calculation Spectrum properties Windowing	UN, AP		Calculate spectra of example signals and windows	
Home Page	Digital Audio Signal Processing (figure)	MO, OR, EV, CO		Repeat the orientation ground	
Analog/Digital- and Digital/ Analog-conversion	A/D-process: Sampling Quantization Coding Step by step	UN, AP, EV		Small questions and thinking: How many bytes takes 1 min of CD-audio? What is the difference between A/D & D/A-conversion? If possible: Record a new signal example	

DETAILED STUDY PLAN Course: Introduction to Signal Processing
Page: / Date: Autumn 1994

TOPIC AND THE DEVE-LOPMENT PHASE IN THE ORIENTATION GROUND	STUDYPERIODS KEY CONCEPTS AND SPECIAL ORIENTATION GROUNDS	STUDY PHASE (MOtivation ORientation UNderstanding APplying EValuation COntrol) AND ITS CON-TENTS	DURATION	STUDENT'S TASKS AND LEARNING PHASE (MO, OR, UN AP, EV, CO)	INTERACTION: (Group-student-tools-teacher) TEACHING METHOD AND DIVISION INTO GROUPS
Home Page	Digital Audio Signal Processing (figure)	MO, OR, EV, CO		Repeat the orien-tation ground	
Filters	Filter as a black box Definition of impulse response and transfer function Convolution Basic filter types	UN, AP, EV		Small questions and thinking: What convolution theorem actually is and is it useful? What is the key idea with the transfer function? Calculate spectra of filters. Perform filtering with example signals.	
Home Page	Digital Audio Signal Processing (figure)	MO, OR, EV, CO		Repeat the orien-tation ground	
Signal Processing	Concepts: Analog - Digital Linear - Nonlinear Time-variant - Time-invariant Real-time - Non-real-time DSP basic ope-rations	UN, EV, CO		Small questions and practical examples Get the big picture	
Home Page	Digital Audio Signal Processing (figure)	MO, OR, EV, CO		Repeat the orien-tation ground	
Grand Finale	All above	UN, AP, EV, CO		Design filters, calculate spectrums and perform filtering with example signals See and listen results	

Original design © Veli-Pekka Lifländer

CODE-EXAMPLE “Home Page”



This diagram describes signal processing in practice. Click on blocks to study individual lessons or use "Next Page"-button.

● Get next page: "Signal Processing"

Figure C-1: Final “Home-Page”

In QuickCBE it is easy to create pages like the one in figure C-1. The code is more like a layout script rather than standard object-oriented Lisp. The code example here demonstrates the syntax.

First of all, the defpage macro takes care of creating a new instance of the desired class (defined with :window-class slot). The :v slot arrenges elements vertically. The HotAreas and their properties (especially their locations, help texts and actions) are then defined on top of a pict-file. A set of button controls are placed below the pict. Finally, a few window properties are defined. Here is the code:

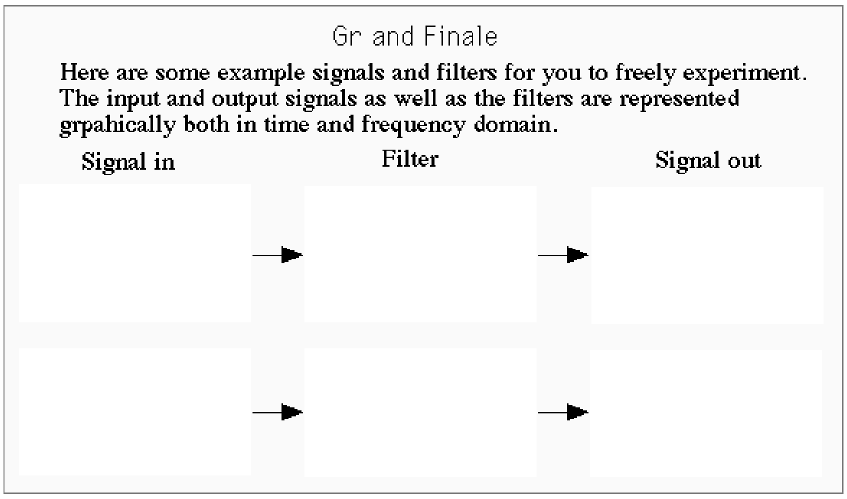
```
;;;HOME-page

(defpage home
  (:v (make-pict :pict-file "ccl:CBE;Pics;home.pict"
                 :hot-areas (list
                            (hot-word
                                 :location '(1507437 2949502)
                                 :word "¥ Introduction"
                                 :hot-handler #'(lambda ()
                                                  (switch-qs-page
'intro1)))
                            (hot-highlight-on-off-area
                                 :location '(4915207 16187447)
                                 :qs-help-string "¥ Signals"
                                 :hot-handler #'(lambda ()
                                     (switch-qs-page 'SignalPage1)))
                                 ;:oval-width 20
                                 ;:oval-height 20)
                            (hot-highlight-on-off-area
                                 :location '(8192077 10616932)
                                 :qs-help-string "Sound signals are
recorded with a microphone"))
                            (hot-highlight-on-off-area
                                 :location '(7209083 11731114)
                                 :qs-help-string "¥ Analog/Digital-
conversion"
                                 :hot-handler #'(lambda ()
                                     (switch-qs-page 'ADDA)))
                                 ;:hilite-p nil)
                            (hot-highlight-on-off-area
                                 :location '(7209153 14680372)
                                 :qs-help-string "¥ Signal
Processing"
                                 :hot-handler #'(lambda ()
                                     (switch-qs-page 'SP)))
                            (hot-highlight-on-off-area
                                 :location '(7209292 11731322)
                                 :qs-help-string "¥ Digital/Analog-
conversion"
                                 :hot-handler #'(lambda ()
                                     (switch-qs-page 'ADDA)))
                            (hot-highlight-on-off-area
                                 :location '(7471504 12059049)
                                 :qs-help-string "The result sounds
come out through a loudspeaker"))
                            (hot-highlight-on-off-area
                                 :location '(4850111 16187886)
                                 :qs-help-string "¥ Signals"
                                 :hot-handler #'(lambda ()
                                     (switch-qs-page 'SignalPage1)))
                            (hot-highlight-on-off-area
                                 :location '(18415786 23003468)
                                 :qs-help-string "¥ User Info and
Statistics"
                                 :hot-handler #'(lambda ()
                                     (switch-qs-page 'cbe-stats)))
                            )))

   (controls (prev-button 'intro3)
             (navigate-button)
             (notepad-button)
             (button :text "Next Page"
                     :qs-help-string
                         #'(lambda (x) x
                                 (format t "¥ Get next page: ~S"
                                    (get-page-title (find-qs-page
*next-session*))))
                     :button-handler
                         #'(lambda ()
                                 (switch-qs-page *next-session*)))
                        ))
   :title "Home"
   :view-size #@(500 396)
   :window-type :document
   :window-class 'cbe-qs-page-window
   :view-position #@(65 60))

;(show-qs-page 'home)
```

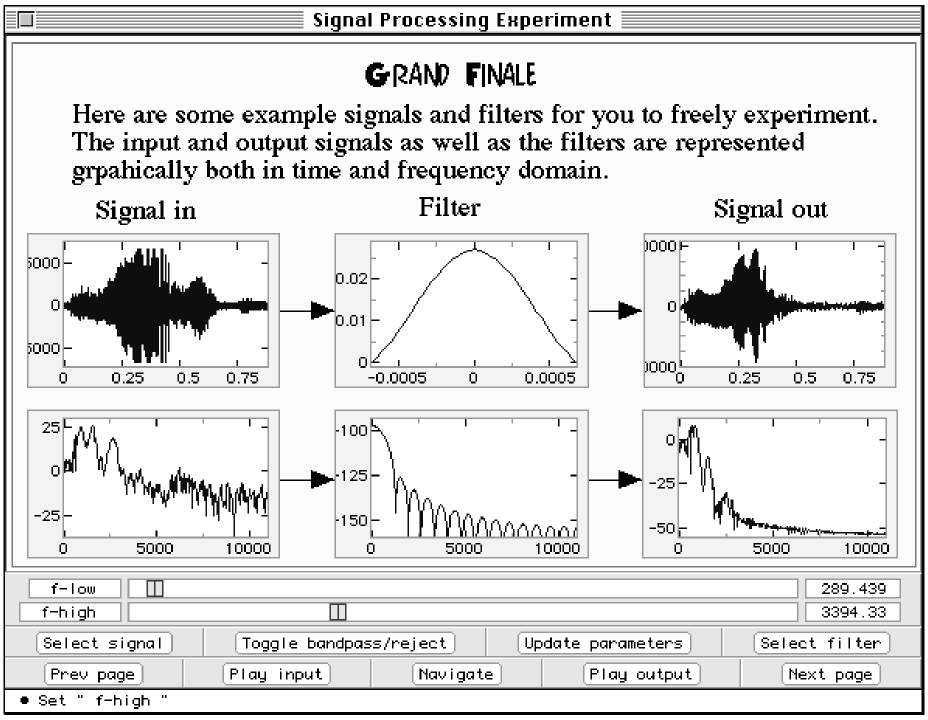
CODE-EXAMPLE “Grand Finale”



Gr and Finale

Here are some example signals and filters for you to freely experiment. The input and output signals as well as the filters are represented grpahically both in time and frequency domain.

Figure D-1: the graphical frame for the experiment (note! the original file contains vector information (fonts) that does not print correctly)



Signal Processing Experiment

GRAND FINALE

Here are some example signals and filters for you to freely experiment. The input and output signals as well as the filters are represented grpahically both in time and frequency domain.

f-low 289.439

f-high 3394.33

• Set " f-high "

Figure D-2: the final, interactive experiment layout

This code example demontrates a rather complex experiment. The code includes two dialogs for choosing example signals and filter-types, generating example signals (impulse, sine, white noise, speech), filter design using slider controls and the final page layout. Here is the code:

```
;;; filter-dialog

(defparameter *filter-examples-dialog* nil)
(defparameter *filter-examples-list*
  '(("Lowpass" "Filter" " ")
    ("Highpass" "Filter" " ")
    ("Bandpass" "Filter" " ")
    ("Bandstop" "Filter" " ")
    ))

(defun filter-final-dialog ()
  (let* ((seq (MAKE-DIALOG-ITEM
               'sequence-dialog-item #@(9 35) #@(183 65) "Example
filters" 'NIL
               :CELL-SIZE #@(267 16)
               :SELECTION-TYPE :SINGLE :TABLE-HSCROLLP NIL :TABLE-
VSCROLLP T
               :TABLE-SEQUENCE *filter-examples-list*
               :view-nick-name 'filter-examples-list
               :table-print-function
               #'(lambda (x s)
                   (princ (first x) s)
                   (princ " " s)
                   (princ (second x) s)
                   (princ " " s)
                   (princ (third x) s)))))
    (setq *filter-examples-dialog*
          (MAKE-INSTANCE 'DIALOG
            :WINDOW-TYPE
            :DOCUMENT
            :WINDOW-TITLE
            "Example filters"
            :VIEW-POSITION
            #@(20 250)
            :VIEW-SIZE
            #@(200 170)
            :CLOSE-BOX-P
            t
            :VIEW-FONT
            '("Chicago" 12 :SRCOR :PLAIN)
            :view-nick-name 'filter-dialog
            :VIEW-SUBVIEWS
            (LIST (MAKE-DIALOG-ITEM
                   'STATIC-TEXT-DIALOG-ITEM
                   #@(50 9)
                   #@(120 16)
                   "Choose a filter"
                   'NIL)
                  seq
                  (MAKE-DIALOG-ITEM
                   'BUTTON-DIALOG-ITEM
                   #@(70 140)
                   #@(62 16)
                   "Select"
                   #'(lambda (x) (eval-enqueue `(read-final-filter-
examples-dialog-and-select ,x)))
                   :DEFAULT-BUTTON T)
                  ))))
  *filter-examples-dialog*)

;(filter-final-dialog)

;;; example signals

(defparameter *in* *speech*)
(defparameter *filt-ex* *out*)
(defparameter *lp-filt* (make-bandpass-signal
                         ;:scaler (scaler self)
                         :f-low 0.0
                         :f-high 300.0
                         :order 31
                         :to 'float-signal))
(defparameter *hp-filt* (make-bandpass-signal
                         ;:scaler (scaler self)
                         :f-low 0.0
                         :f-high 2500.0
                         :band-reject t
                         :order 31
                         :to 'float-signal))
(defparameter *bp-filt* (make-bandpass-signal
                         ;:scaler (scaler self)
                         :f-low 500.0 :f-high 3000.0
                         :order 31
                         :to 'float-signal))
(defparameter *bs-filt* (make-bandpass-signal
                         ;:scaler (scaler self)
                         :f-low 1000.0 :f-high 3000.0 :band-reject t
                         :order 31
                         :to 'float-signal))

(defparameter *bp-filt-spect*
  (real-magnitude-spectrum (make-bandpass-signal
                         ;:scaler (scaler self)
                         :f-low 500.0 :f-high 3000.0
                         :order 31
                         :to 'float-signal)))

;;; filter dialog

(defun read-final-filter-examples-dialog-and-select (item)
  (let* ((cont (view-container item))
         (filt-ex-view (find-view cont 'filter-examples-list))
         (coord (first (selected-cells filt-ex-view)))
         (ind (when coord (cell-to-index filt-ex-view coord)))
         (filt-ex (when ind (elt (table-sequence filt-ex-view)
ind))))
  (cond ((equal (first filt-ex) "Lowpass") (setq *filt-ex* *lp-
filt*))
        ((equal (first filt-ex) "Highpass") (setq *filt-ex* *hp-
filt*))
```

```
        ((equal (first filt-ex) "Bandpass") (setq *filt-ex* *bp-
filt*))
        ((equal (first filt-ex) "Bandstop") (setq *filt-ex* *bs-
filt*)))
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'time-filt
(find-qs-page 'final-experiment))
   *filt-ex*)
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'spect-filt
(find-qs-page 'final-experiment))
   (windowed-fft-spectrum *filt-ex*))
  (setq *out* (noting-progress ("Filtering ...")
                (direct-convolve-simple *in* *filt-ex*)))
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'time-out
(find-qs-page 'final-experiment))
   *out*)
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'spect-out
(find-qs-page 'final-experiment))
   (windowed-fft-spectrum *out*))
  ))

(defparameter *signal-examples-dialog* nil)
(defparameter *signal-examples-list*
  '(("Impulse" " " " ")
    ("Sine" " " " ")
    ("White" "noise" " ")
    ("Speech" " " " ")
    ))

;(defparameter *in* *speech*)

(defun signal-examples-dialog ()
  (let* ((seq (MAKE-DIALOG-ITEM
               'sequence-dialog-item #@(9 35) #@(183 80) "Example
signals" 'NIL
               :CELL-SIZE #@(267 16)
               :SELECTION-TYPE :SINGLE :TABLE-HSCROLLP NIL :TABLE-
VSCROLLP T
               :TABLE-SEQUENCE *signal-examples-list*
               :view-nick-name 'signal-examples-list
               :table-print-function
               #'(lambda (x s) (princ (first x) s) (princ " " s)
(princ (second x) s) (princ " " s) (princ (third x) s)))))
    (setq *signal-examples-dialog*
          (MAKE-INSTANCE 'DIALOG
            :WINDOW-TYPE
            :DOCUMENT
            :WINDOW-TITLE
            "Example signals"
            :VIEW-POSITION
            #@(20 50)
            :VIEW-SIZE
            #@(200 170)
            :CLOSE-BOX-P
            t
            :VIEW-FONT
            '("Chicago" 12 :SRCOR :PLAIN)
            :VIEW-SUBVIEWS
            (LIST (MAKE-DIALOG-ITEM
                   'STATIC-TEXT-DIALOG-ITEM
                   #@(50 9)
                   #@(120 16)
                   "Choose a signal"
                   'NIL)
                  seq
                  (MAKE-DIALOG-ITEM
                   'BUTTON-DIALOG-ITEM
                   #@(70 140)
                   #@(62 16)
                   "Select"
                   #'(lambda (x) (eval-enqueue `(read-signal-
examples-dialog-and-select ,x)))
                   :DEFAULT-BUTTON T)
                  ))))
  *signal-examples-dialog*)

;(signal-examples-dialog)


(defun read-signal-examples-dialog-and-select (item)
  (let* ((cont (view-container item))
         (sig-ex-view (find-view cont 'signal-examples-list))
         (coord (first (selected-cells sig-ex-view)))
         (ind (when coord (cell-to-index sig-ex-view coord)))
         (sig-ex (when ind (elt (table-sequence sig-ex-view) ind))))
  (cond ((equal (first sig-ex) "Impulse") (setq *in* *imp*))
        ((equal (first sig-ex) "Sine") (setq *in* *sine-example-1*))
        ((equal (first sig-ex) "White") (setq *in* *randx*))
        ((equal (first sig-ex) "Speech") (setq *in* *speech*))
        )
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'time-in (find-
qs-page 'final-experiment))
   *in*)
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'spect-in
(find-qs-page 'final-experiment))
   (windowed-fft-spectrum *in*))
  (setf *out* (noting-progress ("Filtering ...")
                (direct-convolve-simple *in* *filt-ex*)))
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'time-out
(find-qs-page 'final-experiment))
   *out*)
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'spect-out
(find-qs-page 'final-experiment))
   (windowed-fft-spectrum *out*))
  ))

(defparameter *bandreject-toggle-p* nil)

;;; slider control

(defun read-sliders-and-filter ()
  (let* ((lowslider (find-named-item (find-qs-page 'final-
experiment) 'f-low-slider (find-qs-page 'final-experiment)))
```

```
         (highslider (find-named-item (find-qs-page 'final-
experiment) 'f-high-slider (find-qs-page 'final-experiment)))
         (lowf (value-slot lowslider))
         (highf (value-slot highslider)))
    (setf *filt-ex* (make-bandpass-signal
                     :order 31
                     :f-low lowf
                     :f-high highf
                     :band-reject *bandreject-toggle-p*))
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'time-filt
(find-qs-page 'final-experiment))
   *filt-ex*)
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'spect-filt
(find-qs-page 'final-experiment))
   (windowed-fft-spectrum *filt-ex*))
  (setq *out* (noting-progress ("Filtering ...")
               (direct-convolve-simple *in* *filt-ex*)))
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'time-out
(find-qs-page 'final-experiment))
   *out*)
  (replace-presentation-object
   (find-named-item (find-qs-page 'final-experiment) 'spect-out
(find-qs-page 'final-experiment))
   (windowed-fft-spectrum *out*))
  ))

(defparameter *nil-sig* (make-instance 'float-signal :beg -100 :end
10000))

;;; final page

(defpage final-experiment
          (:v (:o (make-pict :pict-file "ccl:CBE;Pics;CBE-final-
experiment.pict")
                  (:k *speech*
                      :name 'time-in
                      ;:x-interval (make-span 0 1)
                      ;:y-interval (make-interval -30 30)
                      :left-margin 23 :right-margin 8
                      :bottom-color *12.5%-gray-color*
                      :view-position #@(15 129)
                      :adjust-p nil
                      :view-size #@(162 99))
                  (:k *bp-filt*
                      :name 'time-filt
                      ;:x-interval (make-span 0 1)
                      ;:y-interval (make-interval -30 30)
                      :left-margin 23 :right-margin 8
                      :bottom-color *12.5%-gray-color*
                      :view-position #@(213 129)
                      :adjust-p nil
                      :view-size #@(163 99))
                  (:k *nil-sig*
                      :name 'time-out
                      ;:x-interval (make-span 0 1)
                      ;:y-interval (make-interval -30 30)
                      :left-margin 23 :right-margin 8
                      :bottom-color *12.5%-gray-color*
                      :view-position #@(412 129)
                      :adjust-p nil
                      :view-size #@(162 99))
                  (:k *speech-mag-spect*
                      :name 'spect-in
                      ;:x-interval (make-span 0 1)
                      ;:y-interval (make-interval -30 30)
                      :left-margin 23 :right-margin 8
                      :bottom-color *12.5%-gray-color*
                      :view-position #@(15 243)
                      :adjust-p nil
                      :view-size #@(162 95))
                  (:k *bp-filt-spect*
                      :name 'spect-filt
                      ;:x-interval (make-span 0 1)
                      ;:y-interval (make-interval -30 30)
                      :left-margin 23 :right-margin 8
                      :bottom-color *12.5%-gray-color*
                      :view-position #@(213 243)
                      :adjust-p nil
                      :view-size #@(163 95))
                  (:k *nil-sig*
                      :name 'spect-out
                      ;:x-interval (make-span 0 1)
                      ;:y-interval (make-interval -30 30)
                      :left-margin 23 :right-margin 8
                      :bottom-color *12.5%-gray-color*
                      :view-position #@(411 243)
                      :adjust-p nil
                      :view-size #@(164 95)))
                  (sliders (list (slider :title " f-low "
                                         :name 'f-low-slider
                                     :init-value 300.0
                                     :mapping-function
                                     #'(lambda (x y) (* 11025 (/
(float x) y)))
                                     :slider-action
                                     #'(lambda (x) x nil))
                             (slider :title " f-high "
                                     :name 'f-high-slider
                                     :init-value 3400.0
                                     :mapping-function
                                     #'(lambda (x y) (* 11025 (/
(float x) y)))
                                     :slider-action
                                     #'(lambda (x) x nil))))
                  (controls (button :text "Select signal"
                                    :button-handler
                                    #'(lambda () (let (dlog)
                                                   (setq dlog (find-
window "Example signals"))
                                                   (if dlog (window-
select dlog)
                                                       (signal-
examples-dialog)))))
                            (button :text "Toggle bandpass/reject"
                                    :qs-help-string
                                    #'(lambda (x) x (if *bandreject-
toggle-p*
```

```
                                                       (format t
"BAND-REJECT is now ON. Sliders create band-reject filters")
                                                       (format t
"BANDPASS option is now ON. Sliders create bandpass filters")))
                                    :button-handler
                                    #'(lambda () (if *bandreject-
toggle-p*
                                                      (setq
*bandreject-toggle-p* nil)
                                                      (setq
*bandreject-toggle-p* t)
                                                      )))
                            (button :text "Update parameters"
                                    :qs-help-string "¥ Update filter
parameters with slider values"
                                    :button-handler
                                    #'(lambda () (eval-enqueue
'(read-sliders-and-filter))))
                            (button :text "Select filter"
                                    ;:qs-help-string "¥ Listen to
filtered signal"
                                    :button-handler
                                    #'(lambda () (let (dlog)
                                                    (setq dlog (find-
window "Example filters"))
                                                    (if dlog (window-
select dlog)
                                                        (filter-
final-dialog))))))
              (controls (prev-button 'home)
                        ;(button :text "Play input"
                        ;        :qs-help-string "¥ Listen to input
signal"
                        ;        :button-handler
                        ;        #'(lambda () (play-signal *in*)))
                        (play-button :receiver 'time-in :text "Play
input")
                        (navigate-button)
                        ;(button :text "Play output"
                        ;        :qs-help-string "¥ Listen to
filtered signal"
                        ;        :button-handler
                        ;        #'(lambda () (play-signal *out*)))
                        (play-button :receiver 'time-out :text "Play
output")
                        (next-button 'theend)))
          :title "Signal Processing Experiment"
          :view-size #@(590 437)
          :window-type :document
          :window-class 'cbe-qs-page-window
          :view-position #@(20 38))

;(show-qs-page 'final-experiment)
```

SUMMARY AND LAYOUT OF THE USER QUERY

HUT Acoustics Laboratory — Autumn 1995
"Introduction to Signal Processing" -CBE application
Martti Rahkila — User Feedback Query **summary**

The query was answered by: `78 people and then we run out of forms... Total amount of users: 87`
Student number: Year (grade): `average 3.37`

Contents:

In your opinion, did You learn something? If, then what was it?

```
- students learned equally from every area
- the program was also considered as good repetition
- “an old Mac FX is a slow machine for signal processing”
```

Were there any familiar concepts:

```
- some things were previously learned, but equally from
  every area. Occasionally there were a lot of things
  already known.
```

If there were, estimate the familiar contents as a procentage of all contents?
`74.34 % average`

Did You find the program to have too little / just the right amount of / too much educational contents. `-----------------------`

Would You have used the program even if You didn’t get extra credits?
`84,93 % yes`

Should the contents of the program to be available also in paper?
If so, preferrably beforehand / afterwoods?
`- this was hoped for...`

The application:

Were there any unexpected situations while using the program (errors, program crashed, I didn't know how to proceed etc.)? What happened?
- the login phase failed (mistypes)
- the program crashed during the last pages

Was the tempo of the program too slow / perfect / too fast?

```
                        -------------
```

Were the arrangements (1 machine, time reservations) good? yes

Would it be better if there was a "home-version" / network version?
- a windows-version could be implemented with a huge amount of work...
- a network version would be the best alternative, but also requires a lot of work

Should the program be used more than once in order to learn everything?
- yes and no

RELATED INTERNET RESOURCES

http://www.hut.fi/HUT/Acoustics/	Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing

Computer Based Education

http://www.hgur.se/CAL/	CBE (Computer Based Education) index
http://www.uct.ac.za/projects/cbe/	Computer Based Education collection
http://www.strath.ac.uk/CAL/	CAL pages
http://viswiz.gmd.de/MultimediaInfo/	Index to Multimedia Information Sources
http://matwww.ee.tut.fi/hmlab/ homepage.html	Hypermedialaboratory at Tampere University of Technology

Signal Processing

http://www-dsp.rice.edu/spib.html	Signal Processing Information Base
http://www-dsp.rice.edu/splib/	Signal Processing URL Library
http://www.spec.gmu.edu/	Signal Processing Education Consortium
http://www.dspnet.com/	DSPnet On-line DSP info
http://www.yahoo.com/Science/ Engineering/Electrical_Engineering/ Signal_and_Image_Processing/	Yahoo Signal Processing Index

Tools

http://www.mathworks.com/	Matlab
ftp://ftp.mathworks.com/pub/contrib/ signal/spctools	SPCTools package for Matlab
http://www.wolfram.com/	Mathematica
http://ptolemy.berkeley.edu/	Ptolemy
http://www.altagroup.com/	SPW
http://www.digitool.com/	MacLisp
http://www.apple.com/	Apple Macintosh
http://www.macromedia.com/	Authorware, Director
http://www.asymetrix.com/	ToolBook
http://www.allegiant.com/	SuperCard
http://www.glasscat.com/hypercard.html	HyperCard
http://www.metacard.com/	MetaCard