# The Design and Evaluation of Computer Music Interfaces.

Michelle Mary Mc Cormack

Submission for: Doctor of Philosophy

City University
Music Department
September 1996

This research thesis is dedicated to my parents,

*Kevin & Eilish Mc Cormack*

with Love and Laughter

and a never ending Thank You.

Contents

List of Figures

Chapter 8 SYTER
Real-time Composition and Sampling Environment



Chapter 9
Steim Composition and Performance Systems

**Chapter 10**
**POD-X Polyphonic Computer Composition System**



**Chapter 11**
**Specialist Interfacing Considerations: KE:NX**

ACKNOWLEDGEMENTS

I would like to acknowledge the assistance and support given in the realisation of this research and to express my sincere thanks to my supervisor, Dr. Simon Emmerson, of City University, London.

I would like to express thanks to:

CDP Research:

Dr. Richard Orton of York University, Dr. Trevor Wishart, Dr. Simon Emmerson of City University.

UPIC Research:

Alain Despres, Didier Rocton and the staff of the UPIC research group and the CEMAMu; Dr. Peter Nelson of Edinburgh University;

SYTER Research:

Daniel Teruggi, Bernard Parmegiani and the staff of the INA-GRM, Paris.

STEIM Research:

Michel Waiswisz, Joel Ryan, Nick Collins, Frank Balde, Tom Demeyer and the staff of STEIM studio, Amsterdam.

POD-X Research:

Prof. Barry Truax, Dept of Communications, Simon Fraser University, Hildegard Westerkamp, Simon Fraser University;

Specialist Interface Research:

Stephen Olwill of the Drake Music Project Ireland, Therese Burns of the Drake Music Project Ireland and DMPI volunteers and tutors.

All photographic material contained in this thesis remains the property of the author, and thanks are due to all institutions that allowed workstations and equipment to be photographed by the author for the purposes of this research.

Many thanks to Eilish, Maureen and Therese for proof reading and patience, and the myriad of composer friends, especially Francois Evans, who allowed themselves to be questioned and evaluated during the course of this research.

A very special Thank You to Stephen Dunniece, for advice, understanding and very many hours of patience.

## DECLARATION OF DISCRETION

I hereby grant powers of discretion to the University Librarian of the City University London to allow this thesis to be copied in whole or in part without further reference to the author. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

## ABSTRACT

The Design and Evaluation of Computer Music Interfaces.

The purpose of this research is to examine the methods employed to design a computer music system suitable for music composition purposes. The nature of an interface is introduced and discussed (Chapter 2), with particular reference to interfacing issues for computer music users. The nature of the user and the notion of a task in computer music are discussed and examined (Chapter 3), with outline methods for evaluations of users and tasks being introduced. Current computer systems design and analysis methodologies are introduced (Chapter 4), and subsequently adapted for use in computer music design and analysis applications. A set of guidelines is introduced in Chapter 5, specifically related to the design and analysis of computer music systems. These guidelines are formulated from the discussions regarding the interfacing requirements of a musician user and the nature of the tasks he wishes to accomplish employing computer-related technology . An examination of several systems forms a major part of the thesis (Chapters 6 – 10), outlining a variety of available composition tools at the time of writing. Each system introduced and discussed is evaluated in relation to the guidelines introduced for design and analysis of computer music systems. An interface design case study (Chapter 11) employs the design methods introduced during the thesis, and highlights the need for intensive designer – user discourse.

# Chapter 1: Introduction.

Music composition, when integrated with technology, ideally aims to optimise the combined effect of three principle factors:

- The musical discourse
- Machines and their corresponding virtual interfaces
- Hardware interfaces

Any development in this field must apply itself to all three factors in order to provide for a successful compositional working environment. Any software design must relate itself to the end result of music creation, edition, scoring or other general musical task, and should in every way assist the process. By the same measure, creative ideas and musical tasks should be used as the stimulus in the development of both the hardware and the software components in computer music systems.

The author is primarily concerned with the task of music composition in this research[1]. The objective of this thesis is to address, discuss and define composer needs with regard to the types of interfaces which he has, or would like to have, access to.

[1]Similiar tasks may apply to the work of arrangers, musicians who do not compose, or anyone concerned with completing a musically related task with the aid of a computer system, but these will not form the focus of this thesis.

The dialogue between composer and machine, defined in terms of a transduction of energy, as seen in Figure 1.1, must be rendered coherent in order to provide a successful working environment. Energy is transduced in the form of information between the musician and the machine.

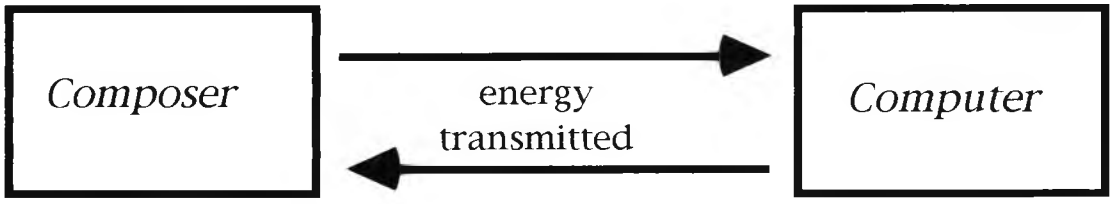



Figure 1.1 Transduction of Energy between Composer and Computer.

The transduction of energy between musician and machine takes place as a combined result of two types of interfacing by the user. Interfacing takes place physically with the controllers and effector surfaces of the physical machine[2], and also virtually, with the software image created on the computer screen.

The interest of this paper lies with the nature of the transducers which composers have access to, or ideally would like to have access to. Issues such as the nature of the user, the task to be performed, the type of virtual interfaces employed, and the physical domain of the system are introduced and discussed. The essence of the problem – a successful transduction of energy from musician to machine and back via the virtual interface – is dealt

[2] Interfacing here includes the impact of light and sound on the user.

with in depth with the intention of presenting open-ended non-system specific analysis and design criteria for future computer music systems. Current design and analysis methodologies are introduced and subsequently adapted for application to music systems.

The main tendency since about the late 1970s has been to rely almost exclusively upon the alpha-numeric keyboard, usually in conjunction with a mouse, as the sole form of physical communication between composer and systems software. However, several newer types of physical interfaces currently exist for composition systems. Later in this thesis, several different physical interfaces and systems which use varied interfacing and tasking approaches will be introduced and evaluated in an overview of methods of physical energy transduction from composer to system and back.

In a technologically advanced culture, if the musician is to improve his working environment, then improved communication with systems designers must be developed. Interaction between users and designers will mean mutual influence and can only result in improved working systems. In the chapters that follow, the nature of the musician user is discussed and classified. The user, however, is a difficult entity for the designer to understand. As Pennycook (1985: pg. 268) writes:

> "Musicians develop musical skills through formal study and practical application. Musical knowledge is translated into sound through the physical gestures of vocal and instrumental performance or, in the case of electro-acoustic

> music, through signal generation, amplifiers, and finally loudspeakers. Much of the musician's craft is absorbed unconsciously as part of the music-making experience. Unravelling these complex interrelationships of knowledge, experience and gesture poses a formidable challenge. Codifying the web of musical attributes loosely referred to as musicianship is further compounded by the fact that each and every musical style is a product of unique socio-temporal forces."

Several streams of knowledge, conditioning, education, sub-conscious traits and social forces combine in undefinable ways to make up the single person who is actively involved in music-making, as a composer, performer, arranger etc.. Joint studies by armies of psychologists, sociologists, cognitive scientists and fellow musician analysts have, to date, failed to define adequately the inner workings of the composer. To a large extent this failure has been attributed to the commonly held belief that so much of the composer's craft is absorbed subconsciously in both ummeasurable and unqualifiable doses. In striving to understand the nature of the user and his task, however, we should be cautioned by the danger of over-definition. As Richard Karpen (1995, pg. 16) writes:

> "Artists are always being asked to explain their working methods and their relationships to their media, as if through that explanation the work itself will become clearer. Sometimes that can be true. But sometimes, and for a variety of reasons, it is not true. It is sometimes true, too, that those with neat explanations about what they do might actually be creating work with finite and too well understood ideas and techniques. In other words, neat explanations do not equal good art!"

It is important to realise that by trying to understand the musician user and the nature of the tasks he uses computer environments to complete, we are not attempting to define the musician in a robotic or inflexible way. We are attempting to define classes and types of musician users as universal and open-ended as is feasible. There should be no attempt made to clone the musician who chooses to employ computer technology. The artistic temperament and tasks to be performed dictate as wide a definition of types of users and nature of tasks as possible within the realms of useful classifications. These classifications can then usefully serve to improve the line of communication between systems designers and end users, for mutual benefit.

With the constant march of technological advance, music composition has widened from purely acoustic work to delve into the electronic realms of sound creation and manipulation. Ever increasing numbers of composers are utilising the highly advanced electronic tools available for sound synthesis, control and performance. The growth of the computer music industry in the 1980s has resulted in a change in systems user types from computer literate experts of the MUSIC-V era of the 1960s, to more novice non-specialist systems users. This change may be seen in terms of the users of the early MUSIC-V type systems who were all computer-literate operating, through necessity, on the command line at levels very close to the micro-processor. Today many musician users of computer systems, e.g. sequencer users, operate at a very high level with a user-friendly graphics front end 'protecting' them from the programming heart of the machine.

With this market of musician users interested in exploring and exploiting the impact of the microchip in all its manifestations upon sound, a whole spectrum of new and innovative work environments has developed with a huge battery of tasks available for completion within the computer domain. To date, musicians can employ electronic architecture in some form to assist in almost any musical task they wish to perform. Rodet (1991, pg. 51) lists the following eleven tasks for which he believes we currently employ computer music tools:

- listening
- modelling
- learning
- recognising
- transcribing
- playing
- problem solving
- interacting with an instrument
- interacting with a player's model
- interacting with a knowledge base
- composing

This last category, composing, can be sub-divided into:

- sound creation
- sound storage
- sound manipulation, in time and/or in space
- timbre alteration
- timbre specification
- instrumental imitation

This formidable list of tasks which may be required by the contemporary user means that the design of a user-friendly system will be a complex undertaking. It is an axiom of this thesis that, in conjunction with actually providing access to any of these tasks within a system, the tools for accomplishing the tasks must be

accessible to musicians from varied backgrounds, either accomplished computer systems users or novice systems users. A system providing a set of tools to achieve tasks should be approachable for the novice while remaining useable and useful to the experienced systems user.

The problem is compounded by the fact that little expendable time is generally available for a musician actually to learn the operation of new tools within a system prior to employment of the system for a task. Working, as most composers do, to a commission time scale or bound to a performance date, little time is usually available to be spent on becoming an expert systems user, as opposed to a composer systems user who is able to get the results required from the system in the shortest possible time, by the shortest feasible route, with as little revision required as possible.

An assessement of the multi-tasking requirements of the computer-based music tool is the core of this research – both the discussion and analysis of such systems as they exist and have been used by the author, and also discussions and design methodologies for such systems as they might come to exist. The primary issue is the need for musician-friendly "front-ends" to be attached to the array of high-level electronic architecture available within the musician's workplace.

Issues of evaluating, designing and implementing user-friendly interfaces are not isolated within the music world. The design of a seemingly simple device such as the electric light

switch has been intensively researched, and something as common-place today as the control panel for an electric cooker is subject to constant analysis, empirical testing and upgrading. There are several definable guidelines for human interaction with any physical interface and it is the successful application of these guidelines in combination with system-specific components which will eventually constitute a user-friendly working environment. Some of these guidelines applied to computer music systems by the author will be contained within this research and where necessary will be refined specific to music systems design.

Within the realm of computer-aided composition, in its widest sense, the ability of the technology to achieve the previously unobtainable has in many cases far outpaced the creation of suitable control methods. As a consequence, the technological advances have either been semi-exploited or even, in some cases, poorly employed. As Max Mathews (in Goebel, 1991: pg. 49) writes:

> "I wonder if we're not overwhelmed by the variety of machines that exist, but rather by the rate at which the world is changing, by the fact that new machines come along before we can learn to live with, and create on the old machines."

Goebel in his paper *My Dream (Machine?)* (1991: pg. 49) goes even further:

> "...we are forcing ourselves to change with whatever

speed is being given to us by product development."

The involvement of musicians using technology in the 1990s is generally not in order to learn the intricate nature of the micro-processor commands involved, or to become high-level computer operators, programmers or designers, but to achieve their varied musical objectives as quickly and as clearly as possible in an environment of optimum control. The fact that the musician is a difficult user to define does not mean that musicians should assume the role of complacent end-user of a system designed without consultation. If possible the musician must actively participate in the design and encourage communication with the systems designer to develop a more open working environment with flexible tasking boundaries.

There is an inherent culture gap between a systems designer who is professionally enmeshed in macro and micro-level specifications for programming algorithms, and the end-user of a system who is necessarily driven by task requirements and practicality within a creational, musical realm. Designers are not generally like the eventual long-term users of a system. Due to the very advanced systems knowledge which they have acquired, any assumptions which they make about the intended user will not be precise without extensive user analysis. One of the most difficult issues in any design is the age-old problem of the specialist designer attempting to assume the role of the novice user of his own system, to code simplicity in the working of the completed interface. In conjunction with this initial knowledge gulf, the designer can rarely be expected to have a valid understanding of

the task applications for the system. It is essential therefore, for high and low-level analysis of the user to be conducted, so that the designer can be provided with as much knowledge as possible prior to taking any user-related decisions at the design stage which will ultimately have a direct effect on the musician's working methods. Ergonomic and psychological profiles should be constructed and applied in model testing before adoption as hard guidelines for the system design process.

It is obvious that the job of the interface designer or analyst, when dealing with musical applications of modern technology, is not an easy one. With an end-user who is, by his artistic nature, difficult to classify, and a task which is difficult to define due to its creative nature and necessary open-endedness, the designer of a computer music system is initially faced with a daunting task. By the same measure the interfaces currently available are not easily empirically analysed, as the task with computer music work for different users is rarely similar. As a consequence, degrees of success and failure of a music interface vary from one user to the next. It is necessary to attempt to generalise about the nature of both the user and the task in order to begin to visualise any form of suitable interface for a computer music station. As Buxton (1985: pg. 267), tells us:

> "When we have developed a methodology which allows us to determine the gesture which best suits the expression of a particular concept, then we will be able to build the user interfaces which today are only a dream."

Some thought about the nature of the gestures involved in composition, defined as tasks, follow later in this thesis, together with discussions about psychological characteristics of users which must be considered during interface design for any form of human-computer interaction to be successful.

The overall objective of the work at hand is to attempt to establish some basic criteria for music interface design. The notion of the interface is introduced and classified with consideration being given to the ergonomic, social and psychological elements of the user and the task, which all play some part in the design process of any system developed to be accessed, controlled and manipulated successfully by a human. Current general design methodologies are introduced and later applied in whole or part to music systems design work.

As the core of this thesis, a set of analysis and design guidelines and systems criteria are introduced and discussed. These are grouped in terms of general interfacing issues, guidelines relating directly to the musician user of a computer-based system, and finally criteria relating to the varied tasks which computer music systems are realised to achieve and assist with.

The guidelines and systems criteria are supported by the design of a systems synopsis grid, a method of system categorisation and analysis developed by the author, to facilitate and improve systems evaluations.

Having presented the core set of guidelines and systems criteria, the thesis proceeds to introduce a contrasting group of computer music systems as employed by the author during the progress of this research. These are subsequently classified and analysed in terms of the guidelines already introduced with respect to the nature of the interfacing employed, the intended system users, and the types of tasks to be achieved. The systems are also categorised using the system synopsis grid.

As a test case for design application of the guidelines at the core of this research and also the design methodologies which are introduced in Chapter 4, a specialist interface and physical workstation design undertaken by the author is outlined and discussed in detail. Some specialist considerations due to the nature of the users' physical ability result in an intensive application of these guidelines, matching user ability, task structures and interface access issues.

It is important to note at this stage that the following chapters include evaluations, insights and declarations by a composer user of computer music workstations, and not by a computer programmer or a systems analyst. All ergonomic and cognitive guidelines adopted are those deemed suitable by a systems user, not a systems designer. The original impetus for the discussions and evaluations which follow was constant frustration with the quality of the interface presented in systems used by the author. The ideal aim of the ensuing work could be defined as a desire to improve the working environment of composers who wish to avail themselves of modern computer and electronic technology,

without actually having to become high level computer users[3] of some description in order to do so. The overall aim is to improve the interaction between musician user and systems designer, in the belief that this will result in mutual influence which can only be beneficial.

This research contains a set of general open-ended analysis and design guidelines designed for and tested against a variety of computer music systems. The hope is that these guidelines will influence in whole or part – or at least be considered by – future systems design experts who are best placed to implement them. Due to the fact that any ergonomic studies will, as a result of the rate of technological advance, be lagging behind the latest processor designs, it is intended that these guidelines may apply in general terms to music systems design and not be related in any limiting fashion to contemporary hardware availability.

[3] A high level user is here understood to be a user who can handle command line syntax and programming languages, together with an understanding of compilers and syntax to control and run them.

# Chapter 2 Introduction to the Interface

An 'interface' is defined in the Oxford English Dictionary (1990: pg. 618) as:

> "a point where interaction occurs between two systems, processes, subjects, etc.."

The interface is the first aspect of any computer system, regardless of the nature of application, which a user will encounter. If the characteristics of the user and the task have been fully defined at the design stages, the interface is the only aspect of the system with which the user will ever need to have direct contact. The human computer interface (HCI), is that part of the system which manages communication between the user and the computer.

As we have seen in Figure 1.1, the transduction of energy between the user and the system is the core issue in interfacing between humans and machines. This energy is transduced across both the physical interfaces (or effector surfaces) and the virtual, or software-designed, surfaces of the machine. For this energy transduction to flow effortlessly, the software-created virtual interface between the user and the computer must be completely free of hindrance and made as intelligible to the user as possible. The interface acts as the translator in the process of a user action seeking a reaction from the computer system, as in Figure 2.1:

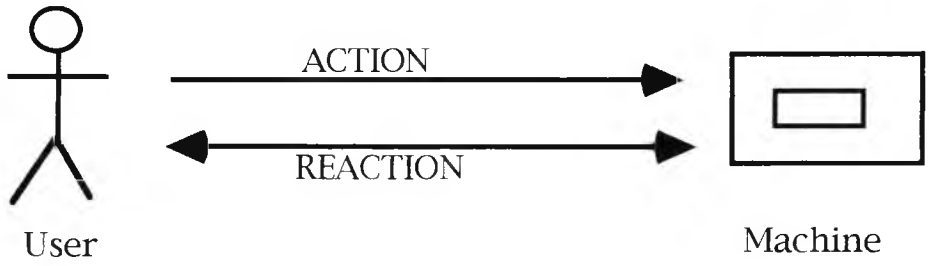



Figure 2.1 Action – Reaction between User and Machine

The research at hand is concerned with both the physical and the virtual interfacing issues in a computer music system. The following components of a typical user interface will be addressed throughout this thesis:

- Content

The content of the user interface concerns issues such as the type of information being exchanged between system and user, and the nature of the task being undertaken.

- Medium

The medium of the user interface concerns the type of hardware and software being employed in the system as a whole, the physical media of the system and the conceptual media of the system; it concerns also exactly what concepts the system has been realised to incorporate, and how successfully.

- Communication

Communication issues concern the dialogue structure between the user and the machine, via the physical and virtual interfaces. The format of the information layout and the user data input and

retrieval is of major concern for the success rating of the communication level of a system.

These interface components should be investigated at the information gathering stage of systems design and should permeate all stages of a designer's work for a successful interface environment to be created.

Acting as the opaque layer between the user and the systems computational activities, the virtual or software interface is not concerned with systems functions or underlying algorithmic programming. One problem with some current music systems virtual interfaces is that they are too closely linked with the underlying system processes, the software-created screen being almost transparent in its design. A user-friendly interface will be as free as possible from presenting the computations which the system is carrying out in reaction to a user's action. It is the virtual soft screen which protects the user from the computer processing environment and makes the system 'friendly'. A system interface does not need to be dictated in design and presentation by the nature of the processes underlying it. The interface is concerned primarily with interaction, not system functions.

Physical and virtual interfacing can be accomplished in a variety of ways, each of which must be evaluated and tested fully with the character and requirements of the user the most important criteria. Interfacing development should not be in step only with the newest available technology but with the

appropriateness of its functioning as a tool for users in response to social and psychological needs within their working environment.

Some physical interfacing methods currently in widespread use for composition systems are:

- Keyboard Interfaces

An alpha-numeric keyboard, acting as a micro-processor controller, on the front end of, for example, a Midi-event-based system, or a digital signal processing-based system. The keyboard would typically be part of a software-based system, a graphics screen environment, and/or a command line environment.

Figure 2.2 features a typical software screen which would be controlled via an alpha-numeric keyboard. The software imaged interface displays sequencing track information to the user, and provides access to several lower levels of MIDI data control. Each of the boxes in the left hand columns on this screen are accessible via the mouse and can have their internal values input from the keyboard when the box is highlighted for input. The dialogue boxes, for naming tracks or instruments are input to directly from the alpha-numeric keyboard when they have been selected for input by the mouse. Most of the on-screen controls can be input to and selected from the keyboard, by way of keyboard shortcuts; combination of the pressing of the command key together with for example, the letter R will result in the selected track or tracks being displayed in score format, or command-T will create a new track in the track listing.

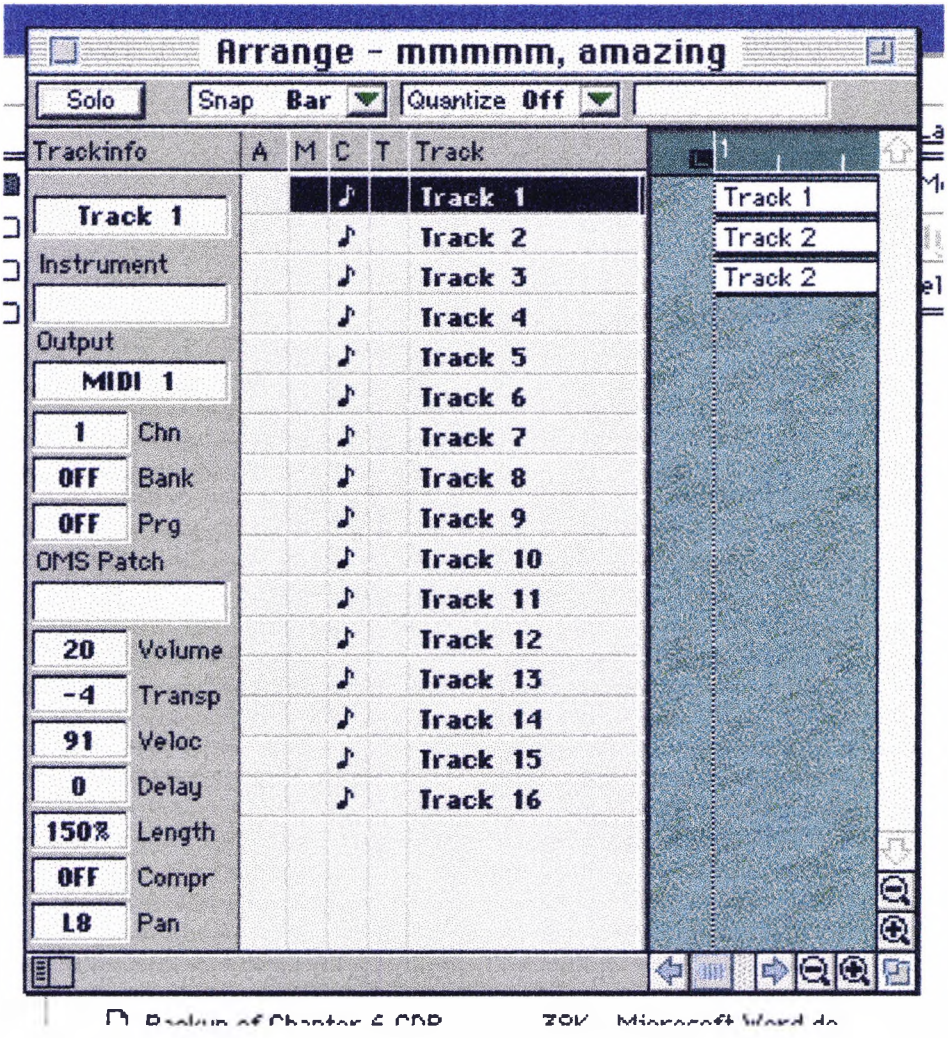



Figure 2.2 Midi-event-based Composition System

- Physical Gestural Controller Interfaces

Acting as energy transducers between machine and musician and back, this type of interface combines two separate approaches:

– Midi controllers, based on traditional instruments, e.g.: keyboard, clarinet, saxophone, violin;

– Other physical gestural controllers built into integrated performance systems which are processor-controlled and

incorporate a software interface for user control of the environment, e.g.:

MIDI Conductor, Steim Hands, Radio Drum[1], etc.

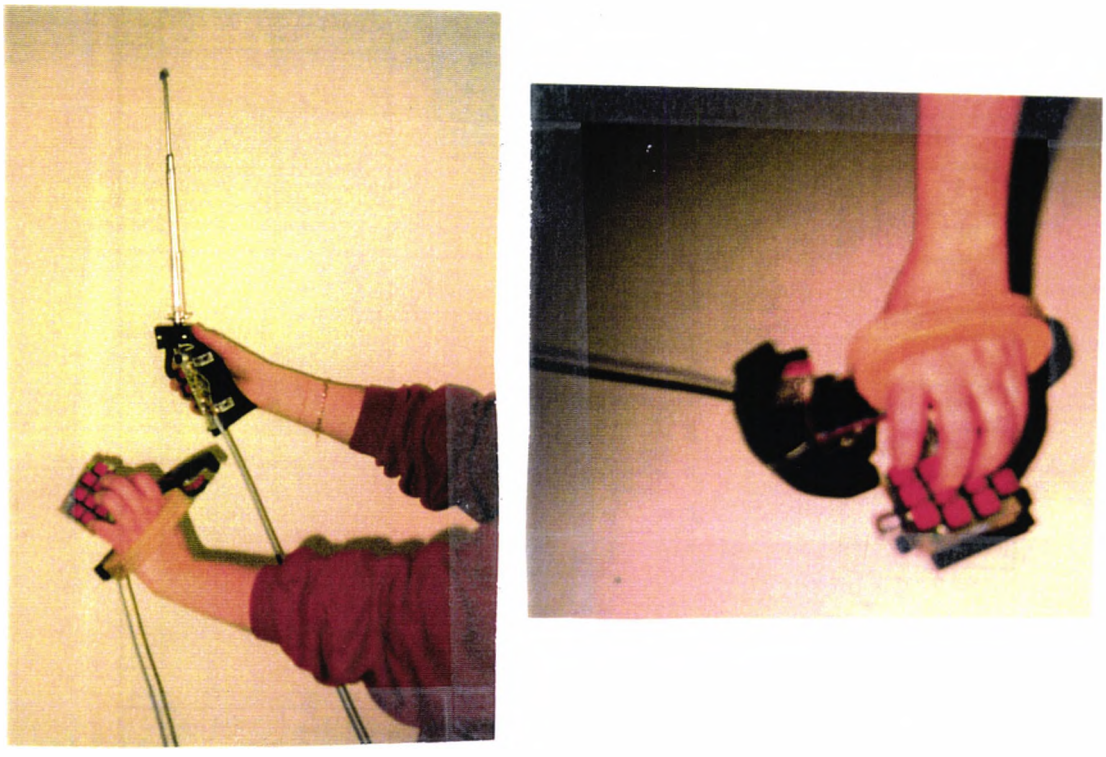


Figure 2.3 Midi Conductor Physical Interface.

Figure 2.3 above depicts the Midi Conductor physical controllers built at Steim Studios in Amsterdam, to interface to the Lick Machine software. These controllers are intended to operate as performance interfaces for composers and performers, and control pre-composed 'licks', or sets of MIDI data created in the Lick Machine software environment. Further details of this interface are to be found in Chapter 9 of this thesis.

[1] Instruments designed at STEIM Studios, Amsterdam (Krefeld, 1990).

- Graphics-based Controllers

Affording input capabilities away from the alpha-numeric keyboard, these controllers take full advantage of user interface research carried out in wider general user applications such as bank machine designs and technical medical equipment.

– UPIC graphical compositional system, digitiser tablets, light pens, touch screen, mouse

Figure 2.4 below shows the graphics drawing-board environment of the UPIC composition system, designed by composer Iannis Xenakis. Full details of this interface are in Chapter 7 of this research.

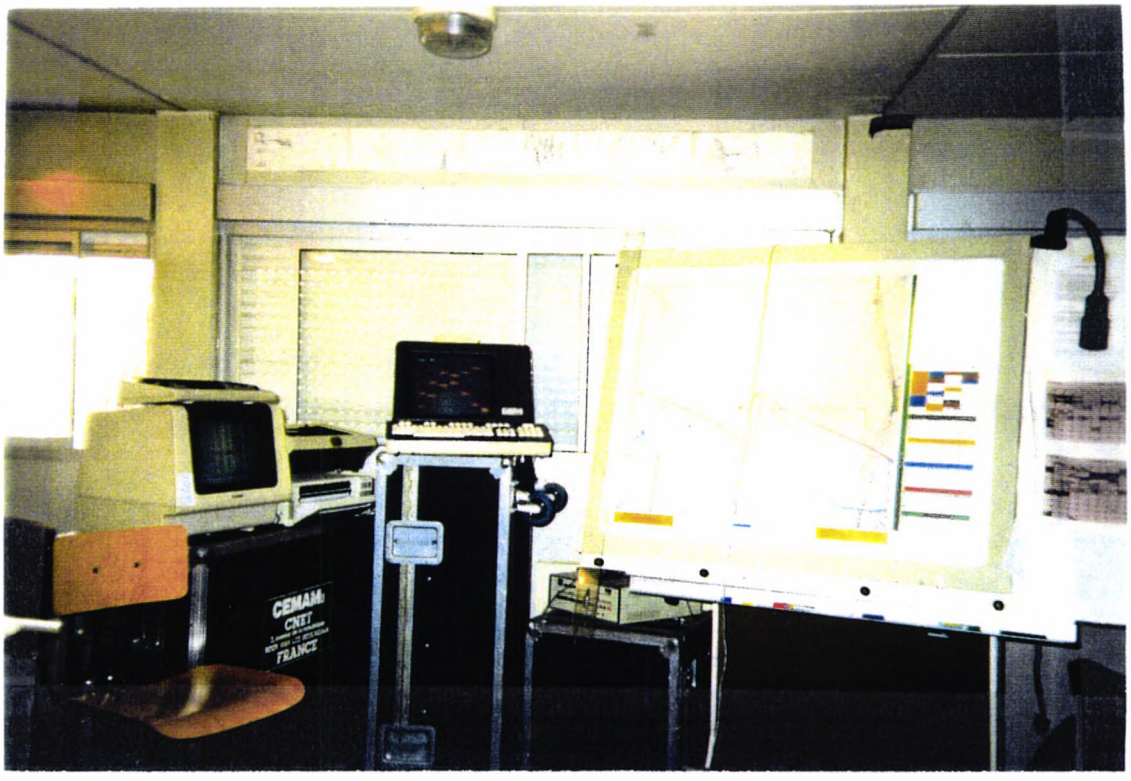



Figure 2.4 UPIC Graphics Board Composing Interface

The list of available physical interfacing tools is long and varied. The process at the design stage of matching the ideal physical interface to the end user is an involved one, and one upon which the success of the system will largely depend.

As we have seen in Diagram 2.1, the virtual and physical interface, in whatever form, undertakes two main types of communication:

- Messages from the User to the Machine

Input messages from the user to initiate a task, or perform a function, as an action (initial input) or as a reaction (input in reaction to data output from the machine).

- Messages from the Machine to the User

Output messages from the machine in response to user input or action. The message may request further information in order to process a task, or perform a function, or display data. The user will perceive the change in system state and react to it in some way.

Once the forms of communication have been defined, and the language of communication and data input have been decided upon – semantic and lexical considerations – the designer is better placed to produce an interface which is not alien in language and application to the user. In the case of too many current systems it appears that the design of the front end, or virtual interface, has taken a very low place in order of priority to processor programming and algorithmic data structures in the design plan.

Consequently, the incorporation of poor quality interfaces result in systems with low user satisfaction, and low performance rates. Several types of virtual interfaces exist on the front end of complex composition systems. Figures 2.5 and 2.6 below depict two of the software interfaces which the author has had access to.

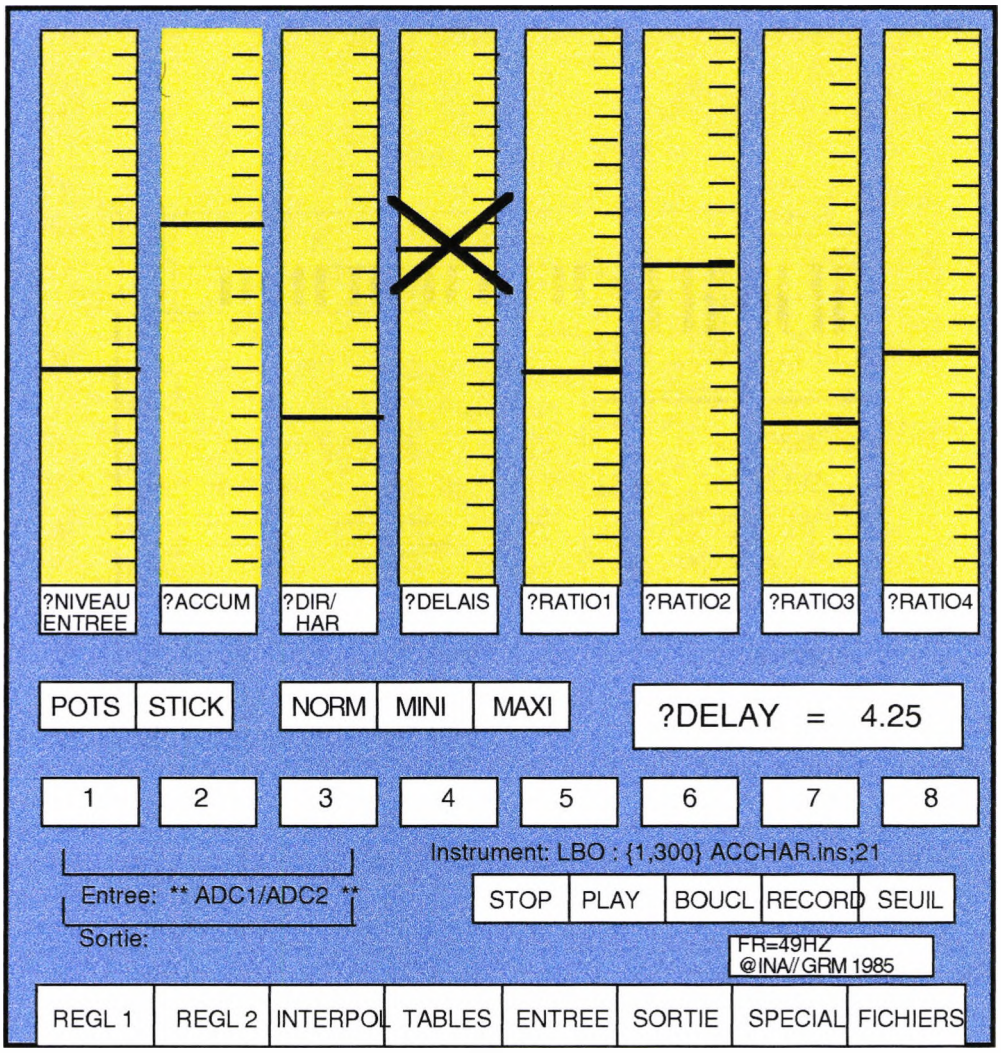



Figure 2.5 SYTER Composition Environment Screen

The SYTER composition environment, designed at the studios of the Groupe de Rechérches Musicales (GRM) in Paris, affords the composer real-time composition and sound manipulation within the

digital domain. This screen depicts the graphical control method employed to vary parameter values in real time within a sound file.

The front end screen of the Lick Machine software package, designed at the Steim studios in Amsterdam, is seen below in Figure 2.6. This figure displays as an overlay several 'windows' or control screens of the Lick Machine software package, overlaid on each other for illustrative purposes.

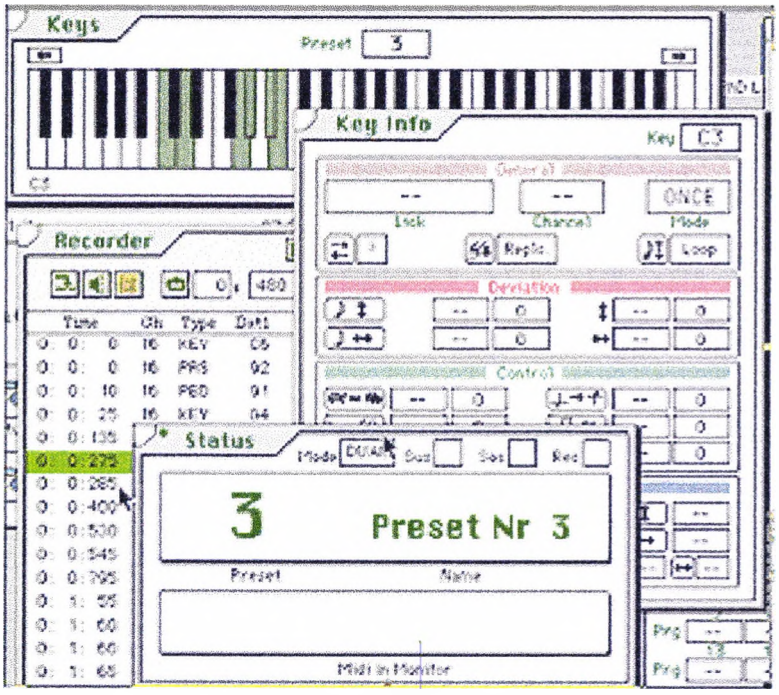



Figure 2.6 LICK Machine Software Layered Screen Image

The Lick Machine is the software control environment for a system which may be employed to create sets of performance data in the MIDI event-based protocol. The composer then has the facility to control these sets of data, or 'licks', in real-time in performance, with physical MIDI interfaces including those specially designed at the studios, such as the MIDI Conductor seen above in Figure 2.3.

The design of the human computer interface (HCI) must be considered at the earliest stages of systems development in order to provide a friendly front-end for the user. The HCI design must be the logical next step after complex user and task analysis, and production of prototype task structures[2]. If the HCI is considered at this stage, and continually readdressed throughout the consequent design and programming stages, then the result will be a system which matches not only the task requirements for the system, but also the users' ergonomic and psychological profiles. The impression which musicians will derive from a system made available to them will depend to a very large extent upon the quality and friendliness of the interface they are presented with, and the ease with which they are able to achieve their musical goals.

[2] Task structures, ergonomic and psychological profiles of users etc. will be introduced and discussed in detail later in this thesis.

# Chapter 3

# The Notion of the User and the Nature of the Task.

In any systems design it is essential that the designer has as defined an idea as possible about the end user of the system he is designing, together with a clear understanding of the nature of the tasks the system is to enable the user to complete. With this knowledge the designer is well placed to make informed assumptions on behalf of the user in the early design stages of the interface before presenting prototypes for testing. In the light of these requirements the present chapter concerns itself with developing a psychological profile of a computer systems user made specific to musicians as far as is relevant. Also, the nature of the tasks involved in the use of a music computer system are discussed, with task characteristics and structure being defined. A prototype task data flow diagram is drawn up, to be employed in later analysis and design discussions.

## 3.1 The User

### 3.1.1 Definition of the User:

#### Control of the Computer CPU by the Human CPU

There are several strands of information that are required by a systems designer who is attempting to put together a profile of a user prior to designing an appropriate interface. By creating

psychological profiles, and by conducting ergonomic analysis upon a community of intended users, the designer can gain a comprehensive understanding of the nature of the prospective end user. As we can see in Figure 3.1 below, in a section of a flow chart representing the process of interface design (after Sutcliffe, 1988), the designer requires detailed specifications and classifications relating to the user in order to proceed with an appropriate interface design.

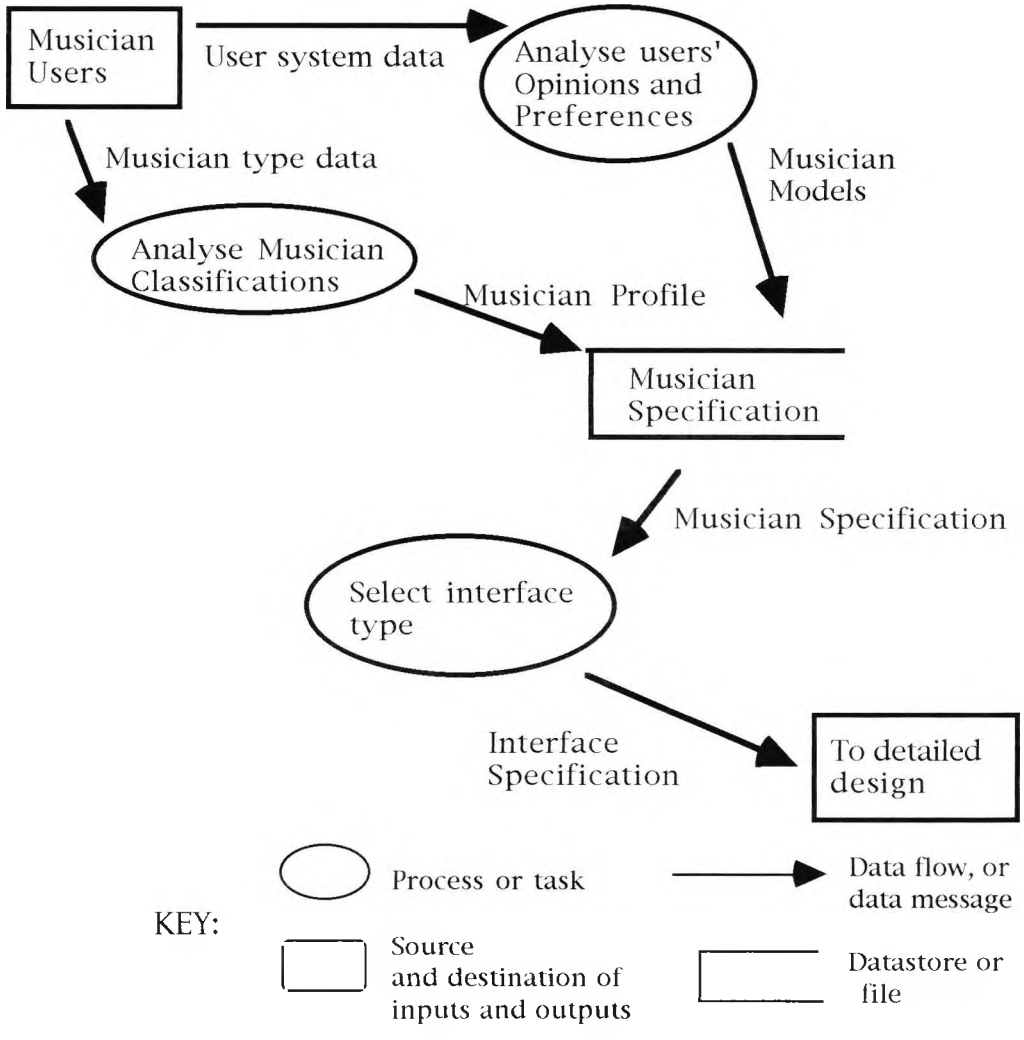



Figure 3.1 User Section of Interface Design Flow Diagram (adapted from Sutcliffe, 1988: pg 51.)

Combining these analyses and profiles with user classifications which allow for novice as well as expert systems users to be considered, the designer will be in a strong position from the point of view of user requirements and abilities, as well as user preferences, before making any decisions about interfacing types for the system in design.

### 3.1.2 Psychological Profile and Ergonomic Analysis.

As Thomas Landauer (1988) writes:

> " The human mind is an artefact of human culture. Although it is not constructed by the deliberate design of a team of people, nevertheless, it is just one realisation of an infinitely pliable system, programmed by culture, education, the knowledge base of the society and the demands of the tasks and environments in which it must find itself."

A human has physical limitations and psychological tendencies which can be examined, measured and analysed by experiment and modelling. For the systems designer it is essential that these traits be fully understood and accommodated if any human-computer interaction is to be successful. If any psychological or physiological tendencies are not addressed, or if any physical limitations are exceeded, then the quality of interaction may be severely decreased.

Through studies in the field of cognitive psychology, it has been shown that humans have proven methods for using skills, learning, solving problems, remembering detail and perceiving given information. Years of close monitoring research has shown that the human mind will copy an action it witnesses; for example, a child will, by reflex, attempt to put an object in its mouth because it watches an adult do the same with food frequently, and we have all been conditioned to open a door before going through it. When the sun is in our eyes and we are trying to focus on a distant object, we will, by a reflex action, put our hand over our eyes for shade. Interfaces which exploit these methods of skill learning, problem solving and learning by demonstration, will have a very fast learning and acceptance curve. Use and exploitation of already present learning strategies, rather than requiring users to adopt new ones, will increase the likelihood of acceptance of a new interface environment.

Human memory is divided into two types, short-term and long-term. Loose analogies of each of these respectively to computer RAM and ROM immediately help the designer to gain some idea of their nature. Experimentation has revealed key features of short-term, or working memory which can be mapped directly into design considerations. These features include facts related to interfacing issues such as:

- Immediate memory recall for details in complex images is poor.
- Distraction causes forgetting of recently learned material.

- Other inputs impair recall. Supplying irrelevant material during input to working memory makes recall worse.
- Memory recall is better if word and picture are presented together, as opposed to either in isolation.
- People remember in the temporal short term (< 30 secs.), by scanning back along the input, thus last in, first out.

(after Sutcliffe, 1988: pg. 27)

By careful consideration of each of these short-term memory working features, the designer can draw important conclusions for interface implementation such as making decisions to accompany pictures at all times with text related to them, and also being careful to provide no distractions during task learning and completion, allowing the user to work serially with the least amount of simultaneous action from other screen events unrelated to the task in hand. It is also useful to attempt to break down task completion into a set of short, serial actions, so that the user's short term memory is not overloaded. Careful planning of these short actions so that they sensibly lead one to the next will also achieve a fast learning curve. Within music systems close mapping of the actions within a necessary execution pattern to the composer's perception of the task will result in system use which is not unduly taxing in terms of memory load for the user.

Correct knowledge of the operation and psychological tendencies of the long-term memory will also improve the chances of successful human - machine interfacing. The most essential experimental find about long-term memory, useful to the designer,

is the fact that recall is best achieved in terms of association. Text linked with visual images at time of learning will be easily recalled by association when one or the other is presented in isolation. Also the presentation of several routes by which single items can be accessed allows for faster recall, due to the psychological knowledge that a fact with several separate accesses to it, or its execution, is most easily recalled. It is important to realise that recall of various different facts will be severely hindered if a similar visual or textual key has been used to represent each of them. As far as possible, variation of cues should be employed to allow the human processor as much opportunity as possible to categorise and separate information for storage and recall. In the systems analyses which follow in this thesis, we will see several very varied approaches to visual stimuli being linked for association to musical tasks in systems processing.

User psychology involves understanding how information is received, understood, evaluated and acted upon by the human. Perception is the process of receiving information from the outside world, and cognition is the mental activity which includes problem solving, reasoning and learning. It is the intelligent consideration and inclusion of human perceptive and cognitive traits in an interfacing design that makes for a successful work environment. Complex interactions of visual, aural and memory processing have to be investigated in order to gain some understanding of how the human user processes and acts upon received information. An 'Information Processing Model' of human perception and cognition, (after Card et al, 1983) can be seen below in Figure 3.2. Based on

their research at Xerox, Card et al. have built up a model which reveals clearly the various levels of perception and cognition within the human being.

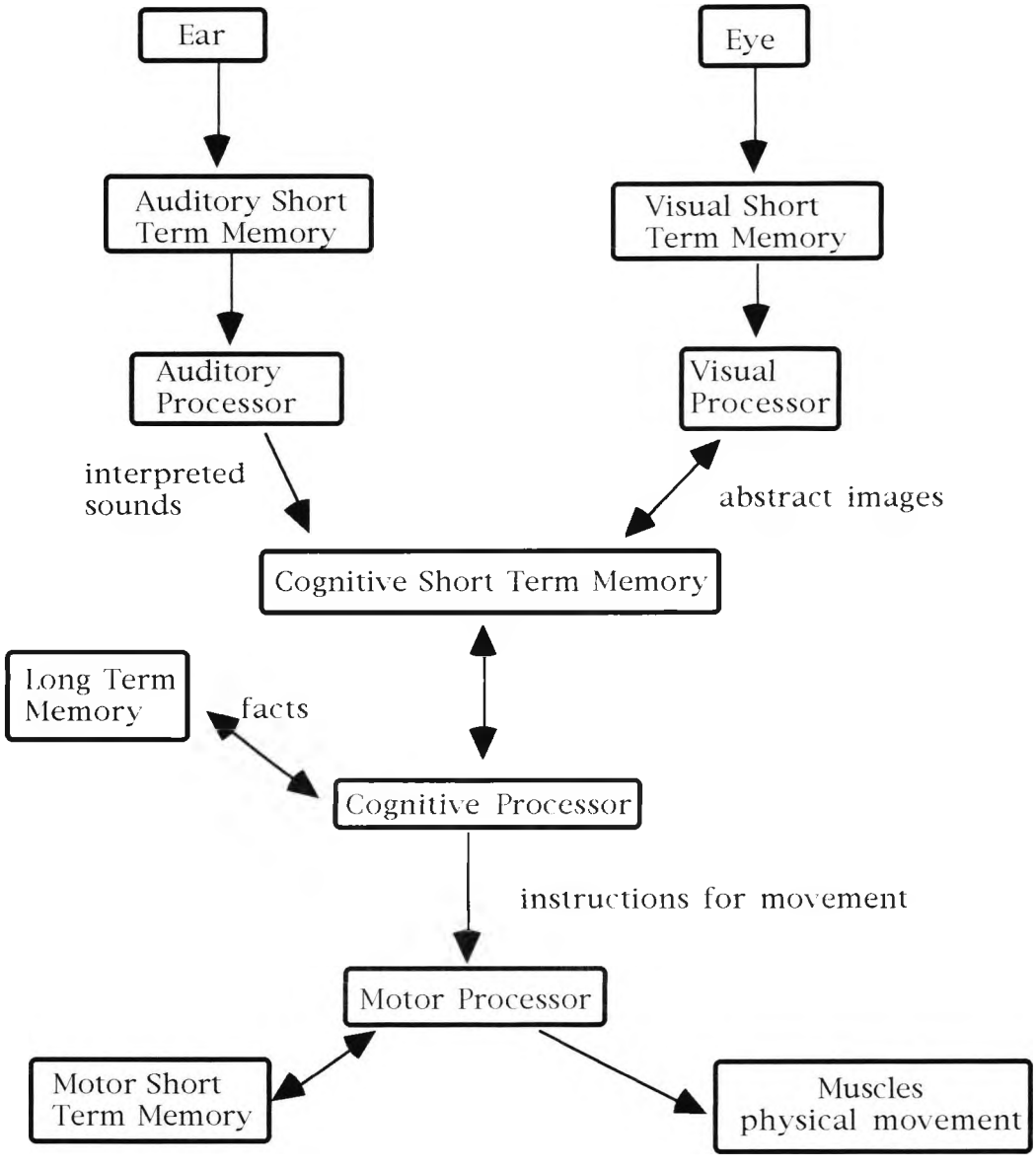



Figure 3.2 Human Information Processing Model after Card et al (1988, pg. 25).

According to the model, each perceptual sense, (e.g. visual, aural), has its own associated processor and short term memory storage

space. Visual and/or aural images are usually temporarily linked – we tend to remember a visual image, for example a racing car going past, together with its aural image, the engine sound as it passes across our sound platform. These linked images are perceived, registered within their respective sensory short-term memory simultaneously and passed to their associated processor. Here, the received information is sub-divided into memorisable facts and physical instructions or reactions to the received information. The facts are transmitted to long-term memory storage and the physical instructions are passed to the motor processor for execution.

A two-way path of communication exists between the cognitive processor and long-term memory to allow for remembered facts to be passed back to the processor at any time to become either physical instructions or additional parameters within cognitive short-term memory, which also has a necessary two-way communication with the cognitive processor. An iterative loop also exists between the motor processor and the motor short-term memory bank, to allow for either repetition or delay in physical action messages being relayed to muscle controllers. What a designer can learn from this model of human perception and cognition is a clear idea about the path of human processing of input data, both from a visual and an aural input source.

This model is particularly interesting to a designer of a system relevant to musician users. If the designer has a better understanding of the interaction of visual and aural stimuli upon the user as demonstrated in the model, then informed decisions can

be made about the combination of visual data being fed to the user on the virtual software interface with the all-important aural stimulus which is the feedback the musician user is most interested in. The constant aural output to the user as a result of a user action to the interface, for example a composer working on a digital waveform, will be aided in its impact if it is accompanied by suitably designed visual stimuli, an area of consideration which includes decisions about the correct use of syntax and language medium for the user in a visual stimulus. We will see in later systems discussions the importance of sympathetic visual stimuli for the composer user of a computer music workstation.

### 3.1.3 Musician User Classifications

For any specific system design, definitions of user populations will simplify the interfacing task by defining the various groups of users for whom the system must function on a 'friendly' level. Musician user populations, identified in the field of interfacing research, include categories for design consideration such as:

- age
- skills
- culture
- education

There are important issues which emerge in each of these user population categories which will effect the decisions being made with regard to a suitable interface for a new system:

- age

The age of the intended user of a system will have some bearing upon the working memory performance which can be calculated, and will also mean that eye function and aural ability may need to be taken into account.

- skills

The users of an intended system will vary with regard to computer literacy and ease of use of the computer medium. With musician users, for example, there will be composers who are comfortable with the computer domain after years of using complex command line systems as in early implementations of CSOUND and PODX systems, and there will be users who may have a very good grounding in digital music theory but who may have little or no experience of such systems use. Also there are novice users who may be, for example, employing a digital waveform lookup synthesis system for the first time as computer users and learning the theory associated with this method simultaneously.

- culture

Within systems design, issues relating to culture need to be addressed in terms of the interpretation of graphic images and the syntax of language in systems messages to the user. Relevant to music systems, culture is reflected in terms of the music notation systems employed, and, for example, in terms of the Western or non-Western scalic systems supported in systems programming, or the system's handling of timbre.

- education

In systems design, the educational background and ability of the intended users must be considered. Differences in educational standards will be allowed for by coding simplicity on the upper levels of operational interfaces. The more naive user should be able to operate the system in a basic, procedural fashion while the more advanced user will be able to access the more complicated functions on lower interfacing levels in the same system. In music systems this may be demonstrated in terms of the levels available in a Midi-event-based system, where first-time or beginner users of a sequencer system can access simple multi-tracking sequencer pages with no difficulty and are subsequently free to access as complicated a level as they feel ready for. For example, the more advanced user might access Midi data byte information and process this individually on a micro level of control.

User classifications for musicians, in which each of the above issues needs to be considered, include classes such as:

- Beginner Musician Systems Users

Those using computer technology for the first time to achieve a musical task.

- Experienced Musician Systems Users

Those who are accustomed to employing computer music systems, whether Midi-event-based or DSP systems, to realise musical tasks.

- Expert Musician Systems Users

Musicians and composers who are proficient in employing computer technology in their musical working environment and also are capable of adapting the system they use to accomplish extended tasks not coded in the original system set-up[1].

In an ideal systems design, each of these user types should be accommodated. A system which allows a beginner user with limited experience of computer systems to achieve simple tasks, and equally is open-ended enough to afford expert users flexibility to extend the tasking ability of the system will have a wide user appeal, as it incorporates user development and allows for increased knowledge to be applied in the workspace. While being approachable by all users across the user ability spectrum such a system will encourage experimentation by beginner users as they work in an 'unthreatening' environment of support.

### 3.1.4 Ergonomic Analysis

In systems design an ergonomic image of the user is essential for the designer to have access to when making interfacing decisions. In Figure 3.3 below several ergonomic issues are considered, mapped together with musical considerations of the system to be designed.

[1]This applies when the system in use allows for reconfiguration of tasks. It applies for example to GRM's Syter programming tasks and to Steim's BigEye scripting. Reconfiguration cannot apply to for example DigiDesign Protools - users cannot configure this but they can apply intelligent music knowledge creatively for example, designing filter set-ups within the system.

| Ergonomic Category | Sub-Categories / Definitions | Considerations For Musician Users |
|---|---|---|
| Time | • learning time<br>• recall time<br>• completion time | Musicians will require extendable completion time for tasks which are open-ended and recursive in nature |
| Concentration | • short term memory load<br>• long term memory load<br>• recognition v. recall | Musicians will require as little memory loading as possible, leaving concentration free for the aural results of each task |
| Functionality | • user-machine mapping | Tasks will need to be intuitively represented, for musician mind - machine mapping to occur effortlessly |
| Acceptability | • Systems appropriateness and proof of making tasks easier to accomplish | Machines will need to afford ease of task completion for musicians to accept the interface with little work involved to divert from the creative task |
| Error Susceptibility | • forward guessing of errors likely to be made | Errors will need to be easily corrected, or in some cases accepted as feasible, as users may retain unexpected sonic results from an error made in input |
| Memorability | • visual and aural linking to aid recall and memorability of interface control | Musicians will require musical syntax and symbols to be encoded in the virtual interface to aid memorability of operations |
| Naturalness[2] | • physical input methods<br>• task processing order | Musicians will work comfortably with a system which fits the musical creative task naturally with few programming intensive steps in the task process. |

Figure 3.3 Musician User Ergonomic Considerations

[2] Naturalness refers to a system fitting to a user's expected model of tasks, as unhindered as possible by system input methods, and physical control methods.

Knowledge of how the user acquires, organises, and retrieves information is essential for design success. Equally important is the complete understanding by the designer of the task to be accomplished by the interface.

## 3.2 The Task

### 3.2.1 Definition of the Task.

For a musician using a computer music environment, the task which he has in mind to achieve with the aid of technology may be very simple, such as creating a hard disk stored image of an incoming sound signal, or it may be very complex, such as the digital manipulation of a signal in some prescribed fashion.

Considering the diverse types of tasks that different users of computer music systems wish to complete, we realise that the designer has to define exactly what range of related tasks a system can viably incorporate in an integrated usable systems environment. According to Pope (1993, pg. 29) some of the tasks requiring computer assistance in the musician's world include:

- Score input, representation, and storage
- Score annotation, mark-up and editing
- Score printing, part extraction and printing
- Exact waveform control, (sample-level control)
- Synthesis device control (module-level control)
- Real-time gesture input (capture) and interpretation
- Graphical interaction with sound, event, gesture and structure
- Sound processing – synthesis, segmentation, processing, mixing

We realise immediately reading this list that one of the problems a designer will encounter, if he limits his task definition research to one source, is bias in the approach to listing computer-assisted tasks. Pope betrays a strong bias towards score creation, editing and printing and then assigns all sound processing tasks to a single categorisation, as "sound processing – synthesis, segmentation, processing and mixing". All of these sub-divisions could easily be a full category in their own right, had the author had an emphasis more on the processing facilities available in some current systems, (such as GRM Syter which has a defined role for processing of signals and almost no symbolic processing facilities at all), and less of an emphasis on the score facilities available in some systems. It may be fair to say that any single music systems user will put more emphasis on a particular field of tasks, perhaps those which that user employs more frequently. It is important therefore for the designer in his early task definition research to consult with more than one source[3] embodying more than one approach in order to create a comprehensive task list.[4]

When we take Popes' classifications in conjunction with the definitions of tasks given by Rodet, (discussed earlier in Chapter 1) we come up with:

- listening
- modelling
- learning

[3] A source in this case is defined as being a system, a user, a research paper consulted, etc..

[4] This does not imply that the designer should or could attempt to incorporate every task in a single system, but he will, as a result of utilising several sources with varied approaches, have a global understanding of the variety of tasks he may be asked to encode a subset of.

- recognising
- transcribing
- playing
- problem solving
- interacting with an instrument
- interacting with a player's model
- interacting with a knowledge base
- composing

and take into account Rodets' subdivision of the last category:

- sound creation
- sound storage
- sound manipulation, in time and/or in space
- timbre alteration
- timbre specification
- instrumental imitation

we may begin to get a more balanced task listing for consultation by a designer.

### 3.2.2 Task Characteristics

When considering task characteristics, we can think for clarity in terms of a macro-task which becomes a series of sub-tasks or micro-tasks, which we need to understand before encoding in a system.

When we begin to examine a task to be completed with the aid of a computer, we must look right 'into' the micro level to

fully implement every stage in its execution. Within the man-machine discourse, we have already described the 'action - reaction' two-way communication which takes place. When this communication is reflected upon the execution of a task, we can begin to serialise the micro level stages involved. As a user approaches a machine to carry out a specified 'action', he will move through a set of stages which may be categorised as follows:

He will:

- form a goal
- specify an action
- execute an action
- perceive the system state
- interpret the system state
- evaluate the outcome

(after Norman, 1988).

Moving through these stages we get a clear picture of the 'action - reaction' coupling which occurs, involving user and machine. The user executes an action upon the machine which will cause some form of a 'reaction' in virtual software image to the screen. The user will then react to this change in system state which he will perceive, interpret and, finally, evaluate, before forming another goal and specifying a new action. It is only when these stages flow as 'freely' as this that intuitive communication will occur. Lack of communication between man and machine at any one of these stages results in a non-intuitive 'action-reaction' chain and will cause user problems.

Within the task structure, there are several elements which must be considered at all times by the designer. It is useful

perhaps to think in terms of a macro task structure involving the following issues for a user:

- cognitive demands
- sequence of operations
- action frequencies
- degree of decision making

As we have discussed earlier, the cognitive load involved in executing an action in a system and subsequently perceiving and interpreting the system state after execution must be considered by the designer. Any user who is expected to spend an inordinate amount of time deciphering a system for a reaction to his action will become very dissatisfied. Equally, if the sequence of operations to be carried out in the system does not closely match the users mental model, then a degree of confusion and frustration will set in. If the user has to repeat laboriously several actions in a row, when perhaps a batch command or some type of system intelligence would have prevented this repetitive action, then a user will, again, become dissatisfied with a system which seems to cause extra work rather than alleviate it. Finally, if specifying an action and then interpreting the reaction causes an increase in decision making unrelated to the task at hand, the user will become fatigued and disillusioned with having to devote decision time and effort to unmusical issues.

When considering the user issues involved with task completion in a system, it becomes clear that the designer is well advised to spend time at the design stage to become as fully conversant as possible with the nature of the tasks his system is to

fulfill. By becoming as familiar as possible with the task structures involved in composition processes, the designer will be able to produce a system which renders the action–reaction line of communication as clear and coherent as possible.

### 3.2.3 Task structure

Looking at defining the steps involved in a task for the realisation of a composition, we can initially think globally, moving from the composer at one end of the scale to the listener at the other. Implementing a model developed by F. Richard Moore, seen in Figure 3.4 below, and employed extensively in later systems evaluations in this research, we can begin to think serially along the path taken in task completion.

In his model, Moore tells us that all the links he makes are optional but that one or more paths must connect the composer at the start of the chain to the listener at the end. Viewing tasks in this fashion is a useful data flow diagram presentation[5] for the designer, and allows for several paths to be taken through the model.

[5] Data flow diagrams are introduced in detail in the next chapter of this thesis.

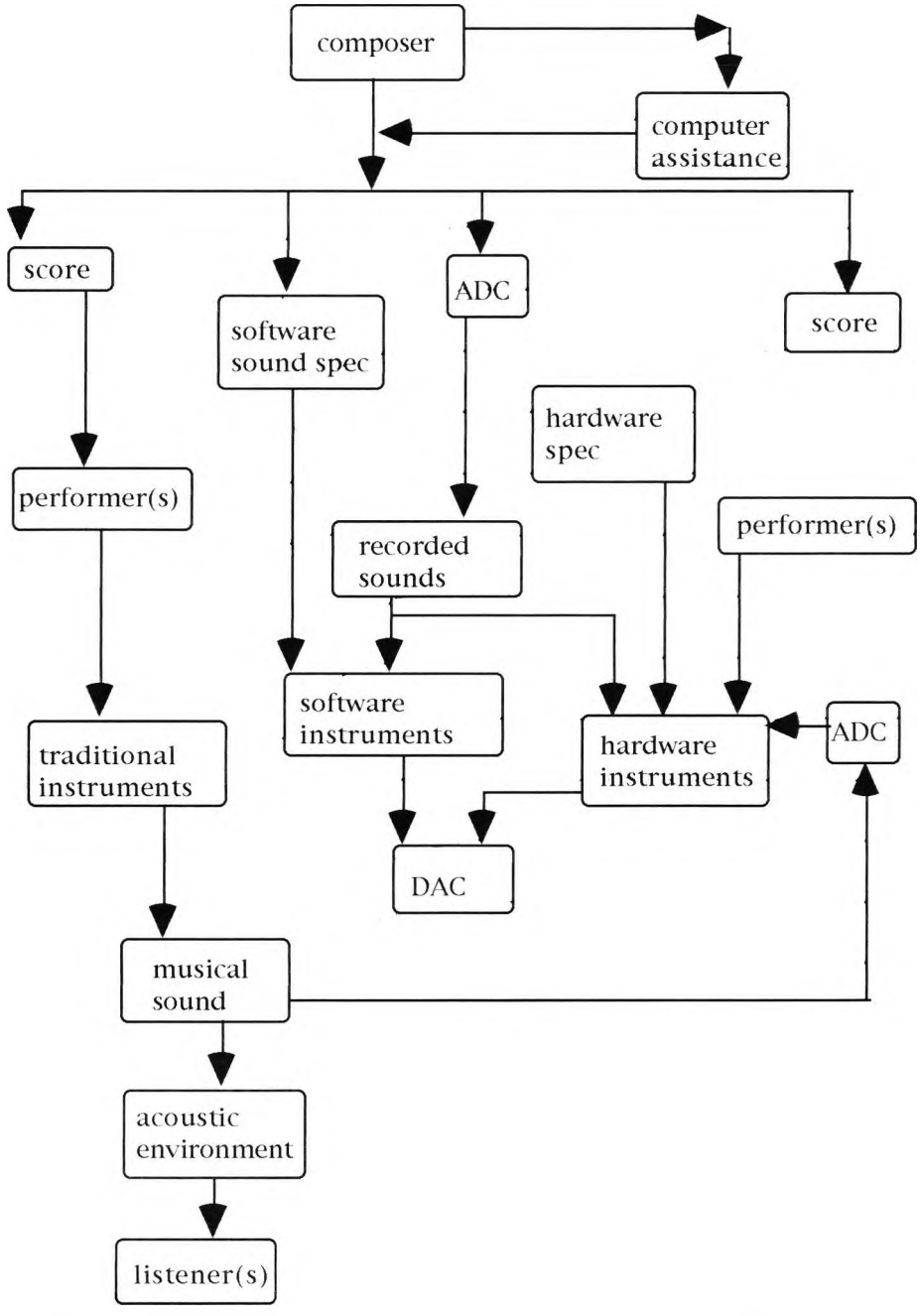



Figure 3.4 **Computer-mediated Music Composition** Model after F. Richard Moore (1990, pg. 399).

In order to implement this model for task analysis, and later as a template of some sort for task encoding within a system, we will need to build in a degree of recursive activity in the serial paths through the information flow. In Figure 3.5 below, a prototype task data flow diagram has been drawn up, based on a macro view of possible tasks within a system designed for digital

manipulation of data with recursive activity allowed for. By considering a prototype task data flow such as this, the designer will be constantly aware of the needs of a composer wanting to repeat various functions, or to experiment with various output signals or data in any fashion at any stage in the global task.

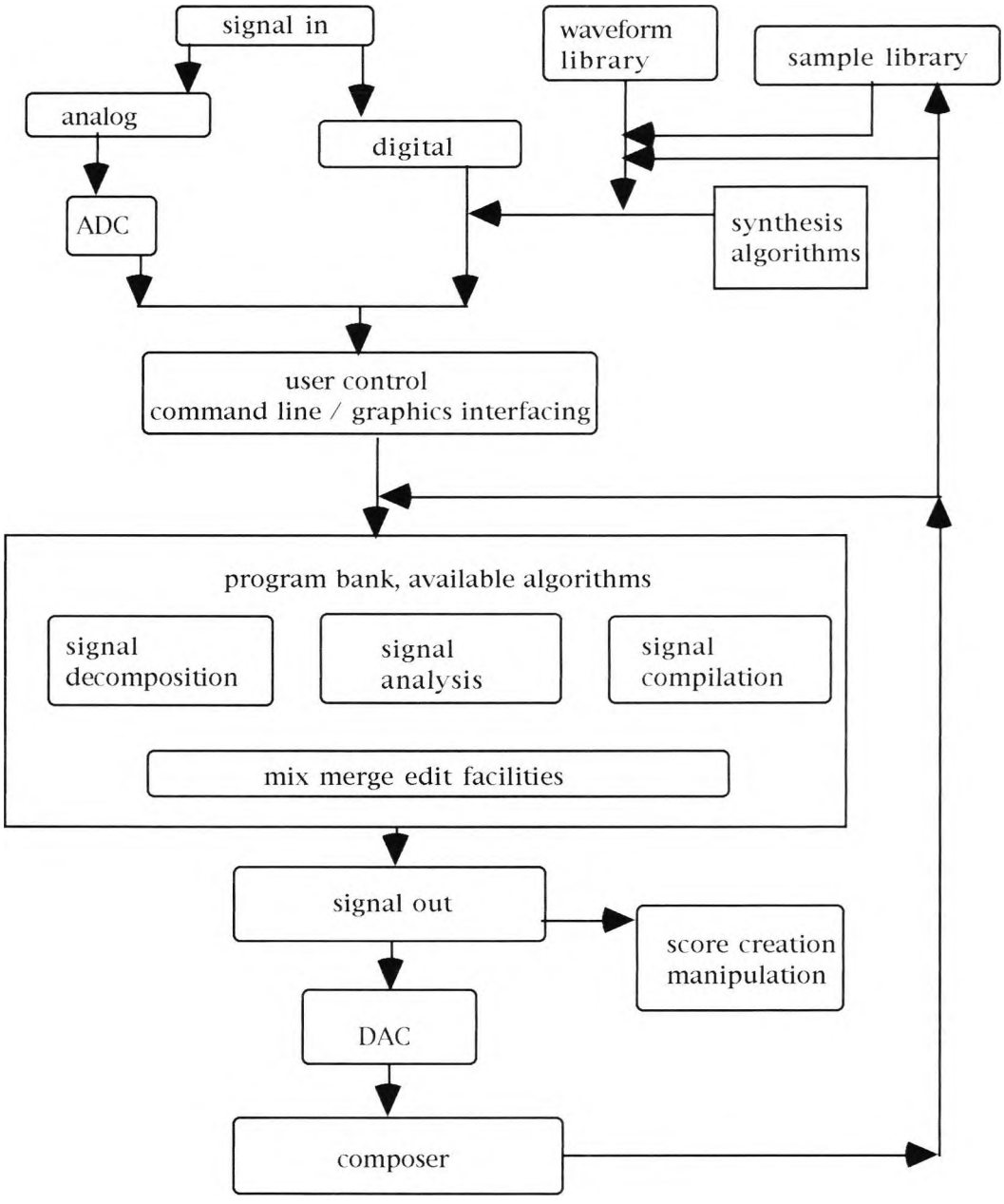



Figure 3.5 Prototype Task Data Flow, Signal Manipulation.

This data flow diagram, and variations of it, will be used in later chapters of this thesis when systems are analysed and evaluated with regard to task handling and implementation.

The relevance of computer systems in the world of contemporary composers is described accurately in the words of Otto Laske (1989: p.47):

> "A computer program for composition contributes structure to a composer's task environment; it informs his planning and transforms his desk into a controllable environment. Most significantly, a program enforces – or at least enhances – the composer's awareness of his problem solving process, since only when he understands his own process can he determine the optimal entry point of a computer program into that process".

It is clear that the establishment of a two-way open and fertile communication path between composer and designer will only serve to result in the creation of a computer composition system which will enhance the working environment of the contemporary composer.

# Chapter 4.

# Overview of Current Analysis and Design Methodologies.

## 4.1 Introduction.

In an overview of current design and task analysis methodologies, it is necessary to qualify certain actions which are at the heart of our concerns in this research. The action of a user, employing a machine for whatever purpose, creates the notion of transduction of energy. As Kirakowski (1988) writes, in a process he titles "Human Computer Interaction: from Voltage to Knowledge",

> "Every tool has three important conceptual components. One is the thing that the human holds, the control surface. Another is what makes things happen, the effector surface. A tool is therefore a control surface linked to an effector surface, and movements at the control surface are transformed by the tool into happenings at the effector surfaces. This transformation is what we may call transduction; the third element is therefore the transducer."

The fields of cognitive ergonomics, design specialisms and task analysis expert sciences are concerned with the nature of transduction of this energy, with the overall aim of making this transduction process as simple, fault-free and untaxing for the user as possible.

The methodologies which are introduced in this chapter are not representative of the entire field of design and analysis techniques available but are a selection of those in use at the

present time within the science of cognitive engineering and ergonomics. The methodologies discussed are those selected by the author to be introduced to the reader, applied to the area of music systems analysis in subsequent chapters and considered later in this thesis when a system design is being carried out in Chapter 11.

The methodologies to be discussed, examined and applied to music systems design objectives are:

- Systems Engineering Methodology (SEM)
- Data Flow Diagrams (DFD)
- Jackson Structures
- Rapid Prototyping
- Soft Systems Analysis (SSA)

Some of the design methodologies listed above are embedded within a larger overall design methodology, *Structured Systems Analysis and Design Methodology* (SSADM). I have extracted and further researched in isolation these subsets of SSADM as deemed appropriate to this research. The SSADM methodology in itself is not deemed relevant in its entirety to the development of a design methodology for computer music systems and environments.[1]

[1] SSADM, (1988), includes cost production forecasting, rich pictures, and other sub-sections of the methodology not deemed to be relevant to the analysis and design of computer music systems.

## 4.2 Methodologies

### 4.2.1 Systems Engineering Methodology

Defining systems engineering as:

> "...the task of conceiving, designing, evaluating and implementing a system to meet some defined need."
> (Jenkins, 1969: pg 7)

Jenkins (1969) defines a methodology in four sections:

- Systems Analysis
- Systems Design
- Implementation
- Operation

He subdivides these sections further:

- **Systems Analysis**
  - Formulation of the Problem
  - Definition of the System
  - Definition of the wider System
  - Objectives of the System
  - Definitions of Economic Criteria
  - Information and Data Collection

- **Systems Design**
  - Forecasting
  - Optimisation
  - Control
  - Reliability

- **Implementation**
  - Document Approval
  - Construction

- **Operation**
  - Initial operation
  - Retrospective Appraisal
  - Improved Operation

This methodology, as the first of five to be introduced and discussed here, is perhaps the most readily accessible to the musician user, attempting to advise a designer about his needs in a system. In a point-by-point fashion, working with the designer, a user could clarify a task, defining the system objectives. The designer would then proceed to codify that task.

This methodology could work on a very simple, concise level when used in a working situation which affords constant communication between the user and the system designer, together with a highly developed set of task analysis prototypes and data-flow diagrams of user-machine interaction and transduction of energy from the user through the control to the effector surfaces. Within music system design this methodology would be useful almost in its entirety. In Chapter 11 of this thesis, parts of this methodology[2] will be employed in a design situation – formulation of the problem, objectives of the system, retrospective appraisal in an iterative design process and forecasting using accrued knowledge of user communities, all sub-sections of the Systems Engineering Methodology, will be used during the design stages for the creation of a musicians user interface.

### 4.2.2 Jackson Structures

Originally used by Jackson in 1975 for program design, Jackson Structures are commonly used to represent possible orders in

[2] Certain sections of this methodology relating to economic criteria and document approval are deemed not to be applicable to the design issues discussed in Chapter 11 of this thesis.

which things may occur. They are a useful graphical representation of the order in which data is changed, received or sent within an overall order of events. They also reveal a processing logic which a designer can employ for the development of program code much later in the system design process.

All Jackson Structures use three integral components:

- sequence, reading from left to right;
- iteration, at any stage built in;
- selection, from two or more options, where choice is a possibility;

Figure 4.1 overleaf is an example of a Jackson Structure. In this example, I have applied the activity of recording a natural sound source via microphone input into a sampler. In this very basic example the sequential flow of activity becomes apparent, together with the hierarchical level of processes involved. As we can see from Figure 4.1, the decomposition of the processes reads from the top down. Iterations are built into the structure by means of a '*' within the box. Possibilities or choices to be considered before action is taken are shown with a 'O' in the boxes concerned. Conditions can be attached to iteration if necessary.

It can be seen that the main benefit of Jackson Structures arises not only from the fact that they may be applied to any formal structure, but more importantly, that they contain the essential elements of procedural programming languages and therefore, for the designer, the gulf between expressing logical structures in this diagrammatic, sequential format and actually

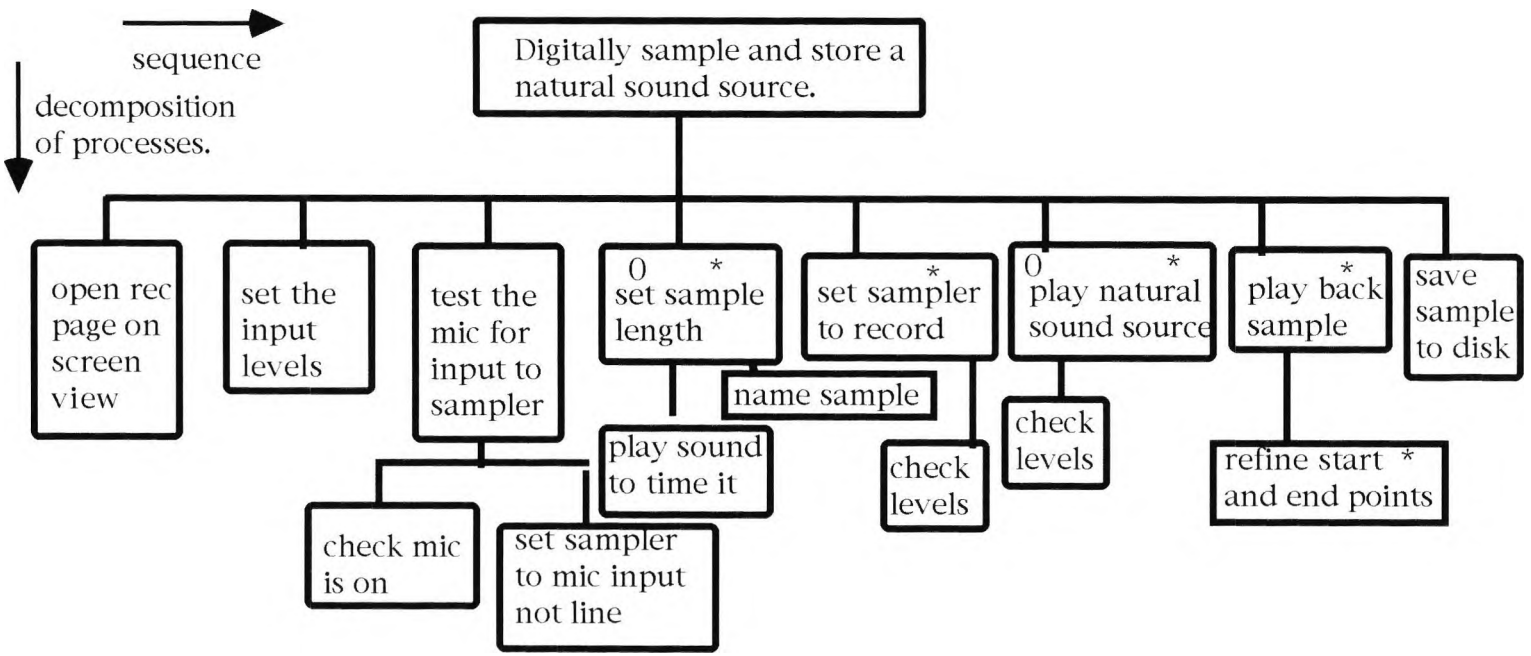



Figure 4.1 Example of a Jackson Structure, digital sampling, playback and storage.

encoding the task for computer use is very narrow. Jackson Structures will be used in Chapter 11, prior to the construction of DFDs, to clarify straightforward task structures within a computer music working environment.

### 4.2.3 Data Flow Diagrams (DFD)

Data flow diagrams, constructions of boxes and circles which contain small amounts of information interconnected in some fashion, have been in use in several disciplines for a long time as a method of displaying a flow of information in a clear, concise and uncluttered fashion. By their pictorial nature, DFDs are non-technical representations of information which are easy to understand for virtually any reader. They are very fast to construct and can be just as quickly changed. With the correct use of the limited number of symbols, all of which carry specific meaning, DFDs allow for the implicit representation of information, making the description very concise. The top-down nature of the DFD, with the ability to expand in side-tree structures and to show iterative processes, allows for simplistic progress through the information being shown to more detailed information at another level, or to step back iteratively, or to loop through partial processes contained within the DFD.

The conventions for the construction of a DFD within systems design and analysis employ a dictionary containing data processes, data flows and data stores. The graphical conventions are detailed in Figure 4.2 below.

represents something outside the system described by the D.F.D, e.g. a systems user.

represents a process, which manipulates or transforms data. This process may be decomposed to many levels of detail.

represents a data store, ie. information retained by the system, either long- or short-term.

represents a data flow, a single line in one direction, indicating the direction of the flow of data to and from processes. The arrow indicates the direction of the flow to which there is a description normally given.

Figure 4.2 Data Flow Diagram Symbols

There is a hierarchical structure within a DFD, which in itself aids clarity for the reader. The top level, or level 1 DFD, will contain an external entity connected to a primary process which then expands itself into a level 2 DFD. Processes on level 2 can be expanded at will until enough detail is indicated.

Each process box can be clearly understood if thought of as a window which may open into another diagram, a level below, holding greater detail each level down the reader goes. A useful accepted notation in a process box which does not open any further to another level is the use of '/*' in the corner, to denote that this process is at its lowest level. Lowest level processes can be described in words or mathematical formulae, if necessary. Any information supplied in the narrative in this fashion at the lowest level is known as Elementary Process Description.

As an example of a data flow diagram at this stage I have reduced the basic task of a composer applying an input signal

to a DSP system and getting an altered signal output to a DFD representation. This level 1 DFD is seen in Figure 4.3.

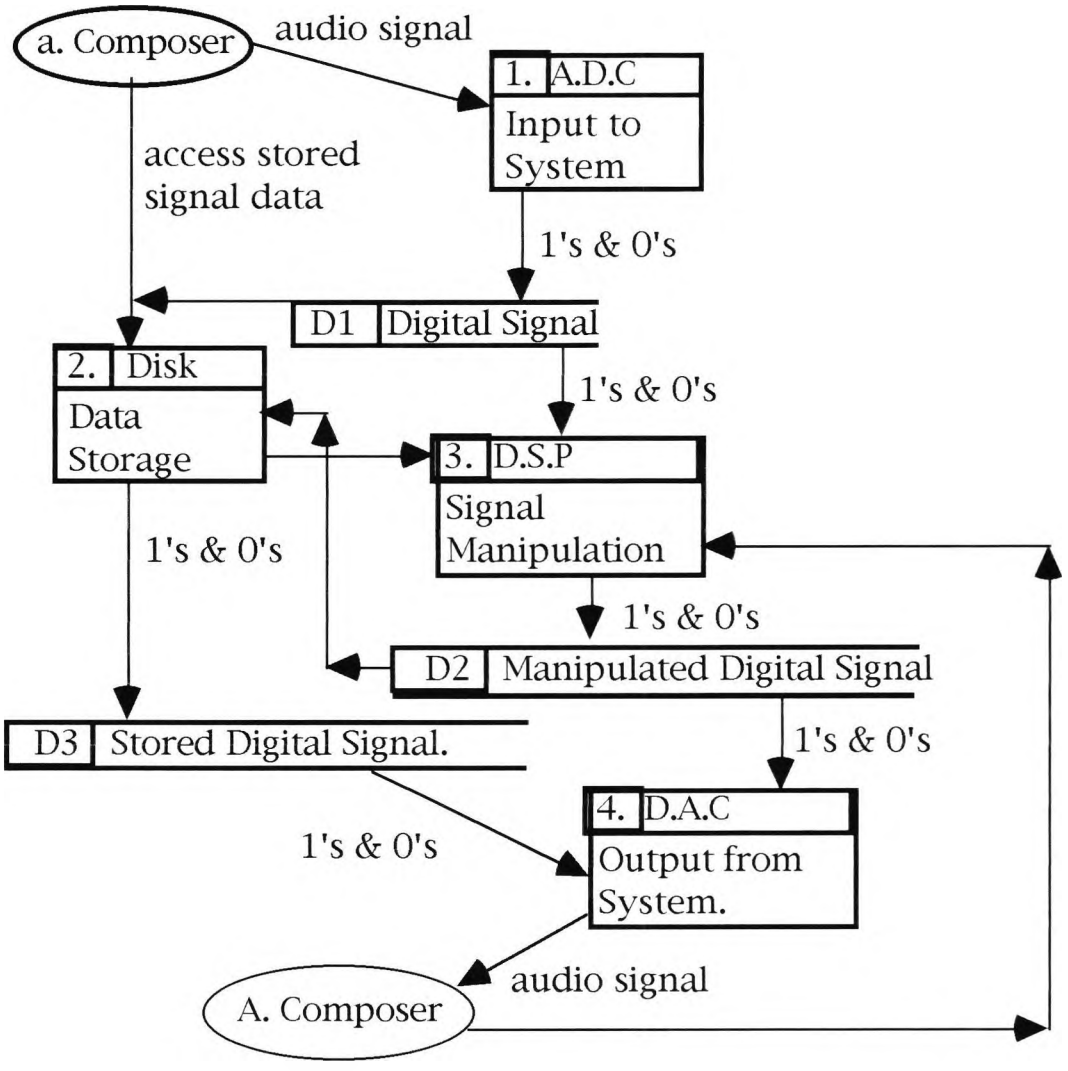



Figure 4.3 DFD Level 1 Composer System Tasking

Data Flow Diagrams are the next level in complexity, following naturally on from Jackson Structures, as a combined method of representing information flow within a system. A variant of a DFD, as a simplified flow diagram, will be seen in Chapter 11. This

as a simplified flow diagram, will be seen in Chapter 11. This variant is applied in a physical working design, to clarify task steps in a specific user interface discourse.

### 4.2.4 Rapid Prototyping

Used as a method of early error detection in systems building, thereby cutting unnecessary costs and improving relevance, rapid prototyping is used to establish true user requirements in an attempt to design and eventually deliver a relevant system at the end of the design process. Within rapid prototyping the functional requirement of the end product is the main concern with the aim being to produce a working demonstration model as early as possible in the overall project time. This model should then ideally be updated constantly and given back to the user for testing and feedback.

The prototype created employing this methodology will only have some of the features of the proposed system and will then inevitably suffer from inadequate performance on several levels. At all stages, however, the prototype should ideally be as easily modified as a DFD, as this will allow for fast turn around and implementation of each update. In current systems design the prototype would be allowed to evolve due to an iterative process around what has been developed as a Rapid System Development Loop, or RSDL, which contains the following stages:

- Rapid Analysis
- Rapid Functional Specification
- Rapid Architecture Design
- Database Creation
- Menu Creation
- Creation of Functions
- Prototype Demonstration
- User Approval

(after Open. Univ. Press. (1984)).

Prototyping has been recognised as an essential part of the design process for many years but in order for it to be fast and not extensively energy draining, the right tools must be available to carry out programming in an acceptable time frame, and to allow for iterative improvement and alteration just as quickly. Although actual prototypes are deemed to be outside the remit of this research, some sub-sections of the R.S.D.L, specifically functional specification, creation of functions, prototype demonstration and user approval will be applied in subsequent systems design in Chapter 11 of this thesis.

### 4.2.5 Soft ( Versus Hard ) Systems Analysis

Within the systems analysis field, there have developed two essentially divergent methods. Before adopting the basic ideals of the so-called 'soft system' approach in this research, it is worthwhile pointing out the differences between the two approaches in order to justify the choice.

Hard Systems Analysis has been developed as a procedure which, as Checkland (1984: pg 11) writes, rests on

> "...the belief that real-world problem solving can be formulated as a search for optimum means of achieving desired, agreed upon goals"

In hard systems analysis everything is viewed in terms of a problem which must be solved by means of specifying goals, monitoring performance and subsequently measuring the success achieved. As Checkland (1978: pg. 22) tells us:

> "....there is a desired state, $S_1$ and a present state, $S_0$ and alternative ways of getting from $S_0$ to $S_1$. Problem solving, according to this view, consists of defining S1 and S0 and selecting the best means of reducing the difference between them. Thus in SE, [systems engineering], (S1 – S0) defines the 'need' or the objective to be attained, and (ADM), [Aids to Decision Making], provides an ordered way in selecting the best among alternative systems which fulfil that need. The *belief that real world problems can be formulated in this way is the distinguishing characteristic of all 'hard' systems thinking."* (Italics in original).

This hard systems approach makes extensive use of simulation and quantitative models with strict measurement of performance at every stage. At all times there must exist a strictly defined target.

In complete contrast, the soft systems approach has developed in response to a failure on the part of the hard system, goal-oriented, monothematic, performance-related method to deal with problem situations which involve human perceptions and behaviour as part of the measurement of success.

Within the realm of soft systems analysis the emergence of the 'human factor' in the design process and in the interaction with a system in a problem solving environment has taken precedence over the monothematic hard approach of finding a solution to a problem. Soft systems analysis has much more flexible boundaries than the hard systems approach and allows not only for consideration at all levels of the human component in the problem solving chain, but also for the fact that each human user will be different, mentally, physically and psychologically, and may also want to employ more than one route through the solution process. Within Soft Systems Analysis the following points are held to be valid and acted upon during systems analysis and design:

- problems do not have an existence independent of the human beings formulating and involved with them;

- people have different appreciations of situations and settings because they see them in personally distinct ways;

> "...each of us looks at, or interprets the world through a kind of personal prism which refracts the light of experience and gives us a unique picture of 'reality' "
> (Open Univ. Press. 1984, T031, pg. 11)

- If problems are personally formulated and defined, then the solutions are personal to just the same extent and will vary for every person;

Within the definition of a problem as 'a doubtful or difficult matter requiring a solution' (OED, 1990), Soft Systems Analysis makes room for the fact that we may be able to fully define the problem, but that rarely will many people agree on a single route to its solution. Allowing as it does for human choice and variance of opinion from one individual to another, the soft systems approach is suited very well to the development of user friendly systems which incorporate a degree of human choice and independent thinking. It is accepted that, with Soft Systems Analysis, a sharing of perceptions must be accounted for. It quickly becomes apparent that a Soft Systems approach will lend itself very well to the design and analysis of music composition systems where solutions to a problem are rarely definable, and indeed the musical task is rarely thought of as a 'problem' at the outset.

The soft systems analysis methodology, after Checkland (1982), breaks down into seven basic stages of which the analyser becomes an integral part and is not in the role of merely solving someone else's problems:

Stage 1: The Problem Situation Unstructured

- First encounter with the problem situation
- Start to become a part of the problem
- Clarify own reasons for being involved
- Identify and consider the roles of client, problem owner & solver

- Experiment with looking at different people in these roles.

Stage 2: The Problem Situation Expressed – Rich Pictures.

- A graphics based cartoon type presentation of the task at hand is drawn up, rich in information about the tasks to be achieved. Pictures can relate much more information than paragraphs of text.
- From the information contained in the rich picture, the analyst can portray the primary tasks and the problem themes.

Stage 3: Root Definitions of Relevant Systems

- Name the Relevant systems
  These are names of systems which address the problem themes identified in the rich picture.
- Root Definitions (considering users, end product).
  Build a sentence that completely defines what the proposed system has to do. As Checkland (1978, pg. 22) states:
  "A concise description of a human activity system which captures a particular view of it."

Stage 4: Construction of Conceptual Models

- Build a model of activities or processes that must go on if the system is to perform what the root definition states. This stage deals with *what* the system is to do in terms of processes, not *how* it is to perform those processes. The conceptual model is built from the list of verbs describing activities stated in the root definition. The relevant system

relates directly to the root definition, which in turn relates to the conceptual model, in turn relating back to the relevant system in a loop.

Stage 5: Comparison of Conceptual Model with Rich Picture.

- In order to detect similarities and discrepancies, an iterative loop is set up whereby the conceptual model is tested against the contents of the earlier constructed rich picture.

Stage 6: Debate with the people involved with the situation.

- discussing the state of play to date with the end user, the problem owner, and the problem solver, and any others available who may have a bearing on the proposed system. The aim is to derive any ideas for change at this stage both from a systems point of view and from a personal view-point.

Stage 7: Implementation of any agreed changes.

- At this stage, any agreed alterations will be incorporated into the system, including changes in procedures, processes and structures and, on the personal level, any alterations in attitude to the solutions provided by the system.

At this stage the analysis of the system, with complete regard to the personal nature of the user and to the fact that there can be several solutions, is complete.

Later in this thesis, with the application of the soft systems approach to systems analysis and design discussions, it becomes clear that this approach works effectively when considering the current systems available to computer music users. There is little or no room for the hard systems approach which calls for strict performance related success rates and the definition of a problem in terms of goals to be achieved. The soft systems approach, which allows for a varied user with permissible variants regarding solutions to not completely defined problems, suits the grey areas of task analysis within computer music systems perfectly.

In the design of a user interface in Chapter 11, the Soft Systems Analysis methodology will be seen to be consistently employed, as issues such as clarifying primary tasks, defining the system role, debating with the user, the designer and others involved in the process, and implementing commonly agreed changes are incorporated in the overall design task.

The systems analysis and design methodologies introduced in this chapter will be employed, consciously and sub-consciously, in analysis sections of the five systems chapters in this thesis. Variations of the design methodologies adapted and, in some cases amalgamated, to best suit the design task at hand will be employed in Chapter 11.

# Chapter 5 Computer Music Systems: Analysis and Design Guidelines.

## 5.1 Introduction.

With a user who requires open-ended and recursive task definitions, and tasks which may constantly vary in structure and nature of intent, the job of designing a computer music system is a wide ranging one. As Pennycook (1985: pg. 268) writes:

> "Many of the interface requirements for a music system are unique to music and have no equivalent within the general realm of computer usage. Furthermore, composition of music, performance of digital musical instruments and manipulation of recorded sound each pose new and substantially different problems for the user-interface designer ... composers, performers and recording engineers usually exhibit highly idiosyncratic work habits, which ultimately must be accommodated by the user interfaces on an individual basis ... as a result of the rich descriptive vocabulary of musical concepts information about music and sound is not easily translated into computer data that is accessible to and readily manipulated by musicians."

However, the job is made approachable with logical analysis of the user and the tasks to be performed. Together with this information, the designer is well advised to perform analysis of already existent systems in an attempt to draw both positive and negative pointers with regard to the creation of a user-friendly environment. To understand the interaction of the user with the system, the aim of the user employing that system, the nature of the task the user wishes to accomplish, and the discourse between

the user and the machine, together with a complementary listing of analytically derived models of success from systems already in use, will begin to define and refine the design task.

The aim of producing a set of analysis and design guidelines is to provide a reference source which may improve the efficiency, user-friendliness, learnability and user acceptance of a new system. Having composed with, analysed and evaluated several systems, some constants regarding design and analysis criteria have emerged for the author. Just as these have emerged from the examination of very varied system types, across the spectrum of user input styles, compositional hierarchies, and user abilities, so they may be applicable to the design of varied systems, irrespective of hardware developments and independent of specific platforms.

Throughout the chapters of this thesis, very varied source materials have been employed, ranging from the specific manuals of the systems discussed and analysed in Chapters 6 to 11, to software engineering articles, physical and cognitive science studies, to statistical analysis, to reports on the design of the Xerox STAR[1] System, and the design of the 1984 Olympic Message System[2]. While several such streams of scientific study and interfacing design were examined and are incorporated in this research, the body of the work and the original impetus for the project is that of a composer, employing computer technology to achieve the realisation of composite musically related tasks.

1 Designing the STAR User Interface, Smith et al, (1990).
2 The 1984 Olympic Message System, Gould et al, (1990)

## 5.2 Design Guidelines

The design guidelines will be presented in the tri-sectional format suggested by the research and discussions contained within this thesis[3]:

- general interfacing guidelines;
- guidelines relating to the system user;
- guidelines concerning the nature of the task to be undertaken within the system.

These guidelines will be employed in whole or in part in subsequent systems analysis, evaluation and design in the ensuing chapters of this research thesis. In this way the validity of the guidelines being presented may be seen, as they are used to analyse and evaluate several existing computer music systems (Chapters 6 – 10), and also employed in the design of a user interface (Chapter 11). Guidelines which have emerged directly from issues discussed in the previous chapters have chapter and page indicators listed with them, to facilitate referencing to source material.

[3] Although the guidelines are presented in three sections, it is inevitable that a degree of overlap will occur.

### 5.2.1 General Interfacing Guidelines

Use natural sentences and natural language words.

*Map the language employed on the interface to that of the user community analysed during the design phases – this will enhance familiarity and create a more secure working environment. Where possible, use musical syntax and conventions to encourage the user in a familiar environment.*

Combine the presentation of text with graphics where applicable (3:44).

*The human memory processor recalls details easier and faster if they have been presented in both text and graphical form together. As far as possible, musical graphical notation and conventions should be employed to exploit user conditioning and natural expectations.*

Choose the input device carefully, considering compatibility with system and user, accuracy within the system, efficiency and ease of use.

*Suit the physical input method to the user, measured in terms of familiarity, physical size of user work space, and totality of the system.*

Design a consistent well organised interface.

*Menu structures, message format (help, error etc.), screen layout design and colour use should be unambiguous and easy to comprehend, with no confusion possible.*

Provide a clear uncluttered screen.

*Aiding visual clarity will speed up use and interpretation of the system. The musician user will be interested in details referring to the task at hand, and does not need the creative goal to be further clouded by unnecessary information.*

Simplicity should be aimed for.

> *"Simple things should be simple, complex things should be possible." Kay, (1990, pg. 253)*

*When the main objective at hand is a creative one, the interface should at all times make task completion as simple and straightforward as possible, in order not to detract from the creative goal.*

No more than 25% of the screen should contain information directly related to the problem at hand.

*Aid learning, useability and user clarity by not overloading the users' visual perception of the system, by confining the active screen area to about 25% of the whole.*

Keep colour usage to a minimum – recommended to employ up to 4 grey scales in a monochrome design, and up to 8 colours in a colour interface design.

*As seen in user cognitive and psychological studies, cognitive load is lessened by controlling the number of greyscales or colours the user is asked to identify and recall.*

Reliable colours are red, blue and green, with purple and yellow as second choices, with use of colour held consistent across the interface.

*Particular colours are more easily recalled by the human visual system, and are easier differentiated from each other. Colour palettes of closely related colours will cause unnecessary confusion and cognitive load.*

Always have the main menu available.

*With the provision of a constant or consistent keypress, or a displayed option, the user should always have immediate access to the main menu, and should not have to step through several sequences of key presses to return to it.*

Menus should only display currently available options.

*If particular options within a menu are not available when the user is within a certain window, or level of the program, then those unavailable or unexecutable options should not be visible, or should at least be grey-scaled down. In this way the user will not waste time accessing an unavailable option.*

Place warnings, data to be monitored and urgent messages in the top right screen quadrant, where users tend to recognise information more quickly. Place routine information not needing constant monitoring in the bottom right quadrant where users are less distracted.

*Information urgently needing to be drawn to the users attention should be visible or appear in the top right of the screen. Conversely, details relating to system versions and unrelated data to the task in hand should be placed if required to be visible at all in the bottom right of the active screen, where the user will not continually be drawn to process it.*

Provide a fast retrace option in the system.

*Users should be able to go back to a previous screen with the use of a single keystroke – either mouse or keyboard – screens should not be hierarchically accessed backwards in a menu. On graphical interfaces, a menu of available screens should be apparent.*

A system design should advocate stability in processing style and generic view.

*When a system is released in upgrade to a user community, the steps between old and new versions of the interface should be measured and achievable in a secure environment of familiarity. Between successive releases of software packages, the user should be able to retain a high level of familiarity, and be led into the updates via tutorials and on-line demonstrations.*

The virtual software created interface should shield the user from the computations happening at lower levels within the system (2:31).

*A user-friendly interface will be as free as possible from presenting the processing information lower down in the system levels - the interface should concern itself with facilitating successful user interaction, and not displaying unnecessary specific systems functions. Musicians do not need to see data being processing during a mixing operation, or to watch data being computed on the screen. The aural output is the creative goal, and computationally how that goal is reached is rarely of interest to the average user.*

Provide shortcuts on the interface for experienced users.

*To avoid frustrating more experienced systems users, provide keyboard shortcuts mapped to menu items, or the ability for high level users to embed macro functions within the system; Give access to more advanced features via hierarchical menu structures or add-on software modules; Allow for the use of user compiled macro commands at all levels to speed up the compilation of frequently executed tasks.*

Keep management of windows simple and fast.

*Single keystrokes should allow the user to open, hide, tile, resize or close windows, and move around the visual hierarchy of the interface; keystrokes for global resizing and tiling will speed up*

*window alterations and avoid wasting time for the user clicking, dragging, etc.*

Use consistent syntax across the interface.

*Matching syntax to the musician user community; avoid needing upper case letters in responses to system prompts, interpret between upper and lower case in filenames to speed up selection from a user input string; – this will facilitate the musician who has non-expert keyboard input skills. Employ universal unambiguous commands – cut, copy, splice, pan, mix; remain consistent in the use of these terms across the interface at all levels.*

Maintain a clear structure throughout the micro and macro levels of the system.

*To aid user understanding and clarity, implement a concise structure so that the user is aware at all times where he is in the system, what he must do next to further task completion, or how he would retrace his steps. A musician using a system which allows easy progress from parameter handling at the micro-level to editing details of a soundfile at the macro level will be working smoothly with no hindrance to the creative process.*

Afford the user complete/complex error handling.

*Design a system which provides clear warnings, easily revoked errors, error checking systems, user leading by way of suggestions*

*to error inputs etc. This leads to a more friendly environment, in which the user will have more confidence, and feel willing to experiment. A system with a high level of recursive activity will give the musician the confidence to try certain steps, without he worry of not being able to undo unwanted actions easily.*

Employ user models for language syntax and medium (3:49).
*A system which exploits traditional musical syntax and symbols will speed up learning for the user, and result in easier acceptance of the system.*

Keep graphics images clear, uncluttered, and not dense, thereby aiding memory retention (3:47).
*Do not overload the user with too many graphical icons and images; all images should relate to the musical tasks at hand, and not require interpretation before the user can employ them. This will result in undue cognitive load, and will divert the user attention from completion of the creative task at hand.*

Keep interface activity to a minimum after introducing a new operation or tool. Do not supply unnecessary material to the user during input (3:52).
*Distractions on the interface will result in the user easily forgetting a newly employed sequence of keystrokes, or new operation. Avoid redrawing waveforms or moving images unnecessarily which results in distracting the user from the task at hand.*

Provide on-line help, together with on-line tutorials where applicable (3:52).

*On line assistance will enhance user ability, decrease the learning time for the system, and lead the user to be more adventurous. Users learn faster by demonstration, so on-line tutorials will speed up the learning curve for a user new to a system.*

Allow the user to work serially with a minimum amount of simultaneous action from other screen events unrelated to the task in hand (3:44).

*Distracting the user with unnecessary screen updating or movements will interpret the flow of the task steps.*

Provide, where possible, several routes to access single items, as this will result in faster user recall from long term memory, when a variation of cues has been made available (3:45).

*Variation of cues allows the user to categorise and separate information for storage and recall. Provision of screen menus in graphical interfaces, together with access by request on command strings in dialogue boxes, will give the musician user confidence in touring around the system, and will reduce cognitive load as he tries to remember the only route to a screen or operation.*

Provide visual portrayal of signals for analysis, decomposition and manipulation

*Graphical representation where possible will facilitate a familiar environment for the musician, where the presentation maps closely to his mental and expected models. Working as closely as possible to the user's expectations of signal presentation, and not to data presentation which may require translation, will result in the design of a system which is quickly accepted and embraced by the user.*

### 5.2.2 User-Related Guidelines

At all times, define the user primarily as a musician, who is employing computer technology as a means to an end. (3:52)

*A system should fit as naturally as possible within the musician's creative workspace, and should not intrude unduly upon or interfere with that creative process.*

Provide the user with informative feedback at all levels of task completion.

*Users actions should always get a reaction – if reaction will require computation time, provide a progress bar or message to the user. Leaving the user guessing if input has been successful results in frustration and lack of confidence in a system.*

Confine the degree of system leading of the user purely to enhancing user-friendliness and ease of operation.

*System leading should not be so strong as to dictate terms and should avoid imprinting a system style on the users compositional activity. If a system embraces a specific compositional approach the user should be made very aware of this from the outset.*

Define and maintain a system balance between generality and strength.

*Design a system which provides a working environment for new users and more computer literate users. Decide if the system will serve all users across the spectrum, concentrate on general simplistic tools, dwell on idiosyncratic components, or be a combination of these.*

A general release system should encourage user development, allowing access at all levels to accommodate all user communities, and to encourage and lead beginners to experiment with micro level tasks and functions in the system (3:51).

*When a system is designed for general release, it should support beginner users with effective on-line help and tutorials, in order to encourage experimentation, and to allow the user to at least get a reaction in response to an action. At the same time, the system should not be too simplistic as to frustrate expert or intermediate level users, but provide them with an environment which facilitates assistance with tasks together with enhanced completion times.*

Reduce user cognitive load to leave the musician user free to concentrate on the aural output, and not to have to worry unduly about system operation (3:52).

*The user should be able to concentrate on aural output as the most important issue, and not be side-tracked or have the composition process interfered with, by using valuable cognitive activity to operate the system.*

Strike a balance between a system which is a gesture transducer and one which is an information handler.

*As systems which are foremost designed to facilitate creativity, computer music systems should attempt to strike a balance between pure data input on the interface and being a gestural transducer for the user. For example, when a user needs to input the shape for a waveform or envelope, input graphically will be more closely mapped to the user natural task model, than to have to input data representing each point in a table. Where gestures can be translated by the system, the user will be able to work more closely to his mental model.*

### 5.2.3 Task-Related Guidelines

Allow the user to easily undo or reverse an action.

*Allowing the user the opportunity to change his mind or retrieve a previous state will enhance users' confidence in the system – if an*

*action is irreversible, the user should always be warned before execution. Users given the chance to reverse an operation will experiment more with the system, and will feel more confident in use. For a musician, the opportunity to experiment with sound files and musical data is invaluable; a computer music system should at all times encourage experimentation by embracing a task path with constant recursive activity built in.*

Provide flexible internal libraries, and allow for extendibility of libraries and algorithms.

*Within systems which allow the user to perform digital signal processing, the system should allow as much flexibility as possible. For the beginner user of the system, closed algorithms mapped in public access libraries, and supported by on-line help and descriptions will enhance understanding and acceptance of the interface. For the expert user at the opposite end of the user scale, open-ended programming facilities, allowing for extensions in the processing abilities in a user defined fashion will increase experimentation and user manipulation of the system. For the intermediate user, a balance between accepting what is available and extending this can be incorporated in terms of the linking of pre-programmed modules etc, on a level just short of intensive programming.*

Design the interface to reflect the task structure derived from user task analysis.

*Structure the interface to reflect the way the user has defined the stages for task completion. The computer music system should be modelled on conventional task steps as far as possible, or when it has to extend these, the steps should lead in a natural sequence and be comprehensive; reflect user expectations and conventions.*

Employ efficient data hierarchy.

*Each new level should interpret data streams from previous levels, decreasing the users' mental load. The system should not require the user to recall details from previous screen levels in a task path, or require him, for example, to recall sample rates or channel details from previous screens.*

Notwithstanding algorithm compilation, keep the time necessary for a user to achieve a task to a minimum.

*Concerning issues including the encoding of the stages of a task, the musician should be able to complete a specified task within a reasonable time frame, mapped to the time taken to complete the same or an equivalent task outside using the system. At all times the aural output is the important factor, and if a user has to spend an unduly long period of time executing a task to hear the intended output, the system will become unsatisfactory. If time frames are extensive for task completion, then concentration and interest will lapse, resulting in frustration with the interface.*

Tasks should be broken down into a set of short serial actions. (3:52)

*Users short term memory will not be overloaded if the user can serialise tasks into a set of small related actions. Learning curve will also be sped up, if tasks lead intuitively from one to another, based on user models. An action reaction should not increase the mental load by added decision making causing fatigue and frustration. Also tasks should be constructed in short easily achievable steps in order to avoid diverting musician concentration and attention from the creative intended goal.*

## 5.3 Computer Music System Synopsis Grid

As a tool to facilitate referencing the main analytical details of computer music systems, and as a method to allow for unbiased simplistic cross comparison of the systems introduced in the ensuing chapters of this thesis, the following Computer Music System Synopsis Grid is introduced, as seen in Figure 5.1 below. The grid is designed to present details relating to computer music systems ranging from hardware specific facts to evaluative measurements of performance and analytical observations.

| 3 | | | 0 | | | 3 | |
|---|---|---|---|---|---|---|---|
| command line interface | | | | | | | graphical user interface |
| data handler | | | | | | | gestural transducer |
| signal processor | | | | | | | Midi event handler |
| micro-level operator | | | | | | | macro-level operator |
| heavy cognitive load | | | | | | | light cognitive load |
| specific system | | | | | | | general system |
| specific hierarchy | | | | | | | open task design |
| low recursive activity | | | | | | | high recursive activity |
| expert user | | | | | | | beginner user |
| system leading | | | | | | | user modelled |
| 3 | | | 0 | | | 3 | |

Figure 5.1 Computer Music System Synopsis Grid

Using this grid, systems are summarised in terms of categories including:

- modes of operation
- signal creation and modification techniques available
- compositional hierarchy embedded in the system
- user mental load involved in the system
- details referring to the discourse between the user and the system interface.

The grid reads from the central position outwards. A score of 3 left would indicate that a system completely encompasses the left bias of the category in question, while a score of 2 or 1 left would indicate that the system in question fits into that part of the

category, but to a lesser degree. At the same time, a system may score 1 right in the same category, meaning that it also encompasses to a certain degree some of the right bias of the same category. A mixture of a left and a right score indicates that the system is in some way a balance of both parts of the category. For example, a system may employ a graphical interface, and also employ equally command line input. This system would score perhaps 2 left and 2 right in the interface method category at the head of the synopsis grid.

The categories which are embedded in the grid are as follows:

- Command line Interface– Graphical user Interface

This category will display the type or types of interface employed by a system.

- Data handler – gestural transducer

This category will identify systems which on one side of the scale involve the user with numerical data to affect an action or a reaction with the system. On the other end are systems which interpret users gestures in some form to extract input to the processor at the interface level. Scores on either side of the central line indicate that a system combines methods of input, to allow the user to work directly with data input and also to transduce data input from a user's gesture in some way.

- Signal processor – Midi event handler

On a very factual level, this category indicates whether a system is designed to handle signal processing, or to manipulate event-based Midi data. A score towards the central position on the signal processing side of the category indicates that the system supports digital processing capabilities, but only to a small degree.

- Micro-level operation – macro-level operation

Within this category, systems are evaluated with regard to the level at which the user operates in the compositional process. A system evaluated to operate at the micro level will allow the user to manipulate data relating to the core parameters within the internal makeup of a sonic event. Systems which operate at the macro level will afford the user the possibilities of score manipulation, or gestural control on full sound files, such as panning control, or mixing and splicing.

- Heavy cognitive load – light cognitive load

Within this category, an analytical evaluation of the amount of user concentration to achieve a task is revealed. The cognitive load in question relates to the degree of attention the user must divert from the musical task in order to operate the system tools and input to the interface in order to complete that task.

- specific system – general system

Relating to the nature of the tasks which a system is designed to accomplish, or to assist the user with, this category reveals how particular systems are deliberately confined to very specific focused tasks, and how others are modular in design, incorporating several sets of varied tasks within single systems.

- specific hierarchy – open task design

Within this category systems which allow a flexible approach with regard to the compositional methods employed by the user will be identified as open ended in their task design. Systems which have a specific hierarchical methodology embedded in them impose a particular compositional methodology upon the user.

- low recursive activity – high recursive activity

Any system which facilitates fast and simplistic error correction, or which allows the user to continually retrace his steps within a task path as he experiments with ideas, will be categorised as having a high level of recursive activity. Systems which are not flexible enough to allow the user to change his mind mid-task, or which do not allow the user to easily and quickly correct an error are identified as having a low level of recursive activity.

- expert user – beginner user

A system categorised as purely beginner user level will be one which is superficial in its task encoding, which allows the user to complete quite simple compositional tasks, and affords no degree of development to higher task levels. At the other end of the scale, an expert user system will be designed to facilitate high level intensive tasks which require refined user input. An expert user system will have little or no entry level tasks which might encourage experimentation on behalf of a beginner user. A system falling equally between both sides of this category will have entry level operations which allow beginners to get reactions to input actions, while simultaneously having high level tasks encoded, to facilitate expert users to work in an environment which is not overly simplistic.

- system leading – user modelled

A system which is user modelled is defined as one in which the user can map his understanding of the steps involved in completion easily to the interface and syntax of the system. There is little or no need for the user to translate his mental task model in order to use the system to complete that task. Systems which are categorised as being leading are those which require the user to adopt the system method of task completion which may be foreign to the users natural mental model.

This system synopsis grid will be applied to the computer music systems introduced later in this thesis.

# Chapter 6 COMPOSER'S DESKTOP PROJECT (CDP)

## Non-Real-Time Composition And Sampling Environment

### 6.1 Introduction to the System.

Developed by a group of composers based in York, the Composer's Desktop Project affords the composer (CDP documentation, 1989)

> "...a completely digital synthesis and audio processing package concerned with the creation, processing, and organisation of digital sound samples...."

As the designers of the system (CDP Advert, 1989: pg. 1) write:

> "Begun in the spring of 1987 the Composer's Desktop Project brought together composers, computer scientists, and hardware designers to develop the idea of a desktop based system based on the Atari ST microcomputer. The result is the **CDP Workstation,** with its own Soundstreamer interface and soundfiling system...The CDP provides a comprehensive computer music workstation for the composer, with:
>
> - *an emphasis on composition and state of the art tools developed in a co-operative research environment*
> - *the advantage of professional digital sound quality, which does not degrade in successive reworkings*
> - *access to major computer music programs such as CSOUND and CMUSIC, custom-designed editing, mixing and signal processing facilities, and an ever increasing range of high level compositional tools"*

As seen in Figure 6.1 below, the Composer's Desktop Project is a system with purpose built units to allow the user to interface his/her computer with the digital recording medium[1].

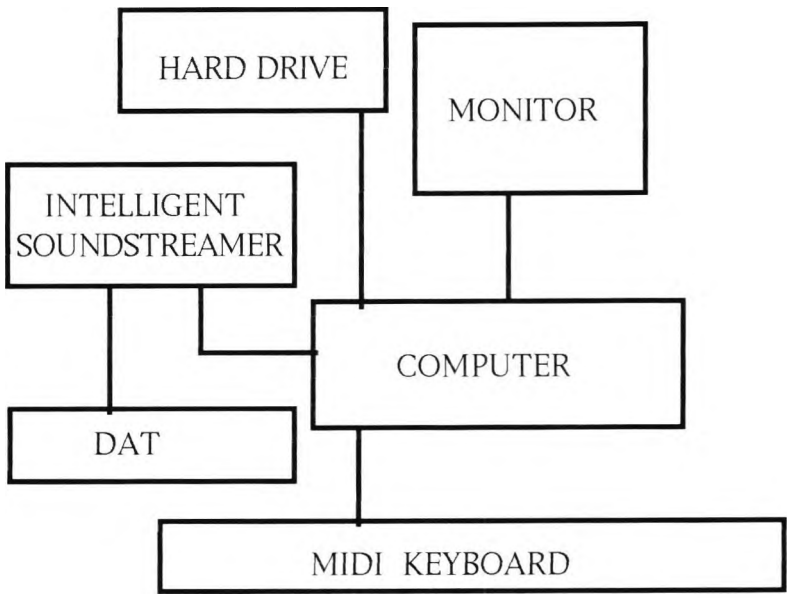



Figure 6.1 Physical Layout of CDP Workstation

The Soundstreamer unit, specially designed as part of the CDP, and the core hardware element of the system, interfaces the computer and the digital recorders. Together, this hardware configuration makes it possible to input sound samples to the CDP programs and to create stored libraries of the sound samples created within and recorded into the system.

[1] At the time of writing, the Composer's Desktop system is available on the Atari Falcon, and all related Atari computers. It has also been made available on the PC range of personal computers, with plans to port the system from the PC version to the Macintosh being investigated. With the Macintosh port, there are plans to support a complete graphics front end for the Granular package of programs.

The sound creation, manipulation and storage options made available to the composer in the system are listed in Figure 6.2. Within a non-real time system, a comprehensive set of sound generation and manipulation procedures are supported.

| | |
|---|---|
| Adsyn | Graphical additive synthesis generator |
| Cdpdesk | Graphical play/record and soundfile control |
| Cmt Midi Toolkit | Command line MIDI environment |
| CSOUNDutils | Full utility library for CSOUND |
| CSOUND | Sample generation and manipulation |
| Disp | Graphical presentation of soundfiles. |
| Dskutils | Comprehensive set of hard disk utilities. |
| GROUCHO | Command line editing package. |
| Midore | MIDI to CSOUND format conversion. |
| Midigrid | Graphical MIDI composition package. |
| Phase Vocoder | Time and Pitch Manipulation. |
| Scorp | CSOUND score editing package. |
| Sndutils | Full range of sound utility programs. |
| Utils | Comprehensive set of basic utilities. |

Figure 6.2 CDP System Program Listing

In this discussion of the sound creation, manipulation and modification programs available on the Composer's Desktop Project computer system, we will concentrate on the CSOUND composition package, the GROUCHO editing suite of programs and the Phase Vocoder signal modification environment. These packages are the ones used extensively by the author, and will be the main focus of the evaluation of the CDP system from the point of view of a composer user.

## 6.2 System Details and Mode of Operation

### 6.2.1 CSOUND on the CDP

Originally developed at the Massachusetts Institute of Technology by Barry Vercoe from work originated at Bell Laboratories by Max Matthews, the CSOUND package was ported to the CDP by Bill Payton. The composer creates music by a two-fold operation, first by structuring *instruments* within an *orchestra*, and subsequently detailing *events* within a *score*.

There are many methods of digital sound synthesis and the CSOUND package supports a set of seven synthesis models, which are:

- Direct synthesis
- Additive synthesis
- Subtractive synthesis
- Sampling synthesis
- Granular synthesis
- Non-linear synthesis
- FOF synthesis

Each of these synthesis methods, as detailed in Appendix A for reference, are implemented in CSOUND by linking together a series of unit generators and signal modifiers in a CSOUND *instrument*, the CSOUND language equivalent of a signal patch diagram, and stored as an *orchestra file*. This *orchestra* is then matched with a CSOUND *score*, a listing of parameter values over time for each of the generators and modifiers within the *instrument*. This coupling of an orchestra file and score listing is then sent to the CSOUND

compiler, via a single command line string. The complete range of generators and modifiers together with a brief description and effect of each is listed for reference purposes in Appendix B.

In order to detail the nature of the interface which the user encounters when working with CSOUND, it is necessary to look at exactly how a user would go about inputting an orchestra and score to the system for compilation. A typical example set of instrument listings as an *orchestra file* is detailed in Figure 6.3 followed by a complimentary score listing in Figure 6.4. Comments for explanatory value have been added throughout the listings and are indicated as comments and not part of the compiler text by semicolons at the start of the comment.

```
; Orchestra File for Instruments examples 1, 2 & 3

sr = 22050          ; sample rate
kr = 441            ; control rate
ksmps = 50          ;
nchnls = 1          ; number of channels, mono or stereo

; p3 = overall duration of event          k = control
; p4 = amplitude                          a = audio
; p5 = pitch
; p6 = rise time for line shaping
; p7 = decay time for line shaping
; p8 = multiplier note value
; p9 = multiplier note value
; p10 = multiplier note value

Instrument 1

  k1  linen  p4, p6, p3, p7     ; white noise produced by combination
  a1  rand  k1                  ; of random noise generator on control
  k2 = cpspch(p5)               ;converts pitch to hertz
  k3 = cpspch(p5)*p9            ;p9, p10 & p11 are acting as multipliers
  k4 = cpspch(p5)*p10           ; on the values assigned to p5
  k5 = cpspch(p5)*p11
```

```
a2  reson  a1, k2, k2*0.01, p8      ; signal is applied in turn to four
a3  reson  a1, k3, k3*0.01, p8      ; second order recursive band pass
a4  reson  a1, k4, k4*0.01, p8      ; filters
a5  reson  a1, k5, k5*0.01, p8      ;
     out a2+a3+a4+a5                ; signals combined and output

endin                               ; instrument closed
```

Instrument 2

```
k1    linen    p4, p6, p3, p7
asig  oscil    k1, cpspch(p5), 1    ; uses function table 1
      out asig                      ; signal output

endin                               ; instrument closed
```

Instrument 3

```
; p8 and p9 = values by which the centre pitch, p5, is modified p9
;times per second by random displacement up to p8 no. of octaves.
; p10 = no. of harmonics which are to be added to the signal


k1   linen  p4, p6, p3, p7      ; white noise produced by combination
k2    randi   p8, p9            ; of random noise generator on control
k2 = int(k2)                    ; conv of cntl to int for buzz
a1    buzz  k1, cpspch(p5), (k2+p10), 2          ; uses function table 2
      out a1                    ; signal output

endin                           ; instrument closed
```

Figure 6.3 CSOUND Orchestra Listing

```
; Score File for Examples 1, 2 & 3.

f1  0  256  9  1  3  0  5  .32  4  .12  .56  10  .65  9  .45  8  .34
7  .23  6  .12  10  5  10  .34

; function table reads as follows: function table no, time oscil is to
start reading table, no. of sample points in table, GEN no. (type of
wave to be looked up and read).

f2  0  8193  10  1
; printouts of the function tables as they are constructed during
; compilation follow in Figure 6.5 for reference.

  t  0 60 3024035       60
 i1     0 10    20000  8.00  7  2  1  1.189    1.498    1.888
 i1    10 10    20000  8.02  2  8  1  1.189    1.498    1.888
 i1    20 10    6000   8.06  4  4  1  1.189    1.498    1.888

 i2     3      7    3500    8.04  3.5   0.5        1
 i2    10     12    3500    8.05  4     2
 i2    22     14    3500    8.09  6     6

 i3    20     10       10000  5.04  4.0   2.0   0.5  10  5
 i3    30      2       10000  6.04  0.5   0.5   5.0  10  5
 i3    32      2       10000  4.01  0.5   5.0   5.0  10  5
 i3    34      1       10000  4.05  0.5   0.5   5.0  10  5
 i3    35      1       10000  4.06  0.5   0.5   5.0  10  5
 i3    36     0.5      10000  4.07  0.25  0.25  5.0  10  5
 i3    36.5   0.5      10000  4.08  0.5   0.25  5.0  10  5
  e
```

Figure 6.4 CSOUND Score Listing

Once the combination of an orchestra file and a score file have been passed to the system for compilation, the user gets constant report feedback as compilation progresses. This report information can be saved as a text file if requested on the command string at the time of compilation. Very typically, this information is used to source the areas of distortion and other problems which are found in the resulting output sound file. Once compilation has started, the first information fed back to the screen and the report file is the

construction of any function tables which have been called into the instrument listing. The function tables for the instruments listed in Figure 6.3 are constructed as in Figure 6.5 below:

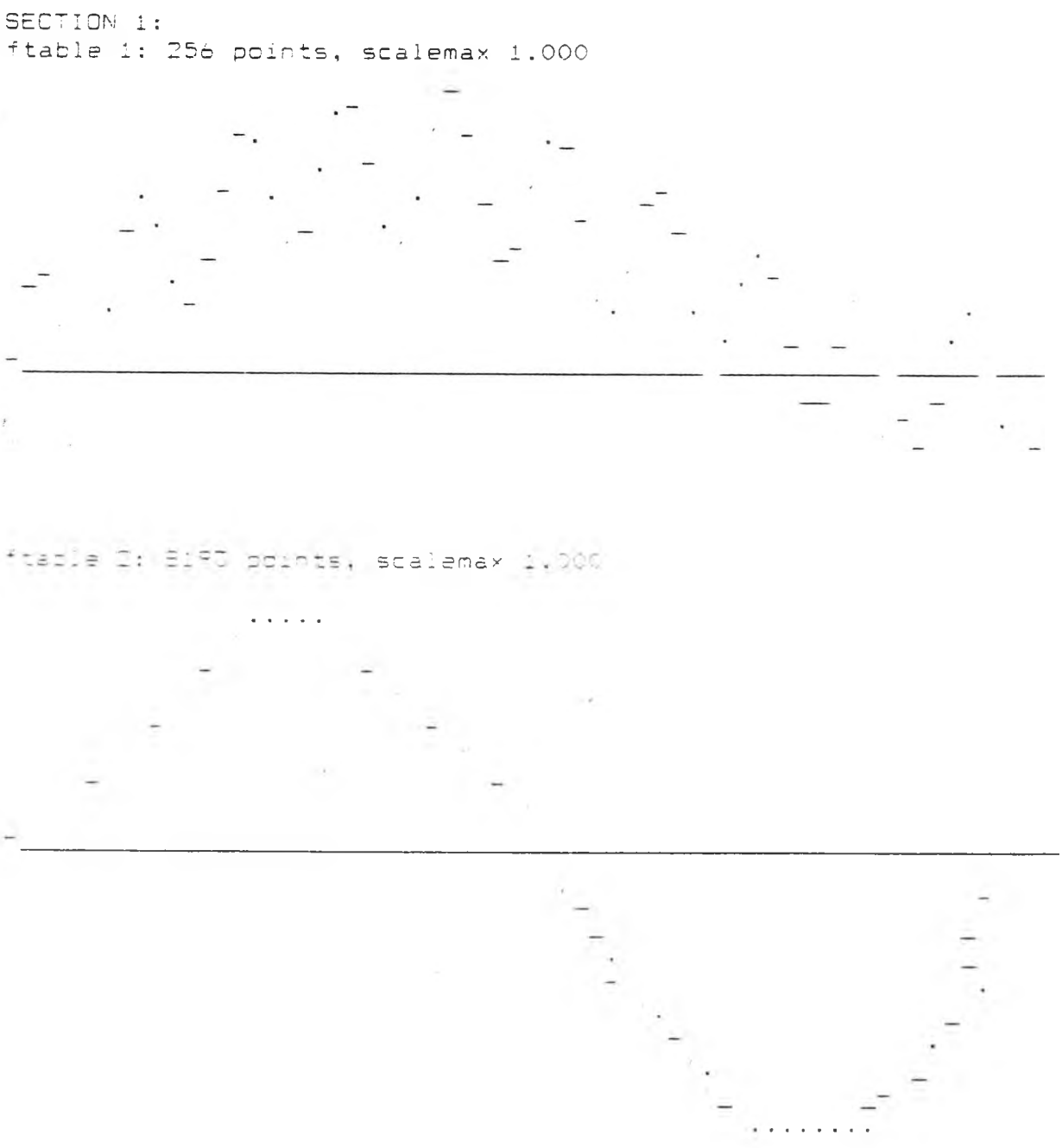



Figure 6.5 Function Tables for Examples 1, 2 & 3.

Within this printout, the user can refer to the GEN listing of subroutines available in CSOUND, a set of composite waveforms which the table look-up oscillator is directed to sample through. Both of the tables in this example, GEN 9 and GEN 10, are generate composite waveforms of simple sinusoids, and are the most used of the GEN set of routines. GEN 10, the most accessible routine to the novice user of a CSOUND orchestra, generates a very simple sinusoid in which the partials must be harmonic and in phase. From the printout, it is clear that GEN 9 does not require this conformity.

In order to find areas of error and problems in a score and orchestra to be able to redo a compilation due for example to distortion in a file, the report file generated during compilation can be very useful. The format of the report can be seen in Figure 6.6 below, which contains only a small portion of a typical report file created during a score and orchestra pass to the CSOUND compiler.

Report File For Sound File VB37

N: 256 M: 256 L: 256 D: 32 I: 32 F: 86 R:22050.0 P:1.00 T:1.00

| 1 | band | max amp | avg amp | avg frq |
|---|---|---|---|---|
| 1 | -43 - 43 | 0.01185 | 0.00657 | -0.2 |
| 3 | 43 - 129 | 0.01242 | 0.00283 | 0.6 |
| 5 | 129 - 215 | 0.01678 | 0.00028 | 206.2 |
| 7 | 215 - 301 | 0.02095 | 0.00044 | 420.3 |
| 9 | 301 - 387 | 0.04334 | 0.01884 | 441.6 |
| 11 | 387 - 473 | 0.08949 | 0.05757 | 441.9 |
| 13 | 473 - 559 | 0.04890 | 0.03147 | 442.3 |
| 15 | 559 - 645 | 0.01183 | 0.00088 | 443.0 |
| 17 | 645 - 731 | 0.00482 | 0.00028 | 699.9 |
| 19 | 731 - 817 | 0.00772 | 0.00206 | 872.5 |
| 21 | 817 - 903 | 0.01836 | 0.00826 | 883.9 |

**Figure 6.6 Report File for a CSOUND Compilation Pass**

As the set of figures above shows, the task of creating an orchestra and score in CSOUND and subsequently creating a successful soundfile by passing the score and orchestra to the compiler is both a complex and intensive procedure, described in the CDP promotional literature (CDP, 1989, pg, 7) as:

> "....an art of considerable subtlety, and can involve sophisticated parameters such as the control of the exact placement or movement of sound within acoustic space."

The interface with which the user has to interact provides complex control of several parameters at the micro level of sonic design.

### 6.2.2 GROUCHO on the CDP

The GROUCHO sample editing package, written exclusively for the CDP, offers a very comprehensive set of sample editing and manipulation features. The package subdivides into four categories:

- Editing and Mixing Programs
- Filtering programs
- Sample Manipulation Programs
- Utilities Programs

The programs available in each category are listed for reference purposes in Appendix C, together with the command line arguments for the operation of each program. The GROUCHO package offers the user a very comprehensive editing and sound manipulation suite, within a command line control environment.

As a user interface, the GROUCHO package presents itself to the user in graphical form on the CDP Desktop, and can also be operated from the command line, by passing single string arguments to the processor, or by setting up batch files, series of command line arguments batched together to facilitate several processes to be compiled over a period of time. A typical command line within GROUCHO, detailed below in Figure 6.7, involves the user stringing together a set of values which are mapped to parameter values expected by the program in question.

**LOOP [-f] i/f o/f o/flngth lpstart lplngth [lpstep] [sfield]**
Note: any parameter in square brackets is optional; parameters relate to infile name, outfile name, outfile length, loop start point, loop length, and further optional parameters including a step value through the file and a searchfield value.

**Figure 6.7 GROUCHO example Command Line.**

In this example where the user wants to create a loop he must specify on the command line the input file from which the loop is to be created, the resulting output file to be created by this looping procedure, the length of the new file, the start time of the loop within the input file, the overall length of the loop, a value, if desired, by which the processor will step through the input file

before starting the second and following loops, and also if desired a searchfield value by which the processor will search for a specified value within the input file before beginning the loop. A further set of values may be supplied as flags to the program if the user wishes to further control other parameters during the processor pass on the soundfile. Flags may include such parameters as sample rate control and channel control variables.

It is clear that the procedures available within GROUCHO are complex in the control that they provide to the user over a soundfile, and the parameter string which must be provided for each program is detailed and complex.

### 6.2.3 Phase Vocoder on the CDP

The Phase Vocoder package is a powerful sample transformation tool capable of controlling time-varying sound spectra. It operates by first analysing an input soundfile, using the Fast Fourier Transform, into a large number of frequency bands, determining the sound level in each band. This process is repeated for each time-window, in the sample. The actual duration of the time-window, the amount of overlap in the windows, and also the number of frequency bands are all controllable by the user on the analysis pass command line.

In order to implement the Phase Vocoder sound transformation utilities on any input samples, it is necessary to first

submit the sound sample to an analysis program. The Phase Vocoder program, when used with the -A flag, implements an analysis pass on the input soundfile. The command line which must be used is seen below in Figure 6.8:

sfpvoc -A -V <infile> <outfile>

notes: flags used in the sfpvoc program instruct the computer to carry out either an analysis of the input file (-A), and to create a report file of the analysis information for the user to access (-V).

Figure 6.8 Phase Vocoder Analysis Command Line

The resulting analysis file, which is stored on the sound file directory in floating point values, is held at a sample rate as specified on the command line for the analysis pass. This analysis file is then used by consequent manipulation programs to alter the input sound sample. While an analysis is being created, the user can ask for a report to be written, using the -V option. This report file will automatically be named pvoc.s. It will provide channel information and frequency and amplitude details within each phase vocoder channel.

When an analysis pass has been made on a sample file, the user has two ways of manipulating the data which now represents the input sound file. One is to feed this data file to a resynthesis option, while simultaneously asking for either a stretch of the time or transposition of the pitch of the sample. A typical example of one of these command lines is:

sfpvoc -T5 -S fred.dat frednew

In this instance, the data file version of the original sound file, *fred* will be resynthesised, and time stretched five times its original length. The use of the -T flag is for time stretching, and the -P option will alter the pitch data. In this case, the resulting data is written to a new soundfile, *frednew*.

The other method which is available to the user at the post-analysis stage is to pass the data file to another program which will subsequently create a new data file which can then be resynthesised by passing to the -S flag. There is currently a comprehensive set of these programs available which support the Phase Vocoder package. These are listed for reference, together with a brief description of each, in Appendix D. It is important to remember that these programs do not manipulate *sound samples* but rather the analytical representations of those samples, and are basically number manipulation programs, operating on mathematical algorithms. As such, they compile their numeric results very quickly, and the time expensive part of the process is when these numbers are to be reconstructed as sound samples, i.e. when the file is sent through a -S resynthesis pass in the Phase Vocoder foundation program.

The Phase Vocoder suite within the CDP provides very powerful signal manipulation tools for the user in a command line environment which remains consistent in syntax and style across the package.

## 6.3 System Synopsis, Analysis and Evaluation.

As the first computer music system and indeed the first computer environment to be used by the author the CDP presented itself as a system which obviously held a lot of promise, and which would allow the user to create and manipulate sound in a completely new fashion, where no number of reworkings of a soundfile would normally reduce its quality. However the system interfacing quickly proved very difficult for the beginner computer user to come to terms with. Even with a firm basic grounding in digital signal processing theory and also a good grasp of the principles of designing an instrumental patch diagram, an extensive amount of thought and planning to design a successful CSOUND orchestra and score was required. The user needs to think about musical processes in a somewhat abstract manner, and only after several unsuccessful passes to the compiler and constant rewriting of incorrect syntax could even the most simple of sound files be produced.

In the course of this evaluation and analysis, several guidelines which are introduced in Chapter 5 of this research will be introduced as start points for discussions of various elements of the CDP compositional environment. In this manner, both the positive and the negative elements of the interfacing between the user and the CDP will be constructively revealed. Prior to these analytical points being discussed, a system synopsis grid is drawn up, in order to collate the system details, strengths and weaknesses from a user interface point of view.

### 6.3.1 CDP System Synopsis Grid.

The CDP system is summarised in Figure 6.10, using the Computer Music System Synopsis Grid introduced in Chapter 5. This grid will facilitate faster reference to system details and specifics in the ensuing analysis section.

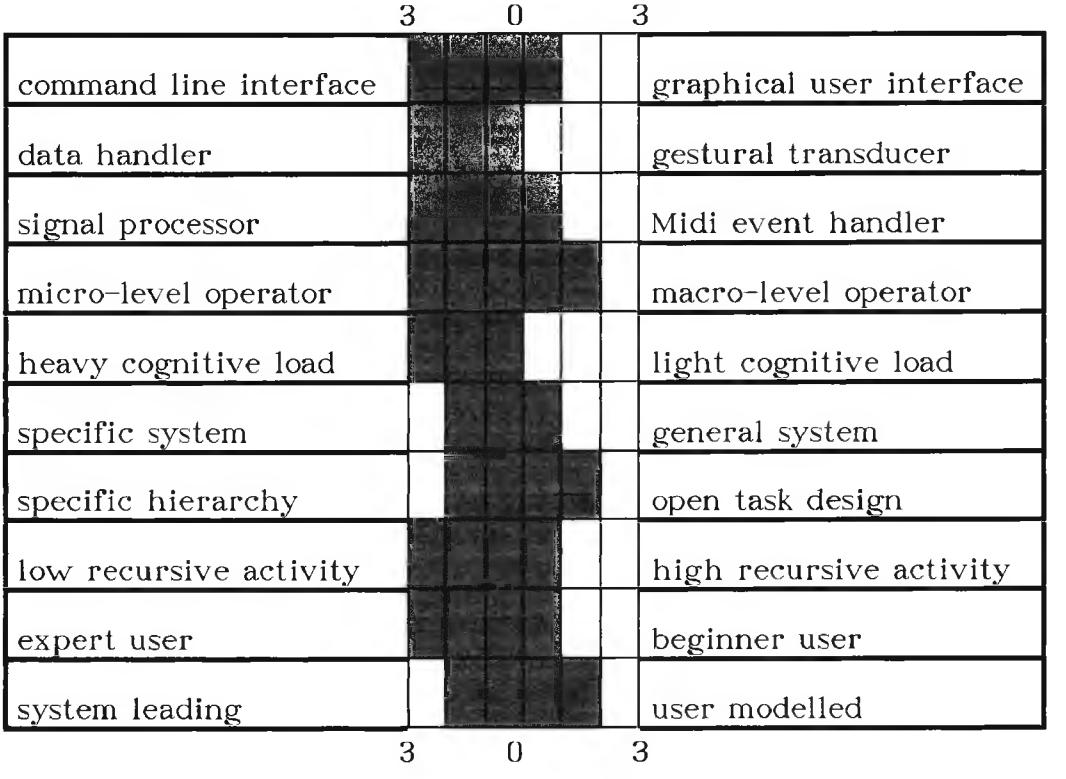



Figure 6.9 CDP System Synopsis Grid.

The system is evaluated as embracing almost completely a command line approach, with data handling being the exclusive method of user input. As mainly a digital signal processing environment, with a small Midi capability available in a CSOUND instrument, The CDP operates almost right across the micro- to

macro levels of task types. Maintaining a balance between specific and general tasks within the digital processing domain, the CDP does not lead the user on one specific path for compositional task hierarchy. As a data intensive system with regard to input syntax, the CDP scores very badly with regard to facilities for error correction or experimentation. Although beginners with the CDP have an uphill climb to get to terms with the system, they are accommodated to an extent with a set of CSOUND demonstration libraries available. As a modular system, embracing several packages for users to access, the system does not lead the user completely away from conventional DSP models, but affords the user an open working environment within DSP programs.

### 6.3.2 Analysis and Evaluation.

Guidelines derived from Chapter 5 are introduced in italics as a start point for discussion. Evaluations and observations regarding the CDP system are discussed immediately following each guideline.

*"...employ user models for language syntax and medium..."*

Within the CSOUND environment, the method of input for scores and orchestras is not naturally mapped to the user models or designed in a fashion which will enhance user learning and acceptability of the system. As the orchestra listings are constantly being explained in terms of signal patch diagrams, it seems much more beneficial to design some form of input for the user in this diagrammatic fashion, allowing the processor to do the work of

reducing the patch to input for the compiler. This style of signal patch is much more representative of the natural users' model, and would increase the learning curve for the system as the front end for input of data would be more accessible to the user. The style of orchestral listing as seen in Figure 6.3 of this chapter is one which requires a deal of interpretation even for a CSOUND expert user. A system which requires the user to learn an entire syntax in order to facilitate simple input is not successfully modelled on any user expectations, and will not be easily accepted and employed by a user population.

*"...on-line help should be provided in order to improve the user interfacing with the environment..."*

One of the most frustrating elements of the CSOUND environment proved to be the absence of some form of syntax checker, in order to be able to check the contents of sometimes complex orchestral and score listings before actually attempting to compile them. As the beginner user struggles to come to terms with complex syntax and language, and also attempts to assign parameters on a control line to match orchestral signal generator and signal modifiers specification, some form of help facility or pre-compilation checker would be very useful, and cut down on user frustration with the system.

*"...provide the user with informative feedback at all levels of task completion..."*

As we have seen in Figure 6.6 (Report File for a CSOUND Compilation Pass), the method used to report details at the task completion stage of CSOUND compilation is far from easily understood. This is also the case with Phase Vocoder analysis report files, which employ a similiar format. It is not useful to have to begin to virtually decode report files before any of the information held within can be used beneficially.

*"...the virtual software created interface should shield the user from the computations happening at lower levels in the system..."*

With the exception of the GROUCHO graphical interface, the user of the CDP is at all times too close to the processing occurring within the system. On the command line, constant output of computational data during compilation passes makes the environment less than approachable, and the user must constantly remind himself that he is involved in musically related tasks, as the interface rarely portrays this fact.

*"...Simplicity should be aimed for..."*

Within several working areas of the CDP, the user is required to input to the system using complex command line strings, as we have seen in the GROUCHO input example earlier, or to construct intricate orchestral listing and corresponding score listings, with much attention being required to successfully input parameters to the corresponding generators and modifiers, or GROUCHO or Phase Vocoder programs. The syntax used throughout the system places a

high cognitive load on the user, and is somewhat abstract from the nature of the intended task.

*"...use consistent syntax across the interface..."*

Within the CSOUND package, once the user has persisted and mastered the art of designing a syntactically correct orchestra and score, the knowledge gained can be used across all subsequent CSOUND operations, as the syntax is held consistently for creating an instrument, passing parameters to control and audio lines, and for accumulating outputs and ending instrumental listings. Also within the GROUCHO and the Phase Vocoder packages, the command line syntax is consistent, for example, flags across the various programs within each package denoting the same parameters.

*:...Maintain a clear structure throughout the micro and macro levels of the system..."*

The CDP environment offers the composer the choice to operate on both the micro and the macro levels of composition and sound manipulation, but only when working across the series of program sets available. When the user operates within a particular suite of programs on the system, the level at which the user is controlling the sound files and/or single sonic events is always apparent. Although the interface is less than 'user attractive', the user is aware at all times where in the task hierarchy his particular sub-task fits.

Operating on the micro level of single event specification in CSOUND, the system does not support a macro level for the composer to operate at; as a system with its main strength lying in the fact that minute sonic events can be designed in detail by the user in a very specific fashion, the CSOUND instrument environment operates on the microscopic levels of sonic control at all times.

When the user begins to employ the GROUCHO program suite, he moves from the micro level of individual sound creation that is the CSOUND package to the macro level of editing and manipulation of complete or partial sound files, working with for example cut, paste, mixing and splicing techniques.

With the Phase Vocoder signal modification programs, the user can operate at the micro-level of signal manipulation, choosing to modify individual events or entire sound files in single processor passes.

*"...afford the user complete/complex error handling..."*

As a user interface, the CDP system is less than forgiving of user errors - any error on the users behalf on the command line usually is time intensive to correct, as the error will only show up once compilation has started in the case of GROUCHO, when the error message will be returned to the screen as the edit fails. With both CSOUND and the Phase Vocoder, any syntactical errors will only become apparent also when processing fails, and any errors relating to samples out of range, or perhaps relating to mistakenly

overwriting a file will not be correctable until the entire processing has been completed, which can be an extensive amount of time wasted for the user. In the case of all three software packages, any artistic mistakes which are detected in the output file of compilation can only be corrected by running an entirely fresh compilation, again a very time intensive exercise. The user has to develop a very self-critical eye for input strings and for the desired nature of the output file before submitting anything to the compiler.

*"...Reduce user cognitive load, to leave the musician user free to concentrate on the aural output, without cause for concern regarding system operation..."*

As the user employs the CSOUND environment to produce a soundfile, the system operation is unduly taxing and also takes priority over artistic decisions for too high a percentage of the working time. Much of the users time is dedicated to designing a complex textual file, with a heavy cognitive load concerning syntax, unit generator and modifier parameter values, and other mathematical functions which must be related and made to match the required input locations. The user is too detached from the aural output at all times in the system operation, and often what has been an intensive period of input time results in disappointing output, which must be recompiled completely to be corrected and made usable.

Most musician users want to operate in real time, with as close an affinity to the musical output as possible, and also with

as musically related a syntax as possible. The CDP environment requires the user to operate in an abstract environment, and to force himself continually to remember the musical aims and intentions in the face of daunting textual and mathematical input languages.

*"...strike a balance between a gesture transducer and an information handler..."*

As a system which deals with data input for instrumental designs, and also with tabular input for data concerning waveforms, envelopes etc. the CDP interface would be greatly enhanced with the inclusion of a gestural input method for wave shapes, and also with a graphical interface which would allow the user to construct patch diagrams for instrumental design in a combination of graphics and icons from a menu. The method of input described in Figure 6.3 of this chapter is too data intensive for most musicians to want to use.

*"...provide flexible easily controlled algorithms..."*

Although the CDP provides a very impressive set of powerful algorithms for sound creation and manipulation, the user must adhere strictly to the highly defined input strings for each of the programs within the system if a processor pass is to be successful. A degree of flexibility within the system with regard to input syntax and to score listing methods would lead to higher user acceptability and a faster learning curve.

As one of the systems in this thesis which is almost exclusively data controlled, and requires a high level of concentration and mental model translation on behalf of the user, the CDP stands alone as a system designed by composers for composers[2], specifically to be affordable by individuals and not inaccessibly housed in institutions.

[2] It should be noted that the composers involved in the design of the CDP did all have some degree of computer literacy.

# Chapter 7 The UPIC System

## 7.1 Introduction to the System.

The UPIC[1] composing environment presents itself to the user in the form of a dual computer system connected to a large graphics table on which the user draws lines and shapes. These are computed as points in a table and analysed to represent sound envelopes, frequency and time information. In a users' report, the composer Henning Lohner (1987: pg. 19) writes:

> "....the operational simplicity of the graphic digitiser actually excludes the need for technical computer science expertise, in contrast to other systems in which such expertise is a prerequisite, requiring armies of assistants. Rather, the mind and creative impetus of the person using the UPIC become the centre of attention. "

With the aid of an electromagnetic pen, the user has complete control over all the elements of his music from the micro-form at the very heart of the sonic material, to the macro-structure, defining the highest architectural level of the piece. Alain Depres (1989: pg. 141), the director of *Les Ateliers UPIC*, tells us of the origins of the UPIC:

> "..the idea which engendered the UPIC arose in 1953. Xenakis was then composing 'Metastasis', his first piece for orchestra. The bulk of the parts were first graphically written, a huge global mass of glissandi, but for performance the parts had to be transcribed into traditional notation legible for each player. The composer had to work

[1] Unité Polyagogique Informatique Du CEMAMu

very tediously!...To imagine, then, a machine capable of not only realising this graphic interpretation, but also of producing the corresponding sound, this was the composer's dream....."

(Translation: M. Mc Cormack).

The dream became reality some 27 years later, developed by CEMAMu[2] within CNET[3] , and in 1978 Xenakis created the first composition, *Mycenae Alpha,* on the machine. The system evaluated in this paper is the third generation of the original system, operating interactively in real-time[4].

The basic concept of the UPIC lies in the graphic human-machine interface with the basic aims of the user interface being to keep the purely technical side of the synthesis operations in the background. One of the more philosophical aims which the designers of the UPIC strive to fulfil is that expressed by Xenakis, (Lohner, 1987:, pg. 19) when he founded CEMAMu:

".....To change creative habits and methods of creation. Since any person can easily use the UPIC, it will become apparent that every person is essentially creative, having more talent than he/she may have suspected, thereby establishing a new general basis of awareness."

With a machine which affords direct access to sound, the developers strive to allow it to address the problems of contemporary composition in a more simple and direct way, as

[2]CEMAMu, Centre d'Etudes de Mathématique et Automatique Musicales in Paris.

[3] CNET, Centre National d'Etudes des Télécommunications.

[4] The version of the UPIC described in this chapter is the system employed by the author when attending the Second International Workshop at Les Ateliers UPIC in Paris in the Summer of 1990. Currently, the UPIC is available on a personal computer, retaining the original graphics principles by using a graphics tablet for arc input etc..

Xenakis hopes, (Lohner, 1987: pg. 19): ".....direct to the mind." employing what is viewed by the design team as a 'universal method of notation' in a graphical environment.

## 7.2 System Details and Mode of Operation

The system, as seen in Figure 7.1 below is the third generation of the original machine, allowing for composition in a real-time environment.

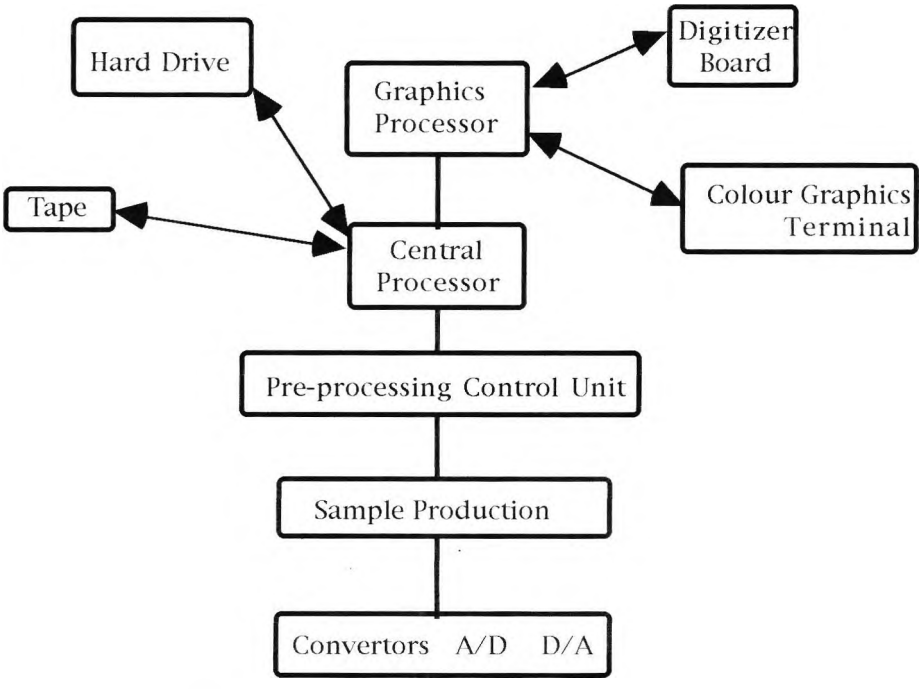



Figure 7.1 UPIC System Layout

The system concept is based on a graphical human-machine interface. The composer draws graphical interpretations of musical events or what systems users refer to as *arcs* [5], on the graphics

[5] Arcs are defined in the language of the UPIC system as the graphic shape drawn on the working board area to represent a sonic event, which comprises a waveform and an envelope associated with it.

board. Figure 7.2 below shows an example of a score page which consists of the arcs created on the drawing board.

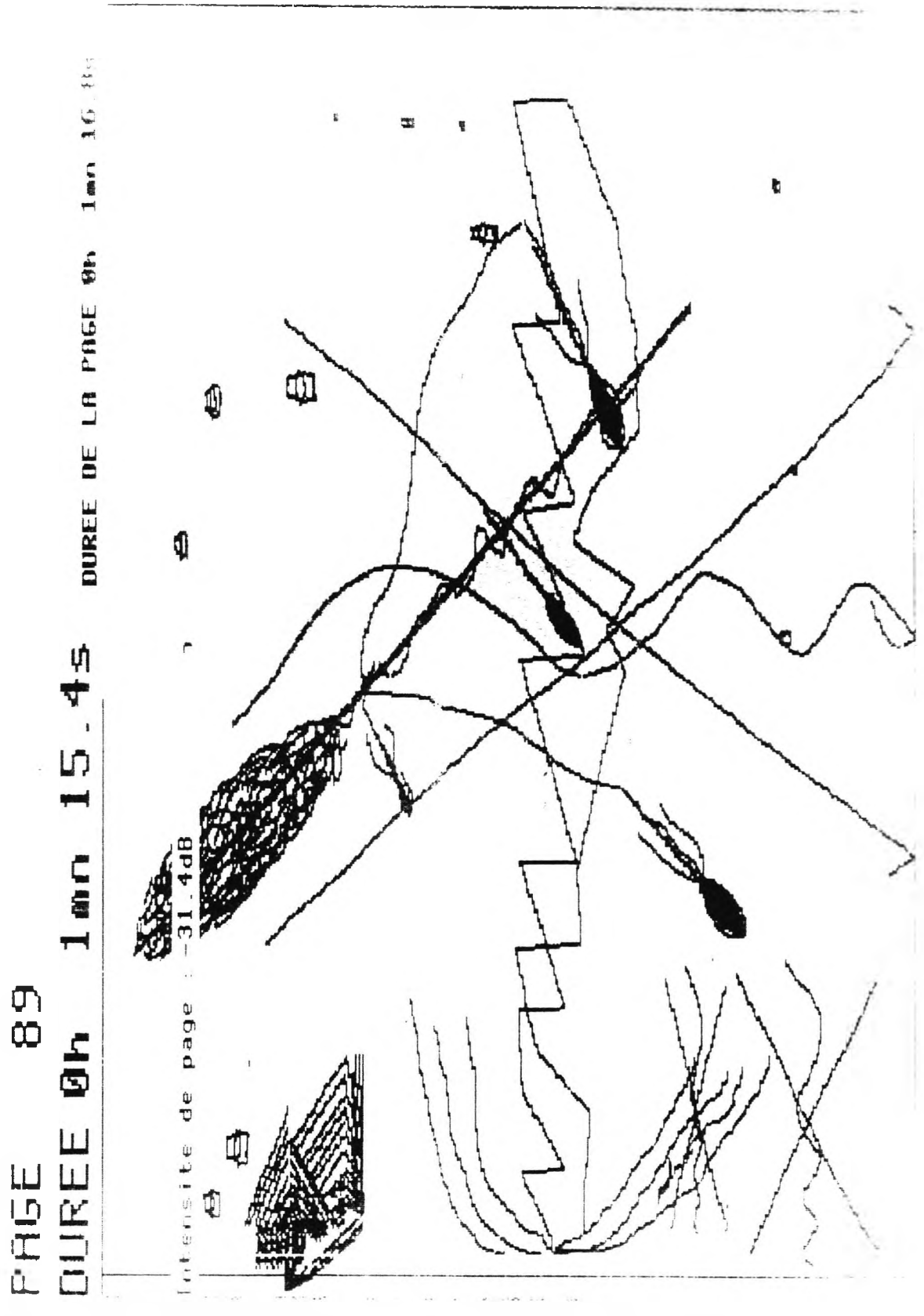
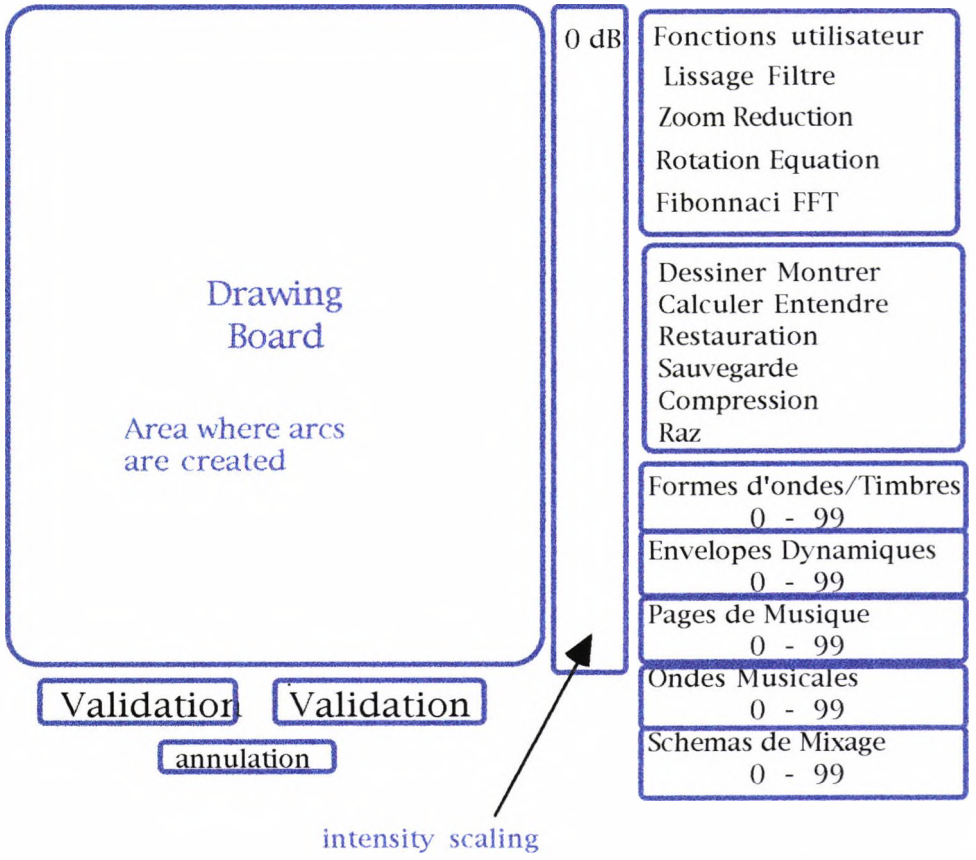



Figure 7.2 Example Score Page consisting of Arcs.

Having already assigned a waveform shape and envelope to the gesture, a dual screen feedback is provided, one in the form of a colour screen detailing graphically all the sonic arcs drawn to date, and the other providing an alpha-numeric representation of all details of the arcs and their associated waveforms and envelopes on the current working page.

The operating system is composed basically of two main parts, the graphics screen and the real-time synthesis unit.[6] The composers' physical working area can be seen in Figure 7.3 below, an area which presents itself to the user like an electrically sensitive architect's drawing board. The physical area the user works on consists of a calibrated rectangle 76cm wide by 66cm high. Underneath the calibrated screen drawing area is a conductive pad containing a tight network of electric fibres. When the electromagnetic pen is in contact with the screen, the x and y co-ordinates of the pen tip position are read to an accuracy of 0.26mm when the pen makes contact with the surface of the drawing area.

[6] The graphic part of the system is based on an INTEL SYS 310 AP including a 80286 processor, a 140 megabyte hard disk, a 6 1/4 inch floppy disk drive and a streaming tape drive. This system is connected to a digitiser and a graphic screen through a bus extension interface.




script in blue does NOT appear on the working board as the user sees it - all other script is as seen by the user during operation of the UPIC

Figure 7.3 UPIC Graphics Board

As Figure 7.3 details, adjacent to the drawing area of the board is an array of function areas, including the waveform and envelope memory banks, all operational with the touch of the electromagnetic pen. Any waveforms and envelopes that the composer creates on the graphics area are compiled and computed in the real-time unit for immediate aural feedback.

Operating on the graphical input board the composer goes through several stages to create sounds. Having selected the

option to create waveforms with corresponding envelopes, the user needs then to listen to the working page interpreted as musical events. In order to do this the page must be submitted for compilation to the processor. The host downloads the parameters of each arc in the memory of the control unit, including the waveform number, the envelope number, the output channel number, starting and finishing points in the page, any pitch variations etc.. All of the envelopes in the envelope bank are held in the control unit and the waveform bank held in the processor unit. When the user submits the graphical score page to compilation the DSP acts like a conductor, it reads the graphics page into its own memory and then sends the current parameters of each oscillator to the processor. At this stage, the processor uses these parameters to compute the output samples. The contributions of each oscillator to each output channel are added together at this point and sent to the digital-to-analogue convertors.

As a digital composition environment operating at a sample rate of 44.1kHz with a low pass filter at 20kHz, the total number of arcs accommodated within one page can be as many as 2000 allowing for a very dense sonic structure overall. The duration of one page can range from 0.6ms to as long as 1 hour, with loading time on average 6 minutes for a page of over 1000 arcs; it is worthwhile remembering that changing the duration or the *ambitus* ( pitch range) of a page which has already been compiled does not require re-loading.

For pitch range specification, the user operates with the ambitus (pitch range) setting, an imaginary line which runs the

full length of the working screen. The pitch range of a single page can be set as narrow as one semitone or as wide as 10 octaves for the entire page. On playback of a page, there is an interesting feature built into the system whereby playback can be controlled by the user with the electromagnetic pen on the surface of the graphics board. With the pen the composer can control the playback speed and can also listen either from left to right across the board (normal play direction) or right to left, listening to the page of musical events in reverse. This versatility, all in real-time, is useful in critical listening to the produced sounds, allowing the composer to home in on any one particular part of the page of music for as long as he wishes. If the user sets the pen on one particular part of the board corresponding to one arc or set of arcs, the corresponding sound or sonic texture will play, as if frozen in time, for as long as the pen is in contact with the digitiser surface.

Before the user can create an arc on the drawing board, he must assign a waveform and an envelope which will constitute the sonic shape and nature of the arc or sonic event. He chooses a waveform and an envelope from the two banks loaded in the system, accessed from the graphics board on the bottom right labelled *Memory Banks* and numbered 1 – 100. In order to know what type of waveform and envelope he is selecting the user will have requested a printout from the system of the contents of each bank. Figure 7.4 displays a typical printout of the contents of an envelope bank.

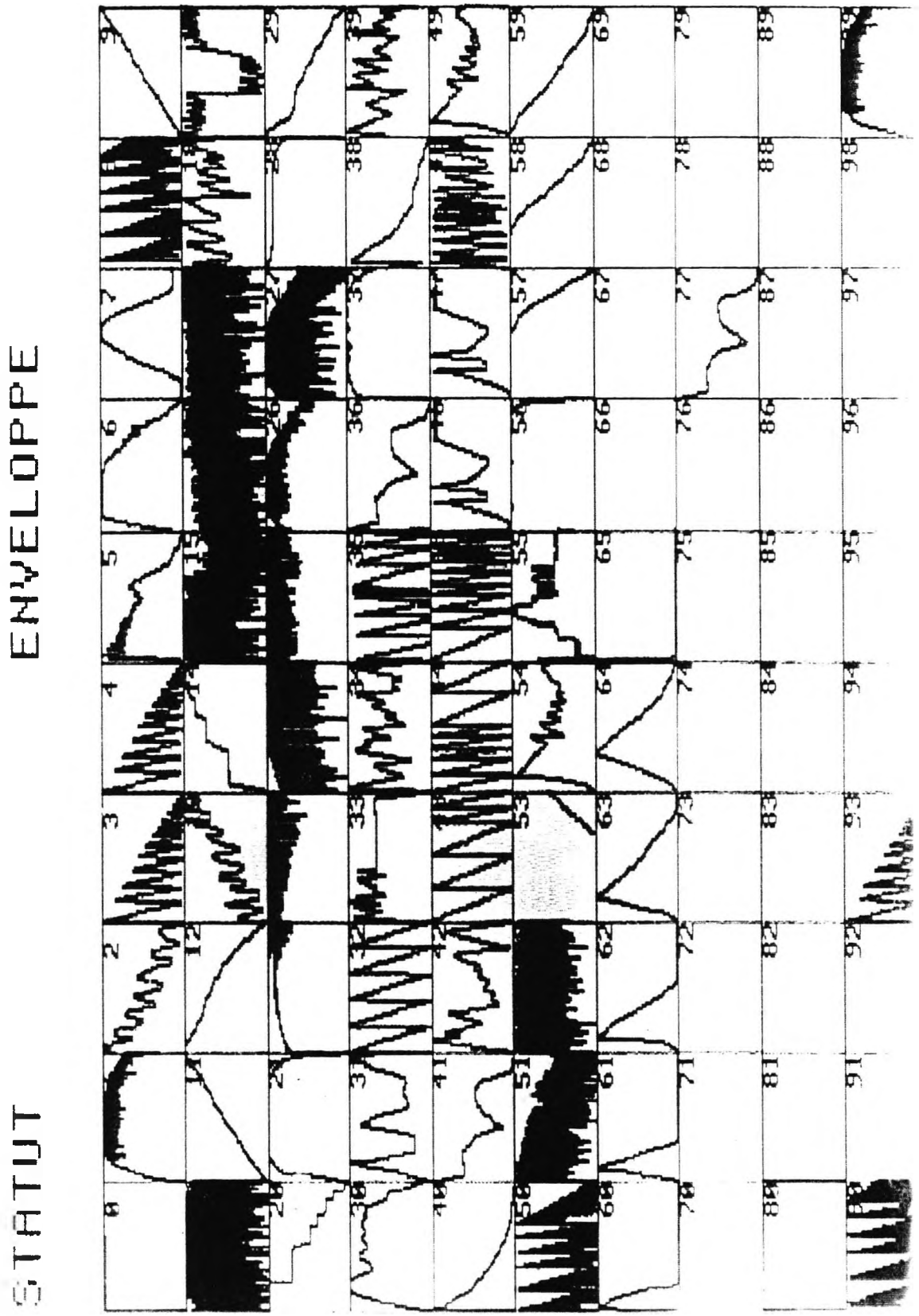



Figure 7.4 Envelope Memory Bank Printout

The user has at all times the opportunity to create new waveforms and envelopes, either manually by drawing them on the board, or by extracting them from input samples to the system.

A step by step operating instruction set for the UPIC composition environment is shown in Figure 7.5, detailing the procedure to draw a sonic arc (as seen in Figure 7.2) on the graphics board working area.

All of these steps are taken by using the light pen on a particular active area on the calibrated drawing board, and the user need never interact physically with the computer system more than actually reading the details of the envelopes and waveforms assigned to any arcs created on the active score page.

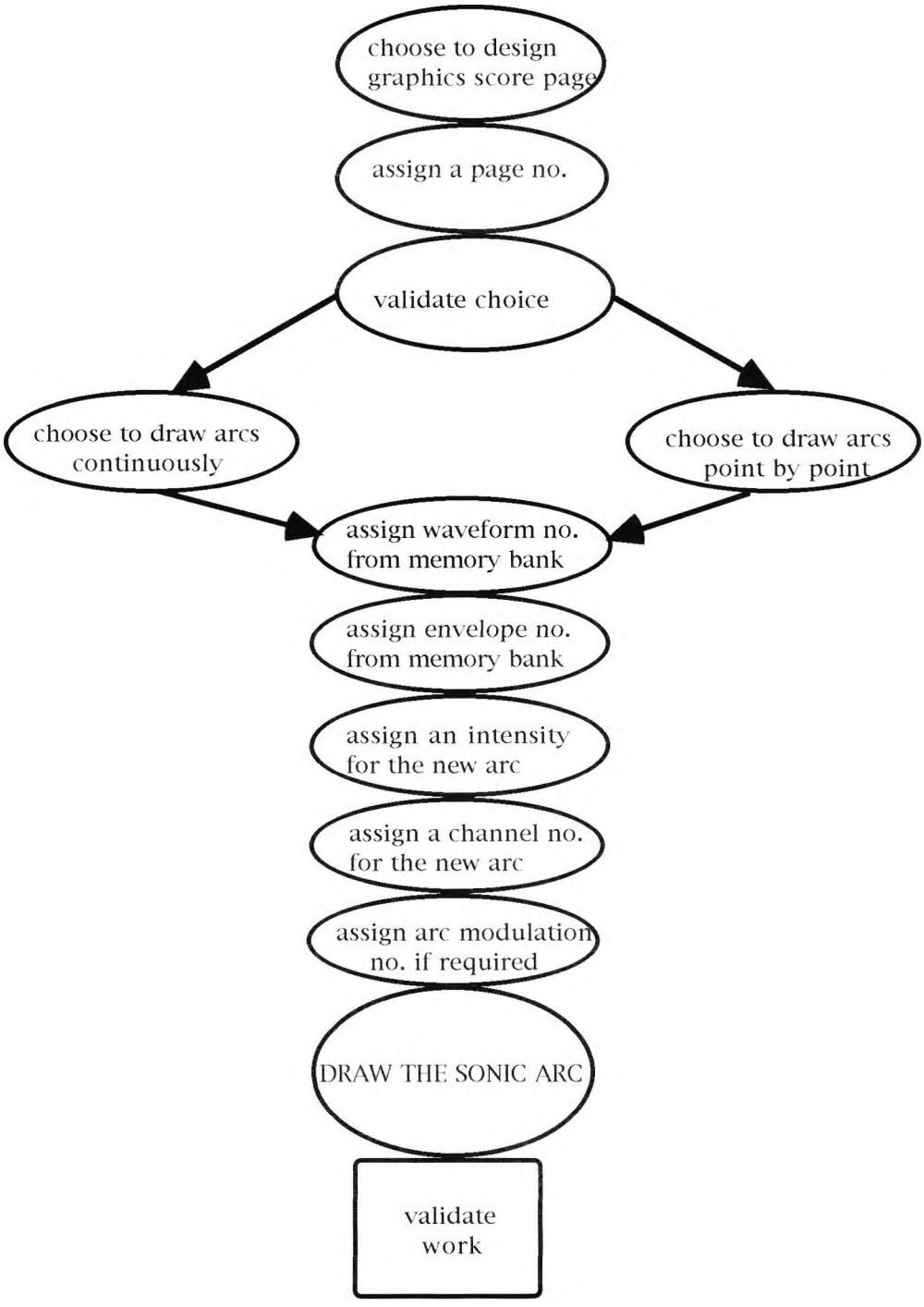



Figure 7.5 Procedure to Draw a Sonic Arc

A printout of the score page as user feedback can be seen in Figure 7.6, and details the information provided to the user during the creation of the score page seen in Figure 7.2.

PAGE 89

| Arc | F. D'o | Env. | Int. | X0 | Y0 | Canal | Arc mod. |
|---|---|---|---|---|---|---|---|
| 000001 | 37 | 50 | +00.0 | 000291 | 003196 | 1 | 000000 |
| 000002 | 37 | 50 | +00.0 | 000298 | 003191 | 1 | 000000 |
| 000003 | 37 | 50 | +00.0 | 000308 | 003227 | 1 | 000000 |
| 000004 | 37 | 50 | +00.0 | 000278 | 003280 | 1 | 000000 |
| 000005 | 1 | 50 | -42.9 | 002239 | 001514 | 0 | 000000 |
| 000006 | 37 | 50 | +00.0 | 002243 | 001514 | 2 | 000000 |

**Figure 7.6 Score Page Printout Details**

The UPIC provides comprehensive frequency modulation operations together with recording facilities for sampling purposes. The user can build up an extensive library of envelopes and waveforms within his own library space as full extraction functions are provided to derive envelopes and waveforms from sound samples recorded via the *Ondes Musicales* sound bank. In conjunction with sampling possibilities, there are comprehensive modification and editing functions provided, which are accessed from active areas on the graphics board. Using these, the user can modify one or several arcs either collectively or individually, within a specified rectangular working area, with the edit functions chosen to be implemented *with* or *without review*. This means that the user can specify an area within which an edit is to occur, and can specify whether each individual arc at a time is to be affected by the edit, or he can decide to do the edit globally within the rectangular area, choosing to affect every arc which is within the area, even those which merely start or end within the area.

As a completely graphics-based user interface, the UPIC operates as a gestural transducer. As the user is able to create a sweeping gesture in graphics form, this is mapped directly to a sweeping gesture in sonic form. Once an envelope and waveform have been assigned to an arc, the physical gesture and shape of that arc is directly related in the closest sense to the production of the output sound the user hears in real-time. Other tangible physical gestures are accommodated on the system, such as the possibility to rotate a sound object in physical space, or to zoom in on a physical object on the screen to increase its size, which maps directly in terms of sound to expanding the sonic gesture to cover a wider pitch base and a longer time frame. Figure 7.7 portrays a single sonic object which has been rotated through successive angles of 90 degrees. At this point, the activity of moving a 'picture' around on a drawing board, and hearing the sonic results in real-time becomes quite abstract, and the mapping of an identical physical object as it appears on the board in various physical positions to what is actually heard in real-time takes some learning on behalf of the user.

The actions of zooming in on a score page element, reducing an object within a page or the whole page in terms of pitch range or time frame, or rotating elements within a page or the entire page as a whole are all carried out by using the light pen in contact with the function box on the board, followed by running the pen along the active drawing area, and watching the computer screen to monitor the toggling values until the desired pitch, rotation angle or zoom amount etc is reached. All of the functions of the UPIC are carried out on the graphics board area,

from single waveform creation, to sampling input sounds, to rotating entire page contents. Even selecting page numbers to be loaded, or copied to, or created is done upon the active area by running the pen along the board until a value is reached.

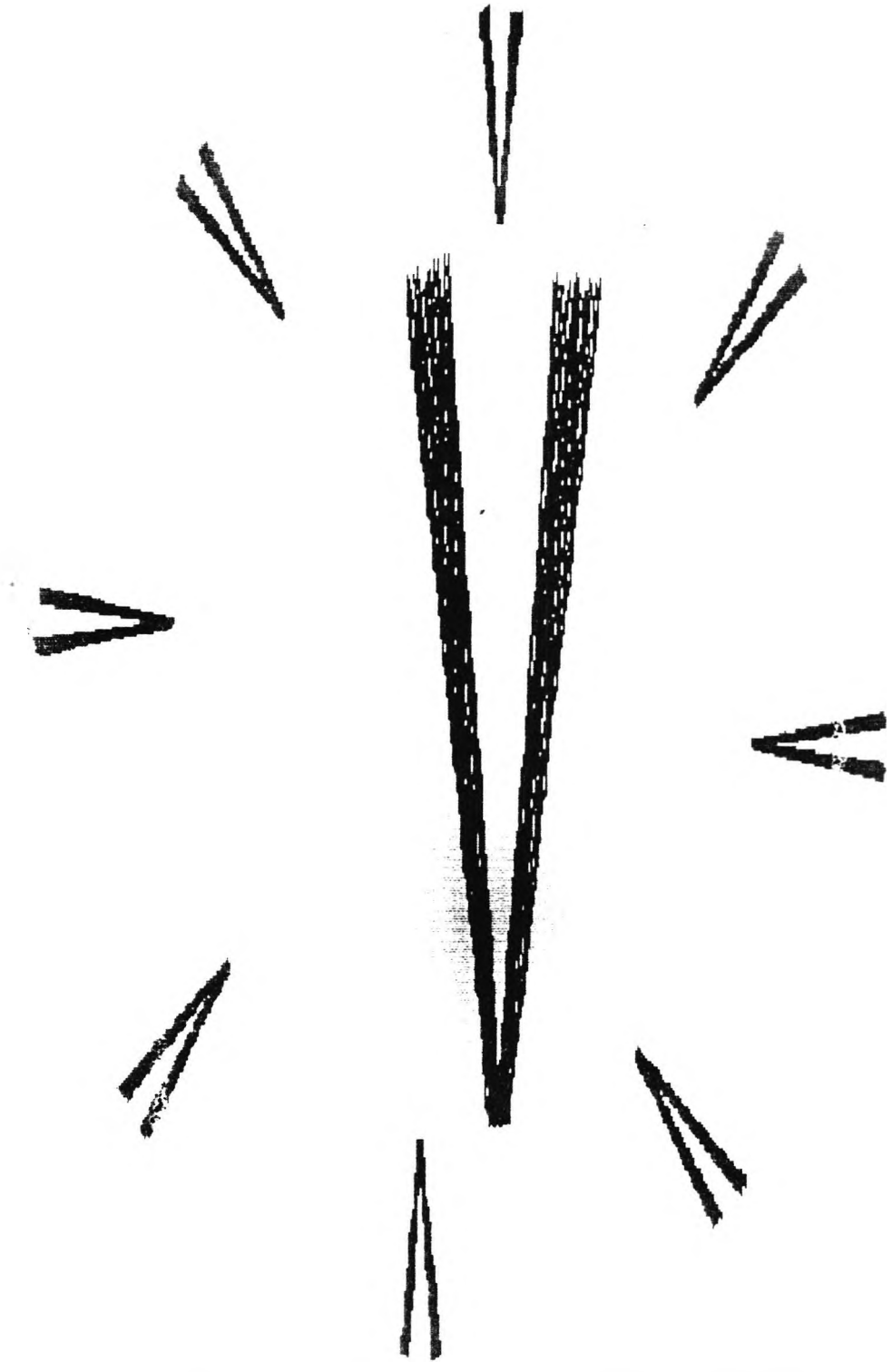


Figure 7.7 Score Page Displaying Rotation Function

With the light pen as the only required physical interface to the system, the user has access to the micro- and the macro-levels of the compositional process at all times. From the most basic micro-level of selecting a waveform and an envelope, then creating a sonic event in time, to actually manipulating at the macro-level entire series of complex sonic gestures in space, the user can span the entire compositional process with the single controller of a light pen, in a completely graphical user environment. This dual approach for composition at the micro-and the macro-levels which the system design encompasses stimulates for the composer a close identification with the sonic arcs at the micro level. This results from successful combinations of waveform and envelope shapes, and from the complex textures at the macro level which the graphical manipulation functions encourage the user to experiment with.

## 7.3 System Synopsis, Analysis and Evaluation.

The analysis and evaluation of the UPIC system centres upon the work which I conducted while at the UPIC workshop, in June of 1990, at Les Ateliers UPIC, Paris. During this time I concentrated to a large extent upon the exploitation of sampled sounds to produce sonic arcs on the working area.

### 7.3.1 UPIC System Synopsis Grid

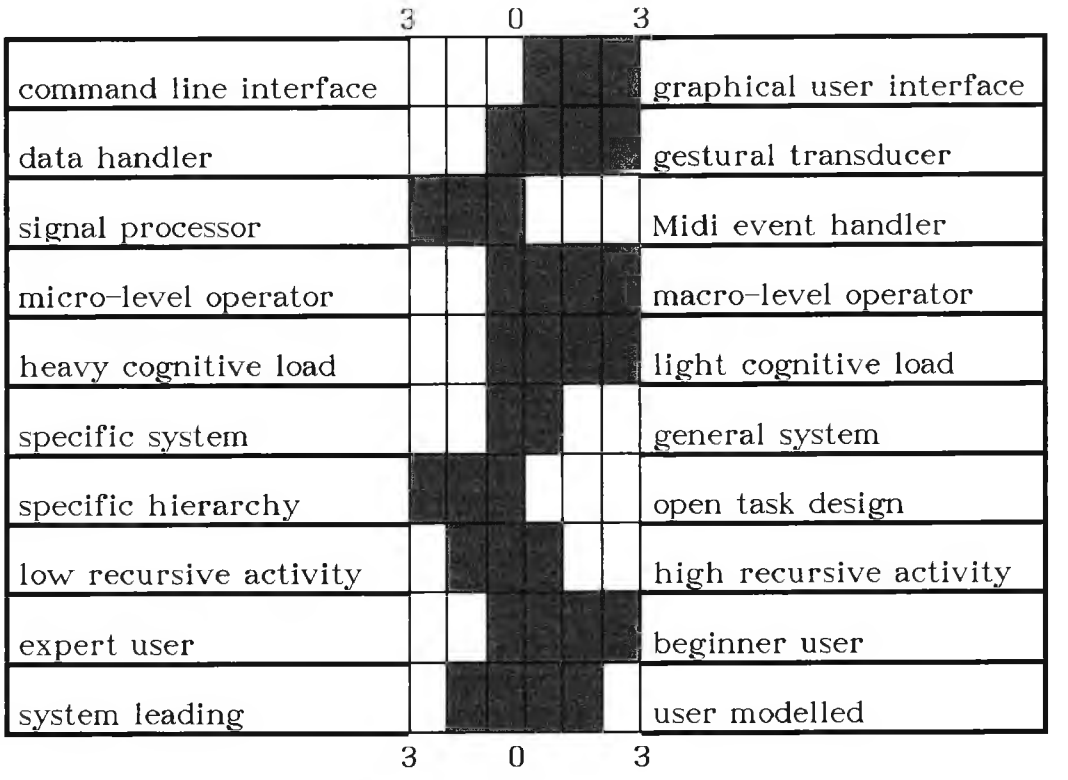



**Figure 7.8 UPIC System Synopsis Grid**

The UPIC system is summarised in Figure 7.8 above, using the Computer Music System Synopsis Grid introduced in Chapter 5. This grid illustrates that the system is completely graphical in its interfacing with the user. and is also virtually exclusively a gesture transducer, with the musician user interacting with data only on the computer screen which details envelope numbers waveform numbers, page numbers etc. As exclusively a signal processing system with no Midi capabilities, the UPIC operates at the macro end of task levels, working with waveforms and envelopes, with macro-level parameters detailing arc numbers and FM arc allocations. As a graphical gestural interface, the UPIC system does not impose a very heavy cognitive load on the user, except when

trying to specify detailed numeric values by using the scale tool on the active drawing area.

As a system which employs only a single input method, and details a very specific composition method, the UPIC editing and error correction capabilities are not free of frustration. The system suits beginners completely, although the mapping of the actual graphical shapes on the drawing board to the aural output is not always immediately apparent. System leading is apparent in the UPIC design, as the user struggles to comprehend the impact of graphical shapes in a drawing space, as they are translated to sonic events in aural space.

### 7.3.2 Analysis and Evaluation

Guidelines derived from Chapter 5 are introduced in italics as a start point for discussion. Evaluations and observations regarding the UPIC system are discussed immediately following each appropriate guideline.

*"...Choose the input device carefully, considering compatibility with system and user, accuracy within the system, efficiency and ease of use..."*

The light pen device with which the user controls the entire UPIC system exploits user familiarity with using a pen, and also

facilitates an immediate input method, with a high[7] learning curve. However, once consideration is given to the accuracy with which the user is expected to be able to user the light pen, and the thickness of the actual instrument, the user becomes less than content with a light pen which proves very difficult to position exactly on the numerical values setting for intensity, or data mapping for page numbers, envelope numbers etc. on the scale which is user controlled across the drawing board.

*"...Simplicity should be aimed for..."*

Using one method of input, and also affording the user the ability to work at all times on one active area, the UPIC embraces simplicity in its design and philosophy. The system operates completely in a graphical domain akin to the drawing skills demonstrated by children. However, the artistic output from the system separates the graphic art from the music.

*"...The virtual software interface should shield the user from the computations happening at lower levels within the system .."*

As the graphics digitiser is the active interface for the user, the UPIC operates perfectly as a system which allows the user to virtually disregard the computations required to create the aural output from the machine. The creative task is not interfered with by the processor activities. In this way, the user is not distracted

[7] A system which is easily learned by the user is seen to have a high learning curve; a system which is more cumbersome to learn is seen to have a low learning curve.

from considerations regarding sound out in order to worry about algorithms, data strings etc..

*"...Afford the user complete/complex error handling..."*

*"...Allow the user to easily undo or reverse an action..."*

Within the UPIC system, the procedures for selecting particular arcs for editing, or for selecting certain areas of the score page for a global edit function are far from satisfactory. In order to delete, or edit arcs on the score page, the user must specify a rectangular edit working area within which two types of edit – *avec revue*[8] or *sans revue*[9] can be effected. The *avec revue* function seems to be the best choice when the user is attempting to alter details on a sparse score page, but the fact that the system requires verification of every arc selected within the specified rectangular working area makes the process very tedious for more complex textures. The *sans revue* function, however, can collectively alter arcs which the user did not wish to change.

*"...Allow the user to work serially with a minimum amount of simultaneous action from other screen events unrelated to the task in hand..."*

In its task design the UPIC allows the user to operate through a set of very defined task steps in order to create a sonic arc on the digitiser drawing board. A process of clicks to select a function,

---

[8] Avec revue - with review, specifying one-by-one selection of arcs within an area.

[9] Sans revue - without review, indicating that all arcs within a rectangular area are to be globally affected by the edit function in question.

combined with a physical gesture to create a graphic shape on the screen is all the task processing that the user is involved with. No simultaneous activity occurs to distract the user from the straightforward task in hand.

*"...Provide visual portrayal of signals for analysis and decomposition, and manipulation..."*

As a completely graphical interface, the UPIC matches and exploits user mental models to provide a cognitively familiar working environment. As envelopes and waveforms can be produced in hard copy printout from the system waveform and envelope memory banks, the user is presented with visual representations of waveforms exactly as his mental model perceives them. This results in a very acceptable and familiar environment for the composer.

*"...At all times, define the user primarily as a musician, who is employing computer technology as a means to an end..."*

The philosophy behind the design of the UPIC engenders the idea that everyone is essentially creative, and only needs the correct tools in order to be able to express that creativity (7:115). As a machine designed as a result of one composer's dream, and realised with the aim of giving everyone, at all ages, the ability to express themselves creatively, the UPIC certainly works from the premise that the user is a creative entity, employing the machine as a means to a creative end.

*"...Provide the user with informative feedback at all levels of task completion..."*

With the main feedback given to the UPIC user being the aural output of compiled sonic arcs, the machine operates closely to a model of a traditional instrument – an action produces an acoustic result, although the output from the UPIC is not consistent as it may be in the case of a traditional instrument. However, the system fails to provide the user with comprehensive details regarding the waveforms and envelopes used on score pages, unless the user physically removes himself from the working area and reads the computer screen listings available to the side of the digitiser board area. Also, when the user attempts to set pitch ranges, or select page numbers, the activity of dragging the light pen across the board until the correct data value is achieved is very badly fed back to the user. The data is visible on the computer screen, which is physically remote from the active working area.

*"...Define and maintain a system balance between generality and strength..."*

As a system general enough in operation as to attract even the very young beginner, and also specific enough in its compositional approach as to challenge professional composers in the task of creativity, the UPIC balances effectively between general application and specific sound manipulation. However, with a very specific compositional approach, only very careful input and selection of sampled waveforms and creative use of envelopes and

editing functions will produce a creative output which will sound individual and not be heavily stamped with the UPIC system identity.

*"...Reduce user cognitive load to leave the musician user free to concentrate on the aural output, and not to have to unduly worry about system operation..."*

There can be little argument that once the user has embraced the UPIC compositional approach, the ease of operation of the interface interferes little with the creative goal. Even battling with less than ideal editing functions, and trying to undo unwanted actions detracts very little from the composers attention, as the aural feedback from the system is immediate.

*"...Strike a balance between a system which is a gesture transducer and one which is an information handler...*

As a system which relies heavily on transducing users actions in the graphical domain, the UPIC is clearly first and foremost a gesture transducer. However, the balance with handling data would be greatly increased if the information relating to numericals for score page numbers, pitch ranges, and page durations etc were handled in a more user friendly fashion. The scaling line which is created on the score page area when a function requiring a numerical is called is very difficult to set exactly, and causes frustration for the user.

*"...Provide flexible internal libraries, and allow for extendibility of libraries and algorithms..."*

The UPIC system provides a useful set of libraries of waveforms and envelopes to get the user started, and also has comprehensive sampling functions built in to allow the user to create his own library additions. This extendibility is one of the ways in which the composer can attempt to make his work sound individual and not merely UPIC-like, as so many scores are heavily identified with this machine.

As an interface which allows the user to remain in the graphical domain, and to create aural textures from physical gestures, the UPIC system gives the composer the opportunity to experiment freely, in a real-time environment, modelled on conventional actions and expectations.

# Chapter 8 Syter

## Real-Time Composition And Sound Processing Environment

### 8.1 Introduction to the System.

The Syter real-time composition and sound sampling environment manifests itself to the user in the form of two computer screens, one high resolution for command line work, the other colour for graphics manipulation[1]. The user operates the command line screen from an alpha-numeric keyboard, and inputs to the graphics screen by way of four-button mouse movement on a tablet surface. The position of the mouse is mapped to the screen totally relative to its position on the tablet surface. It is not position renewable like most mouse operated environments. Figure 8.1 below details the architectural layout of the Syter system as the user approaches it.

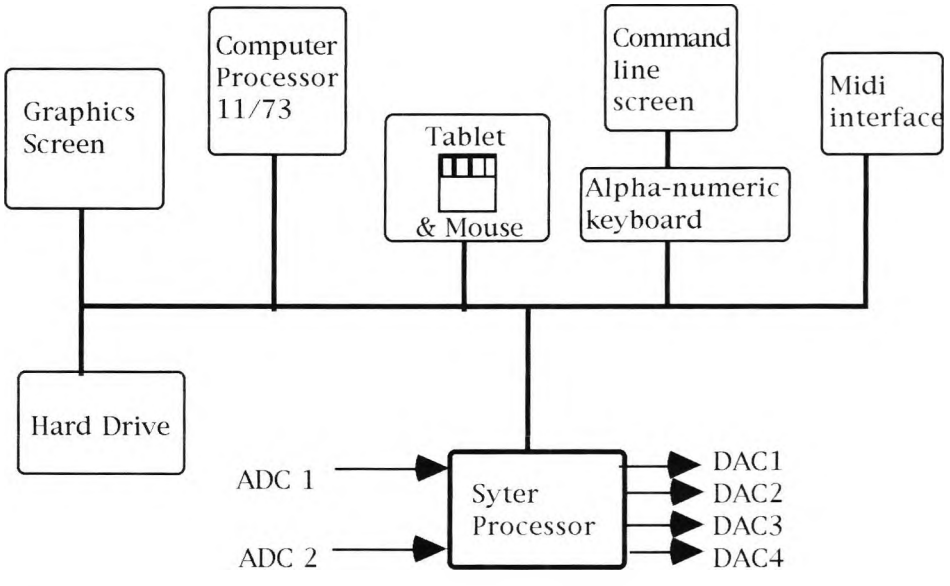



Figure 8.1 Layout of the Syter System.

[1] The versions experimented with by the author in 1991 and again in 1993 were respectively Version 6.2 and Version 6.31. At the time of writing, a variation of the Syter system is available on personal computer systems, as GRM TOOLS, running on the Macintosh computer range.

On the direct control level, the Syter system affords the user two command line interpreters:

- MCR (>) Monitor Control Routine.
- DCL ($) Digital Command Language.

DCL is the default interpreter. The commands available within this command-line interpreter are very similar to those of MS-DOS, the interpreter common to virtually all IBM PCs and compatible machines, e.g.:

- $DIR - directory command
- $COPY - copy command
- $LS - list command

The system allows for several users to operate at different terminals at any one time. Similar to every multi-user facility, each user is attributed file space on the core hard disk.

## 8.2 System Details and Mode of Operation.

Within the Syter system, several working environments are available. The composer can choose to operate on the graphics screen in the real-time (SYG) environment with its set of seven graphical screen areas and a full graphical editing facility (EGS). On the command line, a 'conversational command'

environment, not operating in real-time, affords the user programmer-like contact with the operating system. A working session would typically involve constant interchange by the user between the two working environments.

For the user, within either the graphical or the command line environments, there are three types of files to operate with:

- Instruments:

  a configuration of SYG operators, chained together in a particular fashion to affect an input sound file. There is a library of instruments available publicly on the system and the user also has the option to create and store instruments of his own by 'programming' a configuration of operators together.

- Sounds:

  sound files either recorded by the user and subsequently stored on his partition of the hard drive, or some basic sound already resident in the public library.

- Tables:

  a collection of parameter values constituting the 'internal' information for an instrument e.g.: waveform values, type of waveform etc.

and two library types:

- public - accessible to every user
- personal - accessible only to logged-on user

At all times, the user must only write to his own directory and must never destroy publicly available files.

Sound creation, control and sample treatment in the graphical domain is operated within the real-time environment of the SYG software. The most immediately approachable part of the Syter to the composer, the collection of graphic screens available on the Syter under SYG afford the user real-time control of many sound manipulation techniques.

In traditional acoustic writing, when the composer assigns a note to a particular instrument, he gets a certain timbre produced which would be completely different if the composer had selected a different instrument to play that note. This sound-to-instrument matching process is at the heart of the Syter system. The composer selects any one instrument from an available library and applies it to the sound selected for manipulation. This sound-to-instrument matching is carried out on the graphics screen environment on the Syter.

### 8.2.1 Syter Graphics Screen Environment - SYG.

The graphics screen working areas available are:

- FILE screen
- RULER screens (1 & 2)
- INTERPOLATION screen
- TABLE screen
- ENTER and EXIT screens

The screens are controlled completely by mouse movement and movement between screens is by way of a horizontal menu on the bottom of every screen. A diagrammatic representation of each screen follows, coupled with an explanation of the controls available on each. This will, collectively, afford a degree of understanding in relation to the mode of operation in this graphics environment and also an indication of the functions available for sound manipulation.

- The File Screen.

This screen is at entry level in the SYG area. On the upper-most level of the graphics working environment, this screen very simply allows the user to load, create, update or destroy files.

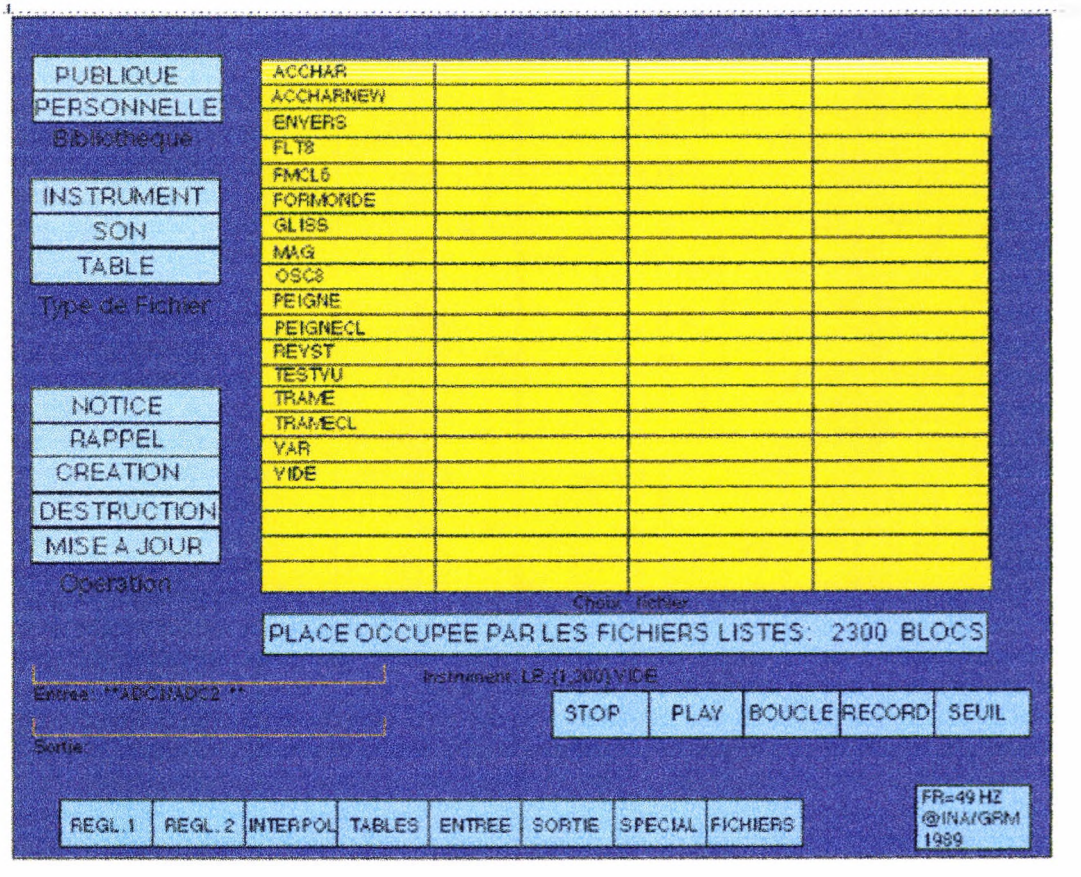



Figure 8.2 The File Screen

Operating completely within the graphical domain, this screen allows the user to select either sound files instruments or table listing from either the public library or his personal library, and to play them back in real time. The user can trigger the file manually, or can set a threshold (seuil) to allow the file to play back automatically. Playback can be either straight (play), or looped repeatedly (boucle). The menu running across the bottom of the screen is common to all the Syter screens, and allows for movement between the various screens. By mouse movement and clicking the yellow mouse button (in this order) the user selects files by choosing in sequence as follows:

- a library
- a type of file
- a particular file
- a particular operation

and then, in normal working circumstances, he would proceed to the next window, having thus specified which file he will subsequently manipulate.

- The Ruler Screens

These are the most important screens in the working environment, as they allow for the control, calibration and visualisation of each parameter within any instrument. There are two ruler screens, allowing for the design and control of instruments with up to 16 variables.

Each ruler controls one parameter, with the name of the parameter at the base of the ruler. The colour of the name indicates whether or not the parameter values can be varied by the mouse movement within the ruler (red = active, green = deactivated). The values can be altered in point step fashion, or continuously. Each ruler consists of 1024 control values.

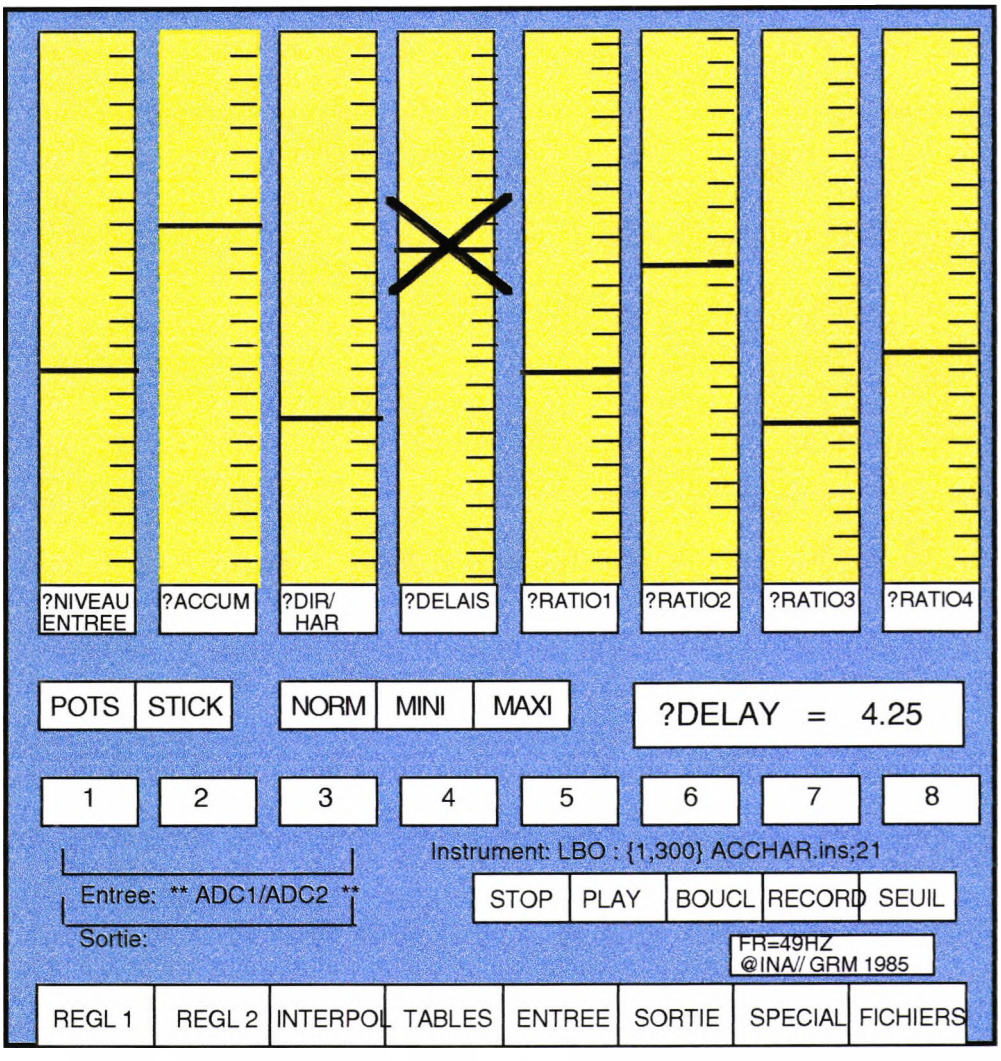



Figure 8.3 The Ruler Screen

The ruler settings can be manipulated individually, as *pots*, as would be the case in the screen diagram above, or they may be set in pairs, 1+2, 3+4 etc, by selecting *sticks*. In this case the

rulers to be manipulated together would graphically merge on the screen, for a single cross hair cursor to alter the level common to both.

With the *norm*, *mini* and *maxi* boxes, the user has the facility to alter the upper and lower value ranges controlled within each ruler. This is done by clicking on the box required e.g. *mini* and then dragging the ruler level indicator to the desired level, followed by setting the *maxi* value in the same way. The ruler then recalibrates its internal values to suit the new upper and lower limits.

- The Interpolation Screen

By memorising the parameter values as previously defined on the ruler screens, this screen allows the user to move around in sonic space between various versions of the same sound by playing in the active screen area with the mouse, depressing the yellow mouse button. The parameter settings from any one particular set of rulers is saved and presented on screen in the form of a ball, with different sizes relative to their importance. The Interpolation Screen is seen in Figure 8.4.

In order to prompt user memory, the sonic balls are numbered as they are created, in case the user wishes to travel between them in creation order. Inside the balls, the parameter values do not change from those as set on the previous ruler screens. In the area between each ball, an

interpolation between the different values of the adjacent balls is produced, the larger size balls influencing the interpolation more, the smaller sonic balls not having nearly as much influence in the resultant interpolated values.

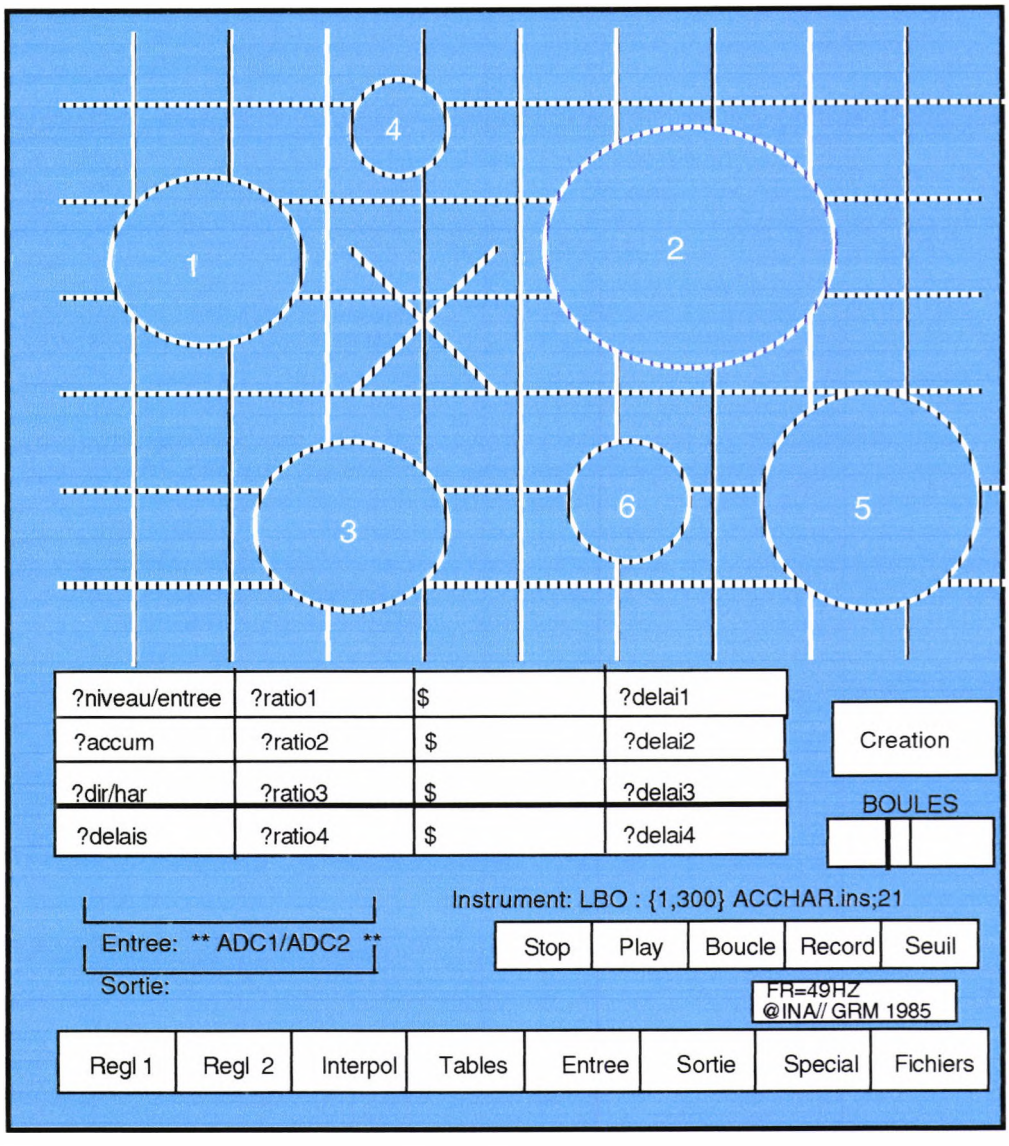



Figure 8.4 The Interpolation Screen

When this screen is first activated and the user creates a sonic play area, all the variables are memorised as they were last set on the ruler screens. However, it is possible to de-activate specific parameters by selecting them in the parameter region

on this screen. Colour is used, as in all the screens, to indicate activation (red) or deactivation (green).

The user will define the physical size of the sonic ball to be created by setting the bar indicator in the *boule* box. More to the left indicates smaller, moving to the right enlarges the ball to be drawn. The box marked *creation* in the diagram toggles between three settings in real time use, creating a new ball, playing a sonic ball or destroying a selected ball.

- The Table Screen

In this screen seen in Figure 8.5, the user can view the constituent tables of any instrument, with an aim to editing/modifying them. The user can view and modify the waveform table representing any instrument, and also can create an amplitude envelope for any instrument. Any modification is done by clicking the yellow mouse button and drawing with the mouse to replace/modify the existing curve.

Within this screen, the user can choose between waveform or spectrum display of the table information, and can also toggle between linear or exponential scaling in the display, in the screen area seen in Figure 8.5 marked as *Modes de Representation.*

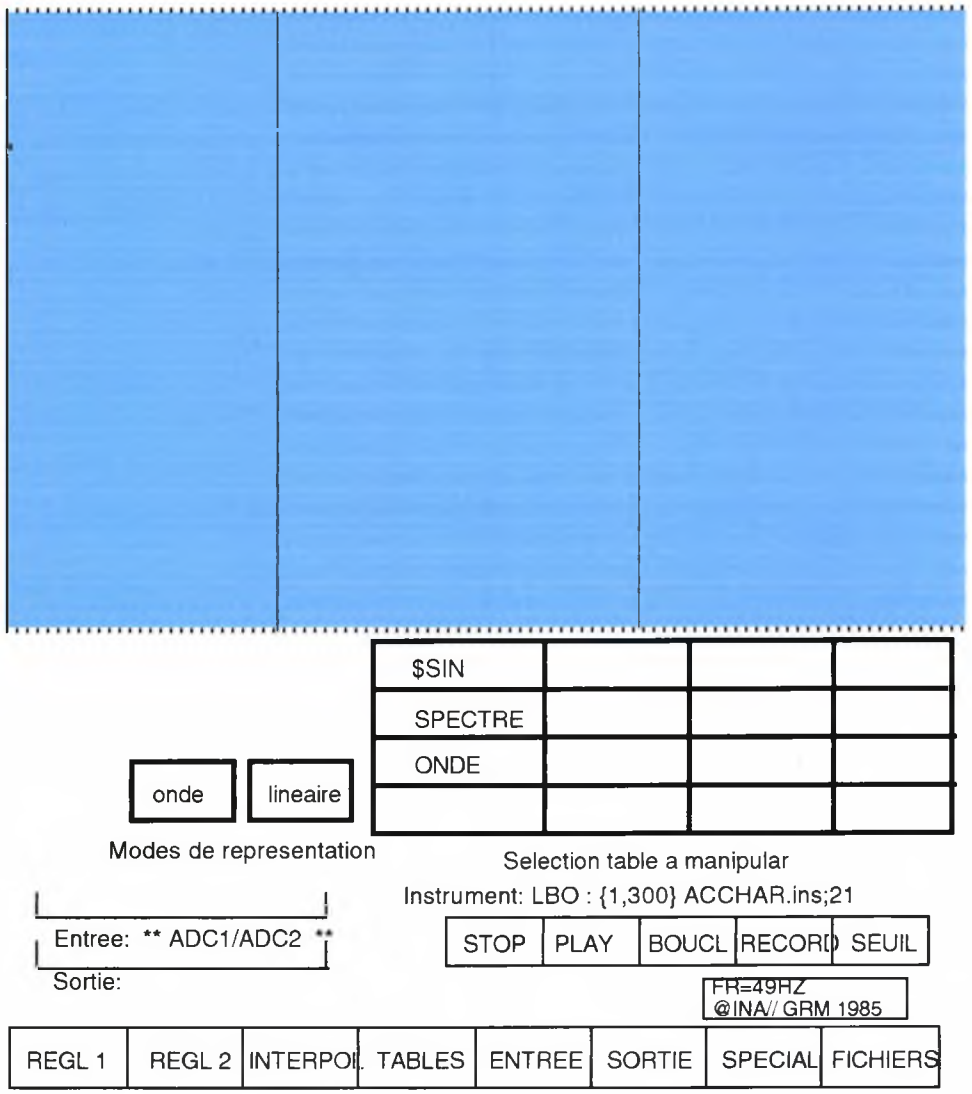


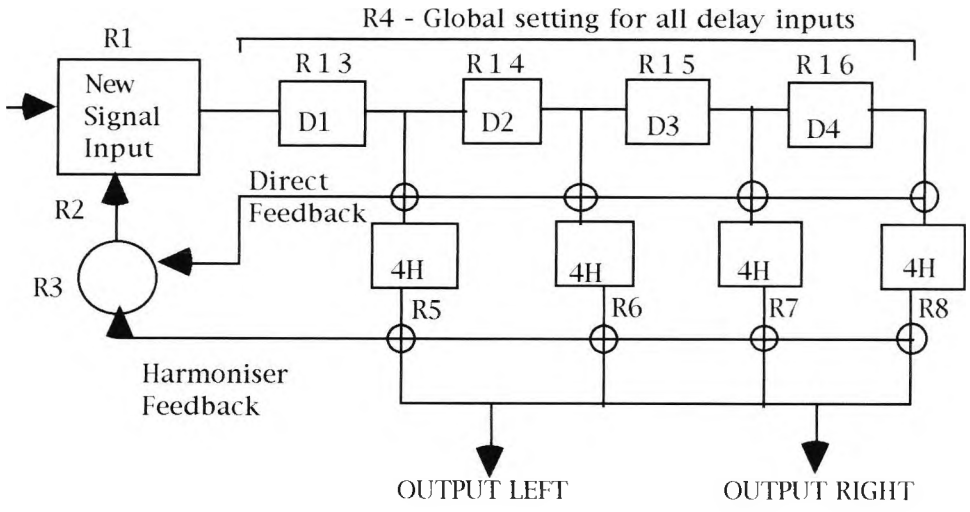

Figure 8.5 The Table Screen

- The Entry And Exit Screens

These two screens enable the user to visualise/control the amplitude levels of sound files coming from (entrée), and going to (sortie), the hard disk recording system. It is also possible to isolate certain portions of the sound file graphically, and to play backwards through a soundfile.

### 8.2.2 Syter Instruments.

A SYG environment instrument represents a certain configuration of SYG operators, easily paralleled for the non-familiar user to a patch diagram. Seen in Figure 8.6 below as an example is the patch diagram of the *acchar* instrument, which is a publicly available accumulation instrument with 4 harmonisers, 4 delay lines and feedback. Included in the diagram is an indication of which rulers on the Syter ruler screen would control the individual parameter values within the patch. As a user, it appeared very useful to reduce each of the publicly available instruments to a patch configuration of this type, in order to be clear of exactly what type of operation was being carried out on an input sound and exactly what parameter each ruler was controlling in the graphics environment.




R3 - ruler 3 controls the amount of direct feedback over the amount of transposed feedback, and vice versa.

Figure 8.6 Acchar Instrument Patch Diagram.

The functioning of the Syter in real-time dictates that an instrument must always be present. Upon boot-up, an invisible instrument *Vide* is loaded by default. Only one instrument can reside in the system at any one time, so loading any instrument results in the elimination of the previously loaded instrument from instantaneous memory.

In the early stages of using the system, the novice user is unlikely to have the necessary competence to design his own instruments. In order to facilitate some degree of user interaction with the system in the early stages, and also to provide a comprehensive instrument library for the practised user to further develop, there is a public access set of instruments on the Syter, available to any and all users, from within any directory, to utilise with their own sound files. In Appendix E a complete list of all the public instruments available is given, coupled with a brief description of each one.

As previously stated, the order of operation is to impose an instrument on a sound for manipulation. Detailed manipulation is carried out on the ruler and interpolation screens, by altering ruler (i.e. parameter) values and also by moving around in sonic space to create interpolations between set parameter values. The original sound can always be heard by loading the blank instrument known as *Mag* to work with the sound sample. This very straight forward method of manipulation and control provides a very powerful real-time sound processing environment.

### 8.2.3 The EGS Real-Time Editing System

The Syter system provides the user with a comprehensive sample editing and mixing environment. With the combination of four different operative screens, the composer has very detailed control over the chosen files he wishes to manipulate, with the resulting output being written to a new file. Control of each screen is managed by the four button mouse on a tablet surface with corresponding colour coded pointers effecting parameters on the screen. The files chosen to be edited in some fashion are tagged as A and A' and the new file output combination of these is tagged B. It is important to realise at this point that A' is not a variation on File A.

The colour coded pointers which vary in operation across the EGS editing screens are as follows:

- Within the A and A' file screen areas:
  - Green pointer – cuts sections from the sound file
  - Blue pointer – selects segments of the sound file outside which the user will hear nothing.
  - Red pointer – not in use in these areas.

- Within the B file screen area:
  - Green pointer – not operational in this screen segment
  - Blue pointer – cuts sound file segments
  - Red pointer – allows the user to select what section he wants to hear before, during and after the Blue chosen area.

Once the user becomes accustomed to the different colour coding to be used on various file areas on screen, the editing facility

within Syter becomes a graphical cut, copy and paste type function.

Within the EGS editing environment the available working screens are as follows:

- The File Screen
- The Command Screen
- The Interpol[2] Screen
- The Repérage[3] Screen

Each of these screens is introduced and discussed in turn below.

- The File Screen

As seen in Figure 8.7, this screen provides facilities for choosing files for manipulation, loaded to the environment as File A, and A', creating the edited result as File B. Level faders provided let the user control the output level of each file upon playback.

[2] Interpol - referencing the action of interpolation of file contents.
[3] Repérage - presents a pitch over time graphical view of soundfiles.

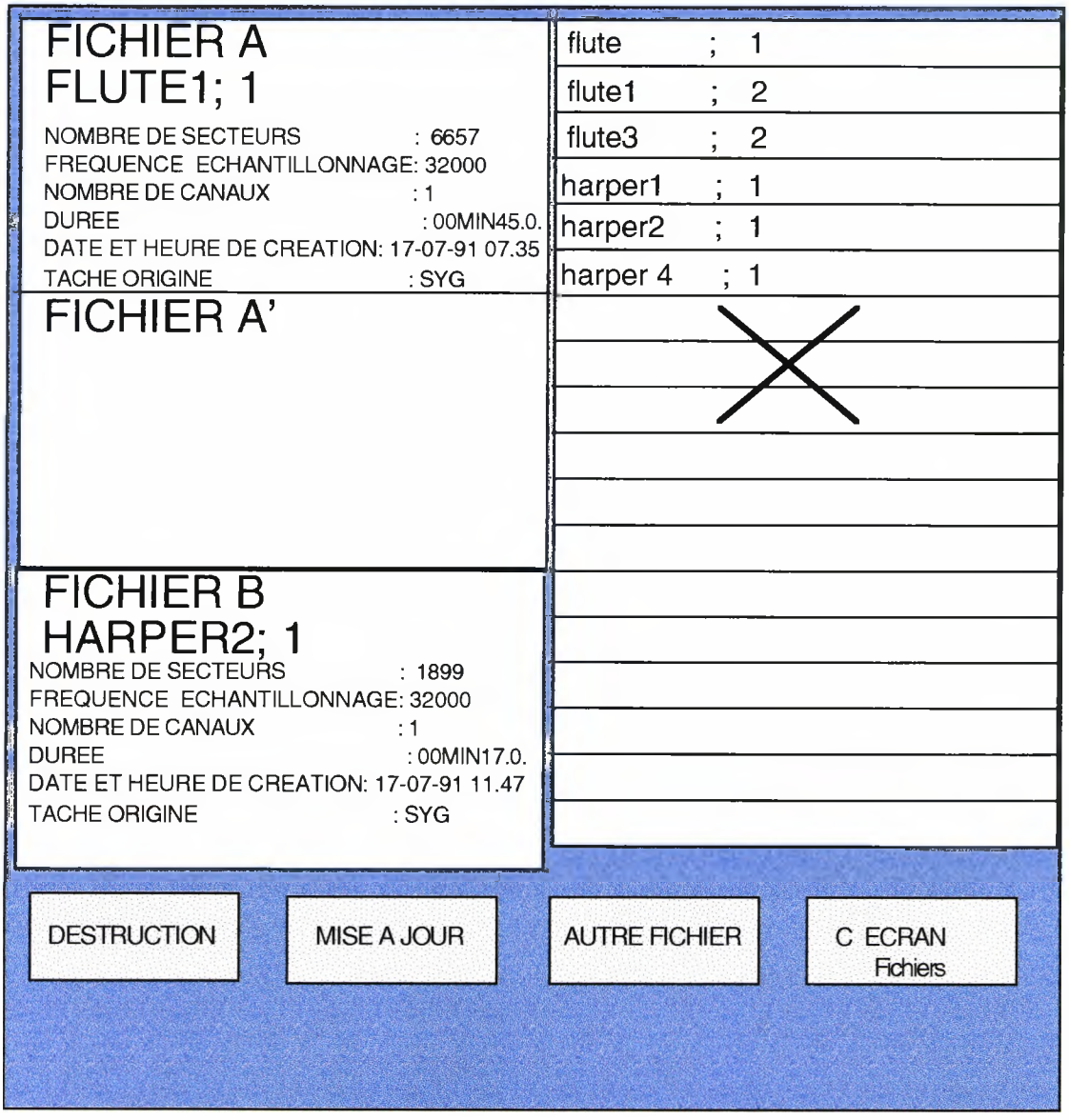



Figure 8.7 EGS File Screen

The directory listing on the right side of the screen displays all the available files in the user's personal directory. As any file is selected as A or A', all the details of that file relating to sampling frequency, number of channels etc. is detailed in the corresponding file section on the left of the screen. Instructions for deleting a file, or selecting a file are located across the bottom of the screen.

- The Command Screen

Seen in Figure 8.8, this screen permits sectional editing of the files previously selected for treatment and contains the commands required for cutting, copying, pasting and insertion of excerpts from files A and A' to the resulting file B.

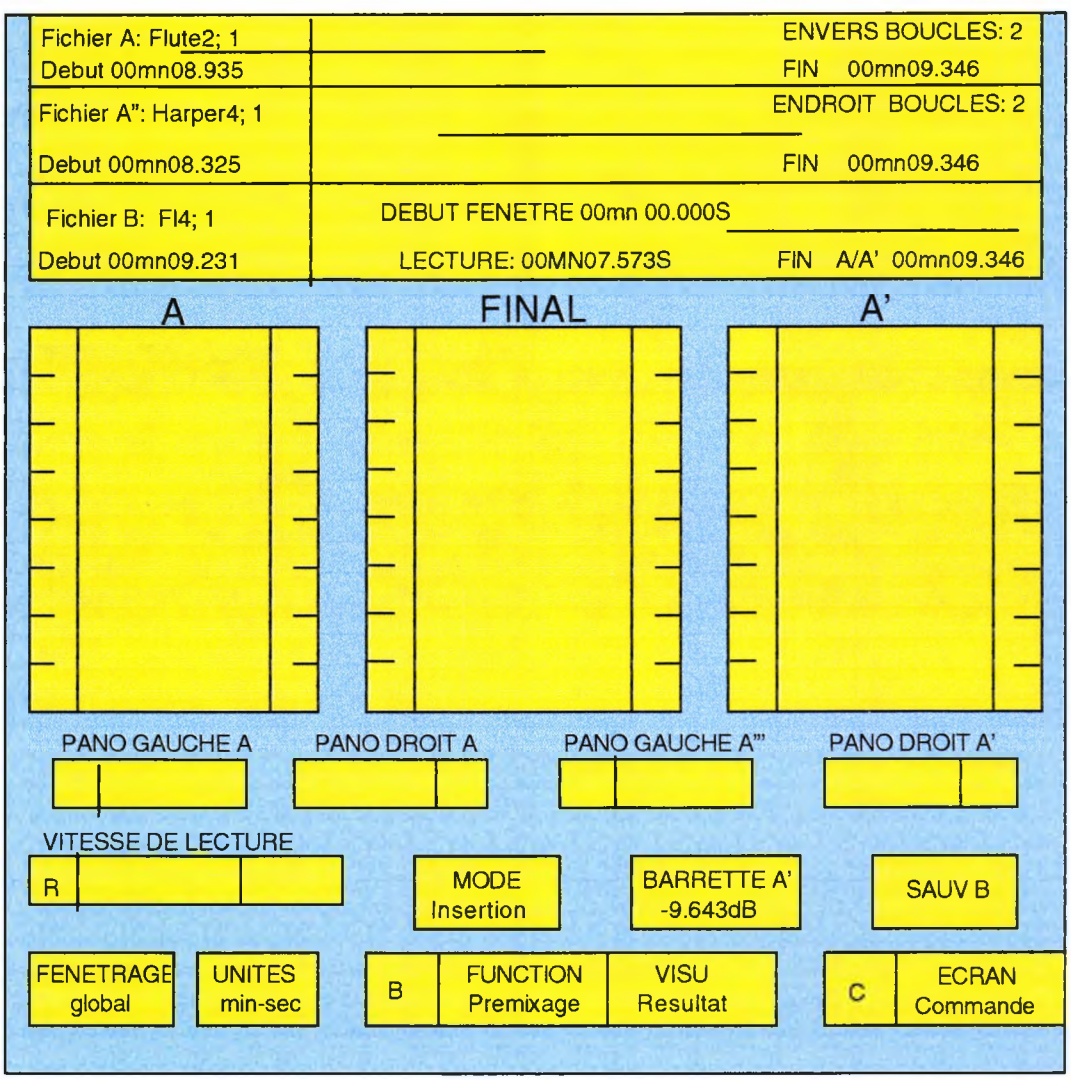



Figure 8.8 The Command Screen of the EGS Environment

Information relating to cross fade points, and layering positions are detailed in the boxes across the top of the screen. Panning is controlled in the boxes below the three main level indicator boxes which dominate the screen by moving the indicator left or right in the box relating to the individual file.

- The Interpol Screen

Similar to the interpolation functions found in the SYG environment where particular parameter settings within a single file were interpolated between, this screen allows for interpolation functions in the same manner but between different distinct files. The user moves the cursor between, around, into and out of the circles representing various sound files to cause mathematical interpolation of the data, relative to the cursor position on the screen. The algorithms involved are constantly updated with values calculated from the cursor's relative position with respect to the centre of the circle with which it is interacting. This depends on the parameters and diameter of each circle drawn by the user on the screen.

- The Repérage Screen.

This screen provides a graphics depiction in pitch presented over time of a selected file during playback from the Syter processor, either one of the source selected files A or A', or the resulting edited file B. Choosing either a linear (0-1) scale or a logarithmic

display (between -36 to +12dB), the user can easily visualise levels and/or points of interest within the sound file. This screen can be very useful to pinpoint areas of concern when difficulties arise due to distortion after various files have been edited together.

### 8.2.4 Command Line Control : Conversational Commands.

Together with real-time operation, the SYG software can also be accessed and operated in what is referred to as 'conversational mode'. A complete set of 'conversational commands' affords the user full and comprehensive communication with the system, in order to allow for system management, control and also programming. The list of conversational commands can be found in Appendix 8B, together with a brief explanation of each one for reference purposes. This very simple command line input gives the user control of system functions such as listing tables in the memory bank, listing the sonic circles used in interpolation screens, name and save files in various directories, and also control the Midi interfacing to the Syter system. The user can also build complete new instrument configurations within the SYG conversational command line environment.

In essence the user draws up an instrument patch in software which will be compiled and written up as an instrument in the user's own directory. This is referred to as programming by the Syter designers. However, the design and

building of new instruments is much more simple than the word might imply. In essence, the creation of a new instrument merely involves the linking together of pre-programmed elements. The user simply selects the Syter modules which he wishes to link together and then, using the conversational commands, creates the links between the modules. In this way, the system gives the composer the ability to create patches without specialist programming knowledge.

Working either completely within the SYG and EGS graphics areas or on the command line within Syter or more typically within a combination of all three environments, the Syter provides the composer with a comprehensive controllable working space, open for experimentation and encouraging exploration within the system.

## 8.3 System Synopsis, Analysis and Evaluation.

The analysis and evaluation of the Syter system is based on two separate residencies at the GRM, one of which started with a taught course presented by Daniel Teruggi[4] introducing the Syter system.

[4] Daniel Teruggi is a member of the Groupe de Recherche Musicales at INA-GRM.

### 8.3.1 Syter System Synopsis Grid

The Syter system is summarised in Figure 8.9 below using the System Synopsis Grid introduced in Chapter 5. This grid should be used to reference analytical detail and evaluative issues in the following sections of this chapter.

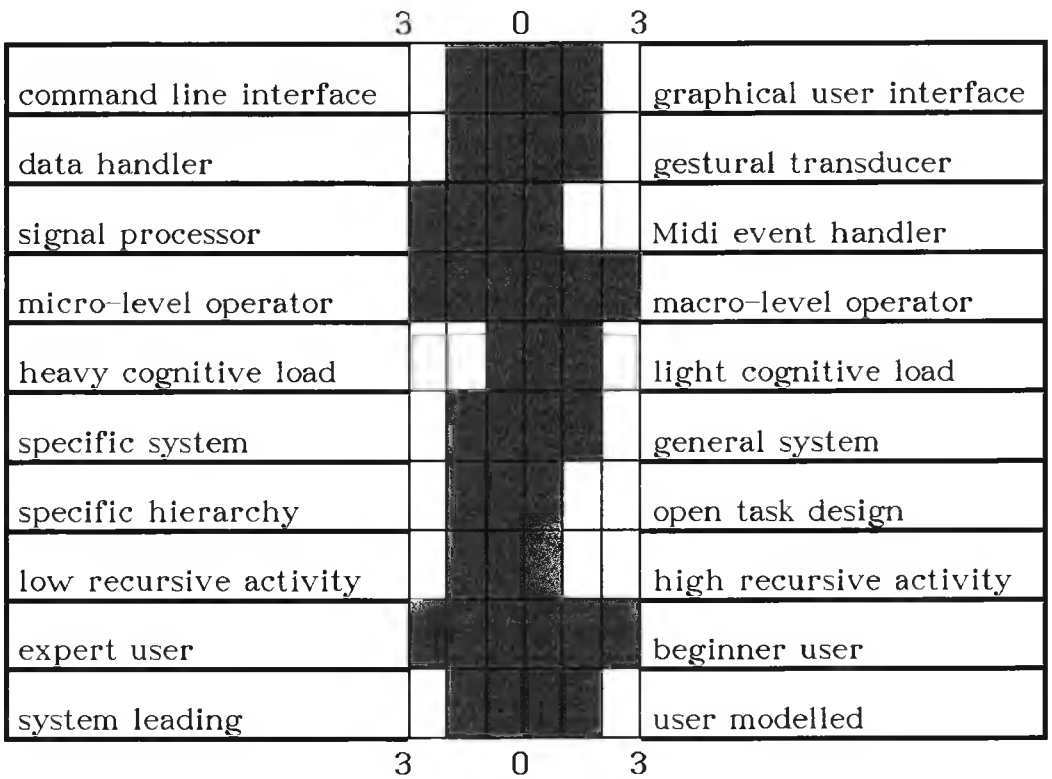



Figure 8.9 Syter System Synopsis Grid

The Syter system is seen to be balanced in its employment of command line input and graphical interface control. Also, the system balances between transducing gestures and handling data input from the user. As a system which is virtually completely a digital signal processor, the Syter handles Midi input via a specially designed operator. The composer using the Syter can work across the compositional range from micro-level tasks to macro-level control of sound files, with a medium to light cognitive load,

depending on which area of the system is being used. The system is classified as equally specific and general in its compositional approach, as it encompasses system particular methods of sound creation, but employs general methods of sound manipulation and editing, albeit with novel interfacing techniques. Editing operations on the Syter are less than straightforward, with much confusion due to the use of various mouse buttons to carry out differently mapped functions from one Syter screen to the next. As a system with a flexible task structure, the Syter accommodates beginners easily with fully functional library files available for immediate use. The system also attracts expert users with the possibilities of internal design of personal instruments, and complex operator combinations possible. As a compositional system, the Syter is balanced well between the degree of system leading and user modelling which occurs, which depends to a large extent on user ability levels.

### 8.3.2 Analysis and Evaluation

Guidelines derived from Chapter 5 are introduced in italics and evaluations and observations relating to the Syter system are discussed immediately following the appropriate guideline.

*"...Use natural sentences and natural language words...Use consistent syntax across the interface..."*

Throughout the various interface screens of the Syter system, user models of language and syntax are very closely adhered to, and held consistently between screens. Expectations on behalf of a user

who believes he is manipulating parameters such as duration, frequency, panning degree, delay values etc are not disappointed. The effect of consistently being able to operate in a familiar environment certainly contributes to the user friendliness of a system where the user can manipulate parameters which he expects from his mental model and conventional signal processing training.

*"...Combine the presentation of text with graphics where applicable ...Keep graphics images clear, uncluttered, and not dense, thereby aiding memory retention..."*

The successful combination of text and graphics across the various Syter screens results in a working environment which does not overly tax the user in terms of processing the interface details before interacting with it. Also, as a result of combining text and graphics, the interface is recalled and remembered easily, and the user works in a familiar environment, free to pay close attention to the quality and nature of the sound output from the system.

*"...Choose the input device carefully, considering compatibility with system and user, accuracy within the system, efficiency and ease of use...."*

The use of a four button mouse as an input device to the Syter is the source of several sources of frustration for the user. The actual physical interface itself, requiring the user to select a button in terms of colour to effect an action or a reaction on the software interface is in itself straightforward. The problems begin to arise

when that colour coding becomes confusing and misleading. The colour coding for particular tasks is not consistent between screens, as we have seen in Section 8.2.3. The user has to constantly remind himself in each screen exactly what button will activate what action on the screen. Also, as a four button device, the mouse action is not comfortable, as the user will comfortably only press buttons using one or two, or at most three fingers on one hand, while he attempts to keep the mouse still at the same time. The employment of such four-button devices seems only to add confusion and frustration to an otherwise clear and concise interface. The screens themselves are sufficiently well laid out that single-button mouse procedures would control the parameters well. The logic in mapping mouse buttons to actions seems unnecessary, and one which in no way enhances the user friendliness of the system as a whole.

*"...Design a consistent well organised interface....Provide a clear uncluttered screen...Place warnings, data to be monitored and urgent messages in the top right screen quadrant, where users tend to recognise information more quickly. Place routine information not needing constant monitoring in the bottom right quadrant where users are less distracted.*

The Syter screens, as we can see in Figures 8.2 – 8.5 and Figures 8.7 and 8.8 are well organised and clearly laid out, with well-sized working areas for manipulating parameters. The details which the user need not continually process, such as the GRM trademark, are placed as recommended in the bottom right of the screen areas, and the workspace is clear and quickly absorbed by

the user. However, the system does not maintain consistency between screens, as the colour coding relating the functions which are mouse activated differs from one screen to another in the EGS environment.

*"...Keep colour usage to a minimum - recommended to employ up to 4 grey scales in a monochrome design, and up to 8 colours in a colour interface design...Reliable colours are red, blue and green, with purple and yellow as second choices, with use of colour held consistent across the interface..."*

As recommended the Syter system maintains colour choice across the interface, and employs the best colours with regard to user visual processing abilities. The only fault is again the issue of colour coding between screens for the mouse colour buttons.

*Always have the main menu available.*

Across the Syter interfaces, the menu of screens available within any environment is always clearly displayed across the bottom of the active screen, and is easily activated.

*"...The virtual software created interface should shield the user from the computations happening at lower levels within the system..."*

As a system which combines graphical control with command line input if the user decides to involve himself with instrumental design, the Syter presents a user friendly interface to the 'surface'

user, who uses only the graphical screens to manipulate sounds with instruments already available in the system library. However, for the user who decides to design instruments for himself, the system still maintains a working environment which is not overly complex or requires high level programming experience. The procedure for instrumental design, although it is referred to as programming by the designers of the Syter, is managed in a fashion of linking operators together on the command line. The user is still shielded from system specifics at levels close to the processor.

*"...Keep management of windows simple and fast..."*

The mouse control of the Syter screens is fast and straightforward, once the user has mapped colour coding of actions active on the screen to the mouse controller.

*"...Afford the user complete/complex error handling...Provide the user with informative feedback at all levels of task completion..."*

The undo functions within the Syter are comprehensively presented to the user, with select and annihilate being a very common combination of mouse clicks. However, the editing functions within the system are less than easily managed by the user, and the task of defining file sections for editing is very frustrating and cumbersome. Delicate and precise mouse pointer placement is carried out for the mixing and splicing functions by placing rulers at points within the file presentations on the EGS command screen. Screen precision and zoom facilities are not powerful enough to be

able to set edit points exactly first time, and the user has to redo the exercise several times before exact points are correctly selected.

*"...Provide visual portrayal of signals for analysis and decomposition, and manipulation..."*

The inclusion of the repérage screen as a point of reference for signal examination is very important in the Syter system, and answers the need for the user to be able to operate with a visual presentation of his mental model during signal processing work.

*"...Define and maintain a system balance between generality and strength..."*

The Syter system operates as a system which is general enough in its compositional model as to allow beginners to employ the system with a high degree of success. With the provision of public libraries, the user new to the system can achieve an action-reaction discourse in a very short period of time. With the extensive use of interpolation techniques, the new user can be encouraged in a creative experimental environment which quickly relates aural results which can be tracked on screen in an interactive fashion. While embracing this degree of generality for some users, the Syter also operates as a compositional specific system, allowing more experienced users to experiment with instrumental design, the results of which can be heard very quickly.

The Syter system embraces a compositional model which maps very closely to the conventional user's mental model of taking an instrument, causing an action 'upon' it, and getting a sonic result. The success of the action-reaction discourse of a composer using the Syter lies in the fact that the user mental model requires little or no translation by the user to effect a creative result.

# Chapter 9 Steim Composition and Performance Systems

## 9.1 Introduction to the Systems and 'The Steim[1] Ideal'.

The BigEye composition and performance system discussed in this chapter in Section 9.3 is one of the latest in a long line of systems developed in a studio environment which concentrates heavily upon the physical gestural content in music creation. As the institution founder of Steim, Michel Waisvisz (Krefeld, 1990, pg 28) tells us:

> "The way a sound is created and controlled has such an influence on its musical character that one can say that the method of translating the performer's gesture into sound is part of the compositional method."

The design philosophies behind the Midi Conductor[2] and the other performance interfaces designed by Waisvisz, including The Web[3], centre, in the majority, around the provision of direct timbral control and manipulation for the composer/performer. Unlike most of the other interfaces discussed in this research, all of the composition tools developed and built at Steim have the integrated identity of being simultaneously a performance instrument.

The unique instrumental interfaces built at Steim are based on the personal musical needs of the composer, addressed by a small team of designers and programmers as an individual project. This very unusual one-to-one relationship between

---

[1] Steim Studio Amsterdam exists as an electronic music studio which is dedicated to live performance.

[2] Midi Conductor is a performance Midi controller interface.

[3] The Web is a performance Midi controller designed by Waiswisz, which controls Midi data by manipulating wires stretched across a wooden frame.

composer and design team is an attempt to resolve the technical demands of any compositional notion. This has resulted in the creation of several very varied physical interfaces as performance instruments which remain unique in both conception and operation in the field of modern interface development.

To date, several integrated performance-composition instruments have been built at Steim. In this chapter, a number of these will be introduced and discussed, including the Lick Machine, the Midi Conductor, the SensorLab, and the BigEye video to Midi interpreter. Unlike other systems reports which have concentrated on one central interface in a system, it seems useful to discuss the set of instrumental, performance and composition interfaces which the author accessed during residential studies at Steim[4]. In this way, the range of interface types, both physical and virtual, which will be available for comment and analysis within this thesis will be extended. The ideal that is Steim is summed up by Waisvisz, (Krefeld 1990, page 31):

> "We want to pursue our original policy of creating extremely personal musical instruments. We believe we contribute more to the possible future quality of electronic music by allowing individuals to make an entirely personal statement, instead of pretending to serve everyone and coming out with dull, middle of the road stuff that makes everyone sound the same. Apart from that, the industry still believes in high technology while we think that appropriate technology is more reliable and easier to modify by the composers themselves."

[4] The author resided at Steim for a period in 1994, 1995 and also in 1996.

This idea is one that will be examined in detail in later discussions about the future design and development realities of musician interfaces and working environments.

## 9.2 Systems Details and Mode of Operation

### 9.2.1 The Lick Machine Software Package

Programmed originally to run on the Atari computer as the host and now ported to the Macintosh range of computers, the Lick Machine software affords the composer user a fully adaptable and controllable Midi environment for both composition and performance without the need of complex computer programming knowledge. Operating as a real-time composition arranger and manipulator of Midi data, the Lick Machine can control short sequences of Midi data, known in the program as 'licks', and can manipulate the data within these licks in real-time in response to pre-recorded commands or to incoming commands from an attached Midi interface.

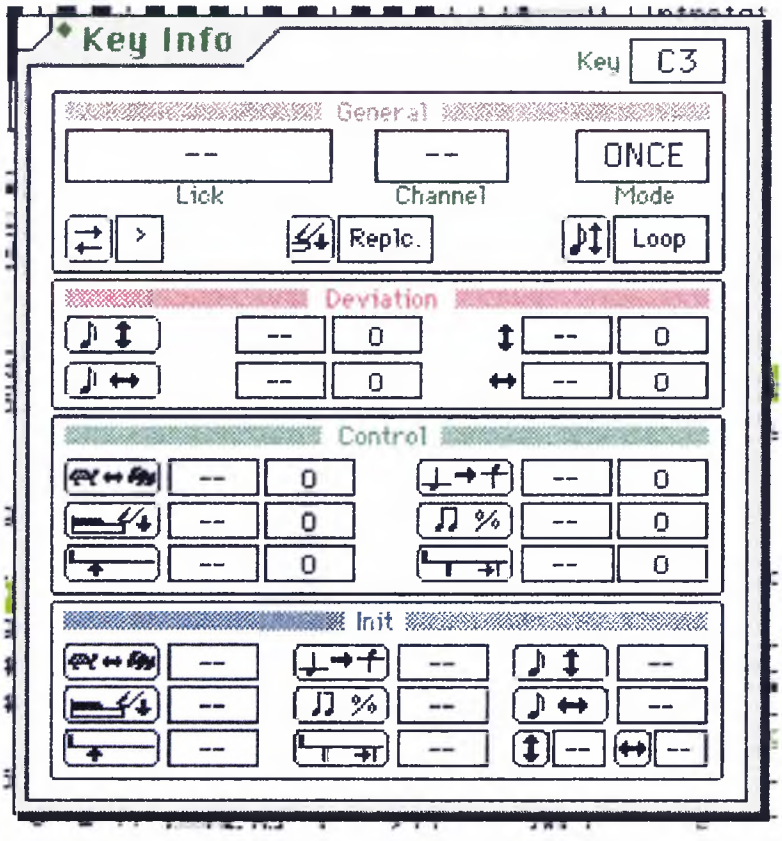


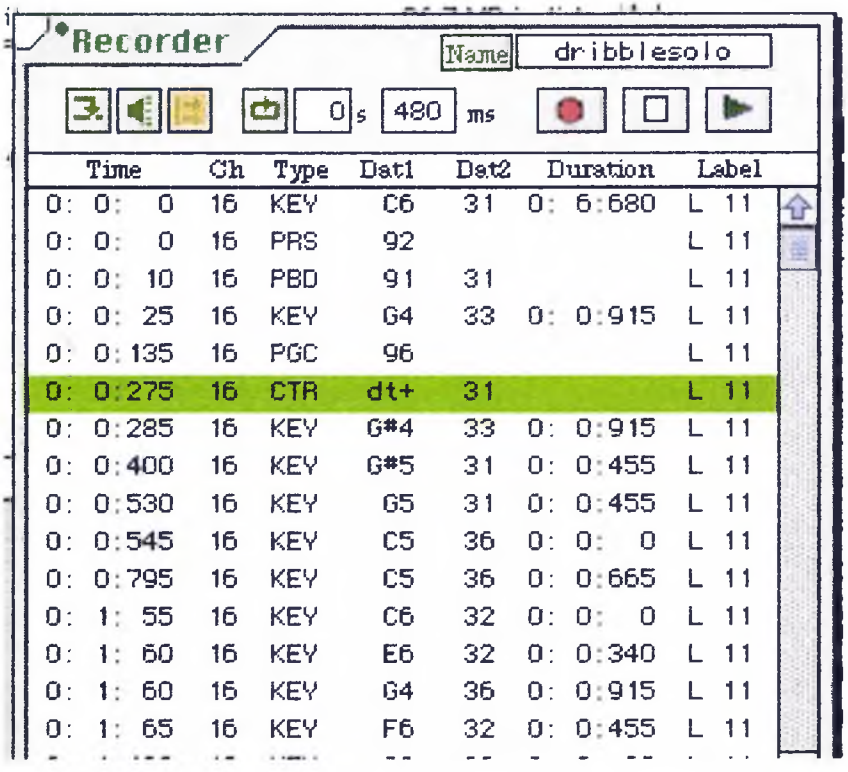

Figure 9.1 Lick Machine Key Info Window

All of the user interfacing screens in the Lick Machine software are graphical in design, with an interesting approach to the use of common graphical symbols to represent musical parameters in Midi. Rather than using words to represent key press, pitch range, or tempo variations, the designers have incorporated coloured graphical symbols as visual triggers. Figure 9.1 above shows the individual lick control screen *Key Info,* relating to a lick assigned to one key, in this case middle C, which relates Midi information to the user in graphics–based icons relating to:

- the lick number
- lick mode
- channel
- control status
- initialisation status for the specific lick

The incoming data to trigger this lick may be sourced from any Midi interface, including the Midi Conductor physical interface, discussed in section 9.2.2 below.

Together with using graphical symbols to replace text on user interface screens, the Lick Machine also allows the user to employ a wide range of keyboard shortcuts to cut down on *mouse move-position-click* sequences on the screen. The recorder window seen in Figure 9.2 which provides the user with a real-time feedback of all Midi events as they occur, allows the user to speed up operation and use of the functions within the screen by providing keyboard shortcuts for several of the functions displayed graphically.




| Time | Ch | Type | Dat1 | Dat2 | Duration | Label |
|---|---|---|---|---|---|---|
| 0: 0: 0 | 16 | KEY | C6 | 31 | 0: 6:680 | L 11 |
| 0: 0: 0 | 16 | PRS | 92 | | | L 11 |
| 0: 0: 10 | 16 | PBD | 91 | 31 | | L 11 |
| 0: 0: 25 | 16 | KEY | G4 | 33 | 0: 0:915 | L 11 |
| 0: 0:135 | 16 | PGC | 96 | | | L 11 |
| 0: 0:275 | 16 | CTR | dt+ | 31 | | L 11 |
| 0: 0:285 | 16 | KEY | G#4 | 33 | 0: 0:915 | L 11 |
| 0: 0:400 | 16 | KEY | G#5 | 31 | 0: 0:455 | L 11 |
| 0: 0:530 | 16 | KEY | G5 | 31 | 0: 0:455 | L 11 |
| 0: 0:545 | 16 | KEY | C5 | 36 | 0: 0: 0 | L 11 |
| 0: 0:795 | 16 | KEY | C5 | 36 | 0: 0:665 | L 11 |
| 0: 1: 55 | 16 | KEY | C6 | 32 | 0: 0: 0 | L 11 |
| 0: 1: 60 | 16 | KEY | E6 | 32 | 0: 0:340 | L 11 |
| 0: 1: 60 | 16 | KEY | G4 | 36 | 0: 0:915 | L 11 |
| 0: 1: 65 | 16 | KEY | F6 | 32 | 0: 0:455 | L 11 |

Figure 9.2 Lick Machine Recorder Window

The red *record* button in the recorder window may be activated by pressing the tab key on the computer keyboard, while the return key will activate *stop* The spacebar will activate *playback* of any lick recorded by the combination of record and stop.

### 9.2.2 Midi Conductor.

The Midi Conductor is a physical performance interface based on hand mounted sensors, as seen in Figure 9.3. As the current 'version' of the original Hands[5] instrument, when coupled with the Lick Machine software the Midi Conductor becomes a controller which allows the composer/performer to manipulate Midi data in real-time through the use of physical gesture.

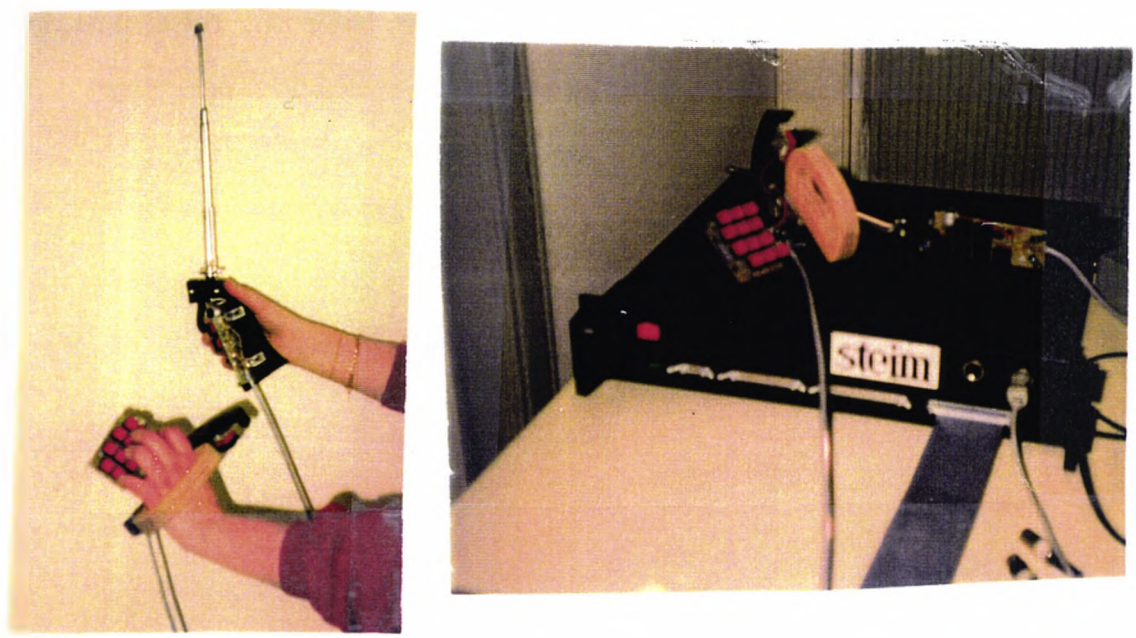



Figure 9.3 Midi Conductor Physical Gestural Controller

[5] Performances and work with The Hands, (Krefeld, 1990) which were born from Waisvisz very strong interest in texture as opposed to time based events, resulted in the creation of the Midi Conductor, a more "generic version" of The Hands, which could be played by anyone, employing basically the same technology but being more adaptable and easier to play.

Feeling to the user like a pair of electronic 'gloves' the Midi Conductor left and right 'hands' are actually fitted for each individual user personally by a Steim technician for comfort and complete control.

The user operates the red or black click switches lying under each finger position, as seen in Figure 9.3, in relation to Midi control parameters within the Lick Machine. The left hand controller contains the black switches which can be mapped to select the Midi parameter to be controlled, while the actual parameters would then be altered by the red switches on the right hand. Also specific pre-composed licks can be assigned to the red switches on the right hand controller. Some of the versions of the Midi Conductor have ultra-sonic controllers built in, which make the difference in distance between the two controllers in performance a controllable parameter, affecting modulation values, or any Midi parameter the user has set this controller to affect. Also some versions have an aerial included, which can be extended in space and used to create pitch bend and vibrato effects on Midi pitch data.

### 9.2.3 SensorLab Analog-to-Midi Interface

Steim has developed the SensorLab in order to accommodate composers and designers who have needed to convert a wide range of signal sources to conventional Midi format. As the designers of the SensorLab (1993 pg. 2) tell us:

"A variety of projects at Steim in recent years have led to the emergence of the SensorLab design. These included entirely novel 'instruments' such as The Hands, the Midi Conductor and The Web of Michel Waisvisz, an ultrasound violin bow, a Midi controlled motorized camera mount and several Midi conversions of conventional instruments such as a Bass recorder, a concertina and a Melodica. The variety of sensors needed to complete these projects (switches, potentiometers, pressure pads and ultra-sound devices) forced our design towards the accommodation of a wide range of real world signals. No doubt this should allow us to make new instruments and interfaces for some time to come."

Operating as a customisable micro-computer which may be removed from the host computer, all programmed by the user as required, the SensorLab is a very flexible tool useful for many diverse applications.


Figure 9.4 SensorLab Analog-to-Midi Convertor

As seen in Figure 9.4 above the SensorLab is a compact controller which will interface to ideally 'any type' of real world signal generator and convert the signal received to Midi format which may subsequently be manipulated in any required fashion. In order to facilitate manipulation of the incoming signal when converted to Midi, the SensorLab has a resident interpreter, known as Spider, which may be programmed by the user when the system is linked to a host computer to run the Spider language interpreter and compiler. The workstation configuration for user programming of the SensorLab is shown in Figure 9.5 below.

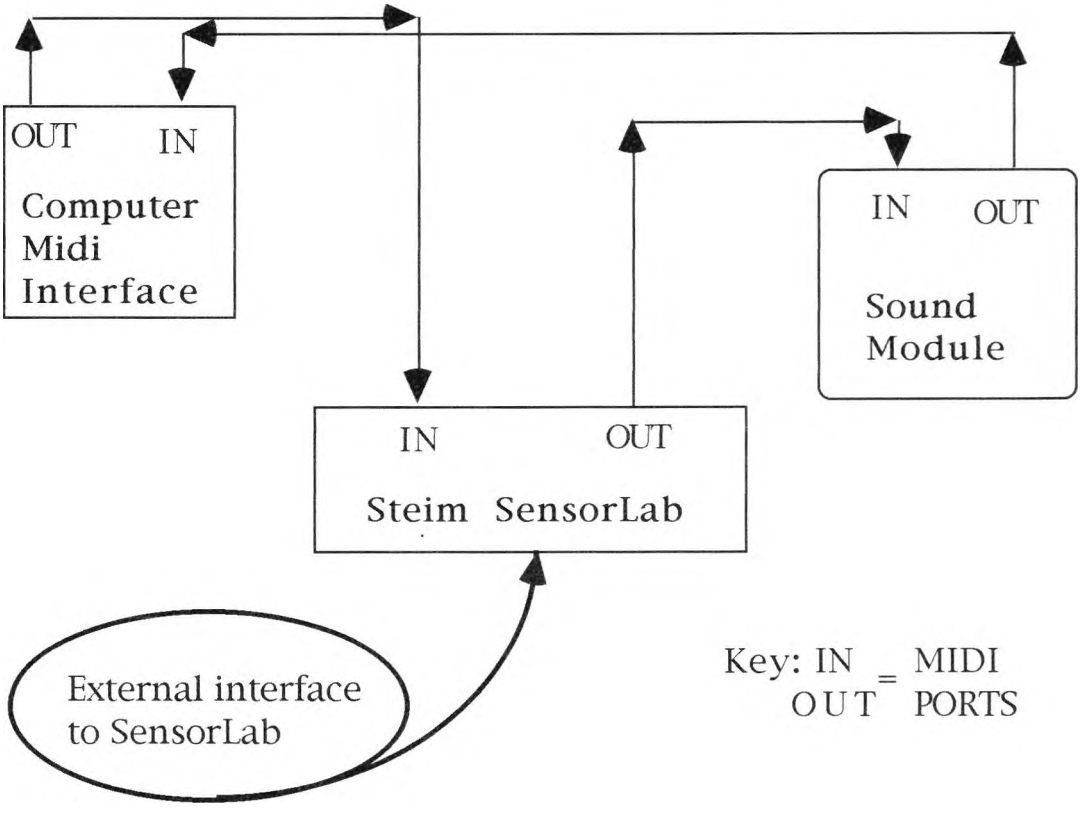



Figure 9.5 Physical Set-up for SensorLab Programming

This configuration will allow the user to write, compile and send code to the SensorLab. To facilitate testing user code on the actual instrument, or whatever interface is connected to the SensorLab, the user can check the Midi data coming out of the SensorLab on Spiders' built-in Midi monitor. Employing a host computer to run the Spider programming environment will allow the user to program Midi maps to interpret the incoming signal to the SensorLab in a desired way. As the unit documentation tells us:

> "Performance characteristics can be programmed via SPIDER a user instrument configuration and Midi mapping language which currently runs on host Ataris, Macintoshes and DOS PC's also communicating with the SensorLab via Midi."

As an 'anything'-to-Midi convertor, the SensorLab accommodates 32 analog input channels, to trigger sections of Spider code to produce Midi output.

Also accommodated within the SensorLab are 3 ultrasound receivers and 2 ultrasound transmitters affording a total of 6 ultrasound control variables. The relationship between the sensor inputs which may arrive from, for example, piezo transducer triggers or an ultrasound combination of transmitters and receivers, and the Midi data created in response by the SensorLab is defined by the configuration file which the user will create in the Spider language. Once the configuration file has been downloaded from the host computer to the SensorLab, the SensorLab may be disconnected from the host and used as a stand

alone signal convertor, converting the incoming signal to Midi data in the fashion predefined in Spider by the user.

> The SensorLab, operating as a
>
> "...'real-world to Midi' hardware convertor
>
> and software toolkit..." (Anderton, 1994, pg. 61)

provides the user with the vehicle with which to express any experimental controller idea.

### 9.2.4
### BigEye Video-Midi Conversion and Control Environment.

Developed at a time when the composer's work medium is becoming increasingly infused with multi-media artistic techniques and performance opportunities, the BigEye computer program developed at Steim by Tom Demeyer utilises video input as the 'trigger' source for Midi data control and manipulation.

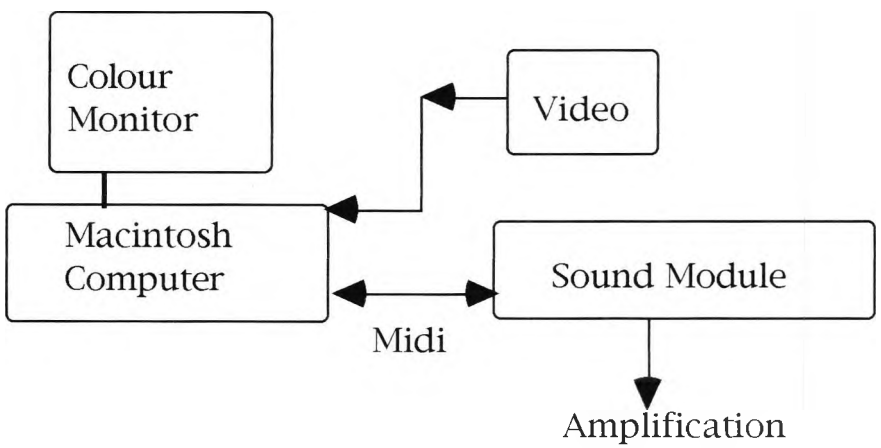



Figure 9.6 BigEye System Layout.

The basic system, seen in Figure 9.6, operates on the Macintosh computer platform, and requires a Macintosh with video input, or a compatible video digitizer. Image tracking or colour shapes in an animation sequence may be used to trigger and control Midi data.

Capturing a real-time moving video image, the system is fully configurable in Midi common format, to enable the user to home in on certain selected parts of the image to act as Midi triggers. Figure 9.7 below displays the most basic channel screen before the user begins to select objects of interest, by going to an image filter window.


Figure 9.7 BigEye Main Channel Screen

Once the user has settled on the basic video image in the channel window, the next step is to go to an image filtering window, in order to tell the program which objects are of tracking importance to the user within an overall image screen. Object tracking in the BigEye environment is carried out on the basis of colour and pixel condition. As Demeyer (1995, pg 6) puts it:

"In the end, when BigEye attempts to track objects, the only thing which counts is which pixels in the image are 'on' and which are 'off'. Off pixels being pixels which are absolutely black and 'on' pixels holding any value other than absolutely black. Colour or brightness information is no longer relevant at that stage."

The five filters which the system provides are:

- Colour Table

defines which colours are allowed to be processed by the program, and which colours are to be ignored by the system;

- Threshold

used to single out objects due to their brightness values, to concentrate on an object which is brighter than another in the same image;

- Difference

allows the user the possibility to select only moving objects from an image of high complexity;

- Mask

operates exactly in reverse to the active region idea; masks out an area of the image completely;

- Despeckle

removes speckle resolution problems from the image, therefore getting rid of visual 'noise'.

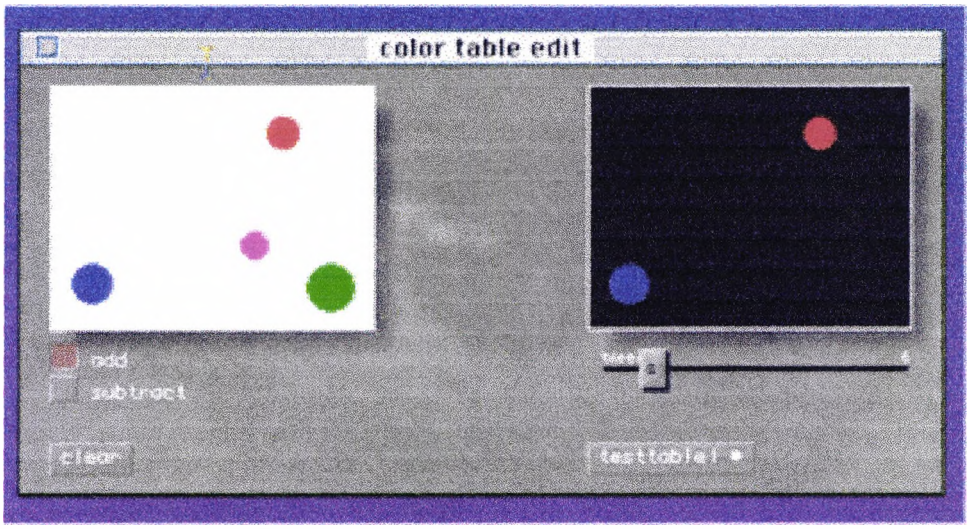



Figure 9.8 BigEye Image Filter Screen

Once the user has selected the object or objects of interest in an image within one of the image filter screens as seen in Figure 9.8 above, the next stage is to define an area within which this object will be tracked and cause Midi triggering to happen. These areas are known as 'hot zones' or active regions. As the system manual tells us (Demeyer, 1995, page 2):

> "The user configures the program to extract objects of interest, based on colour and size. These objects are tracked (up to 16 simultaneously), and their position is checked against a user-defined series of 'hot zones'. These 'hot zones' are drawn by the user and are grouped in 'screens'."

The areas selected as active regions for Midi event triggering may be drawn by the user in several ways including free-hand, rectangular box format and also specifying a series of columns or rows. Figure 9.9 illustrates a region definition screen with rows and columns drawn to specify active regions.

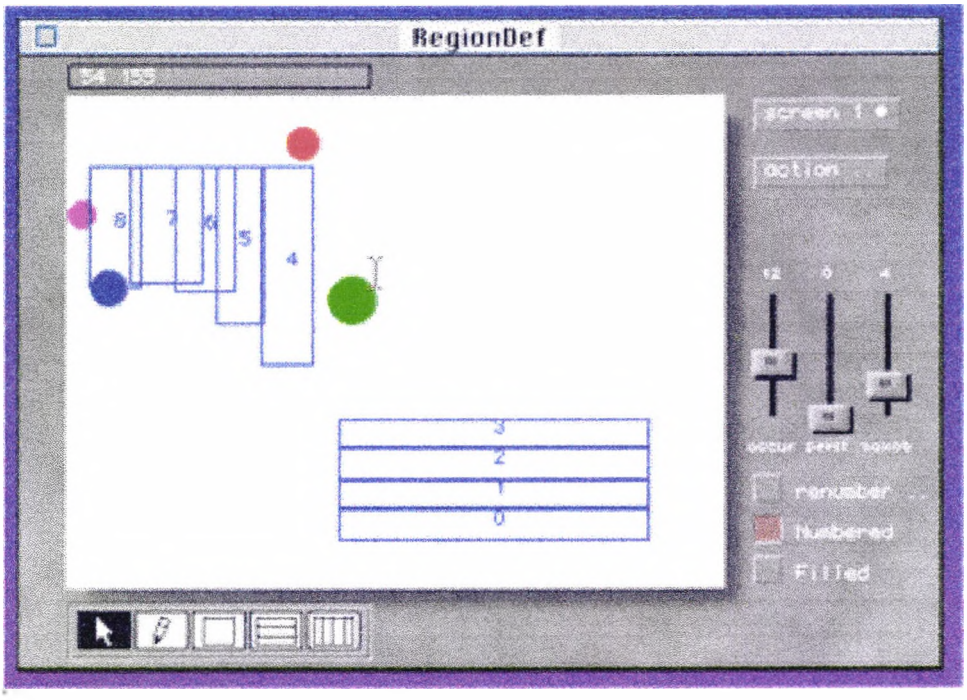



Figure 9.9 BigEye Region Definition Screen

The idea is that the program will track the movement of the objects of interest to the user upon arrival into the region, during its 'journey' through the region and also its departure from the region. These three sets of movement tracking, into, inside and out of a static selected region by an object to be tracked are then assigned Midi information of some description. This means that, for example, an image of a ball bouncing across the video image screen can result in a note being turned on as the ball image entered a selected tracking region, the *new* parameter, that note being pitchbent while moving within that active region, the *move* parameter, and that Midi note being turned off as the moving ball image leaves the tracking region within the overall video image screen, the *end* parameter.

At present, BigEye allows the user to employ not only video image as a trackable source of movement, but also supports

animation and certain movie image formats as possible input image sources.

Once the image has been set and the regions selected by the user, the task of actually assigning Midi information and commands to the three points of activity (*into* the region, *while in* the region, and *out of* the region) must be handled. The user has a choice of two methods of specifying Midi messages to these three tracking stages.

In a very straightforward and technically 'musical' fashion, the user can opt to assign straightforward Midi messages to each of the three stages of tracking. As each region is defined as being of tracking interest for a particular object within the overall video image, the user assigns Midi messages in the form of 'actions' for each of the three tracking stages in turn. The list of actions that the user has to choose from in this mode are in General Common Midi format commands as listed in Figure 9.10. The exception to a common Midi list here is the inclusion of the screen parameter, which allows the user to specify in Midi format the screen number to be used.

| | STATUS | DATA BYTE | DATA BYTE | |
|---|---|---|---|---|
| NOTE OFF | Channel No | Note No. | Velocity | |
| NOTE ON | Channel No | Note No. | Velocity | |
| POLY KEY | Channel No | Note No | Value | |
| CONTROL | Channel No | Control No | Value | |
| PROGRAM | Channel No | Program | | |
| PRESSURE | Channel No | Pressure | | |
| PITCH BEND | Channel No | Low value | High Value | |
| NOTE | Channel No | Note No | Velocity | Duration |
| SCREEN | Screen No | | | |

Figure 9.10 Midi Action Messages in BigEye

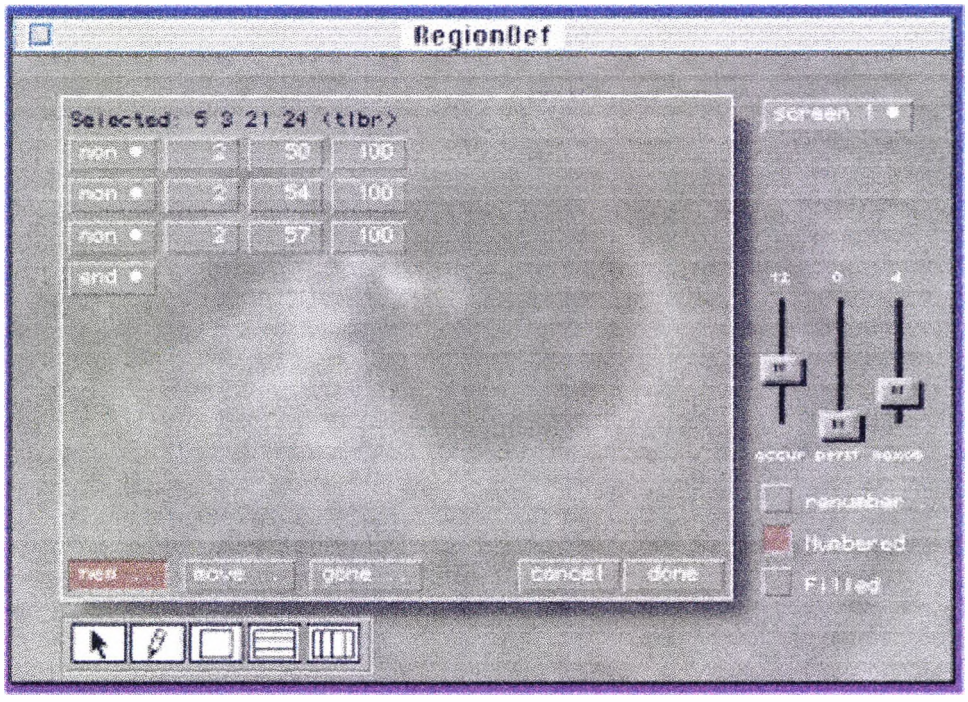



Figure 9.11 Action Definition for a Region in BigEye

In Figure 9.11 above a region definition screen in the action definition stage displays the encoding of a Midi message to a *new* action in the tracking system.

The process of assigning a Midi action to a tracked event in the image screen is as simple in this mode as drawing up a list of up to 10 Midi messages which will be triggered when the corresponding physical movement is tracked in the image screen. Certain object attributes are hard wired for use when this mode of Midi message mapping is used, relating the Midi messages to the objects' physical space in the visual image. These include parameters as listed in Figure 9.12 below:

| | |
|---|---|
| X and Y: | absolute co-ordinates of the object being tracked in the image |
| (X) & (Y): | co-ordinates relative to the region that the object is being tracked in. Object extreme left of region (X) = 0, object extreme right, (X) = 127 |
| vX & vY: | component velocities. up = vY is less than 0, left = vX is less than 0 |
| V: | velocity of the object being tracked, dynamically scaled to Midi values. |
| aX & aY: | object accelerations, differences between current and previous object velocity. |
| sze: | size of the object being tracked into, through and out of the active region. |
| obnm: | object no. seen in the channel window when user asked to see objects |
| lon: | last on message, taking the information from a note stack |
| rgnm: | region number of the region containing the triggering object. |

Figure 9.12 BigEye Object Tracking Parameter Listing

As seen in Figure 9.12, the process of assigning Midi events to physically tracked movements is a musically straightforward process for composers and performers familiar with the industry standard Midi protocol.

Once the user has defined the actions to be assigned to the tracking movement for the selected object, moving back to the channel screen will allow the sonic results to be heard. Any action lists can be edited and altered at will by entering the region action screen again at any time. Using the animation sequence provided with the BigEye as an example, the series of figures following detail a typical series of tasks to achieve Midi data triggering by tracked movement on the screen.

The animation sequence (Figure 9.14) consists of a set of coloured balls moving in various patterns within the image

screen. The sketch below (Figure 9.13) details the path of each coloured ball within the main image screen in the animation sequence.

Key:` Bottom small left and right large in Middle = Green Ball
Left, Swelling in and out as up and down = Blue Ball
Full circle, fattest at centre top and bottom = Red Ball
Completely random thin line = Purple Ball

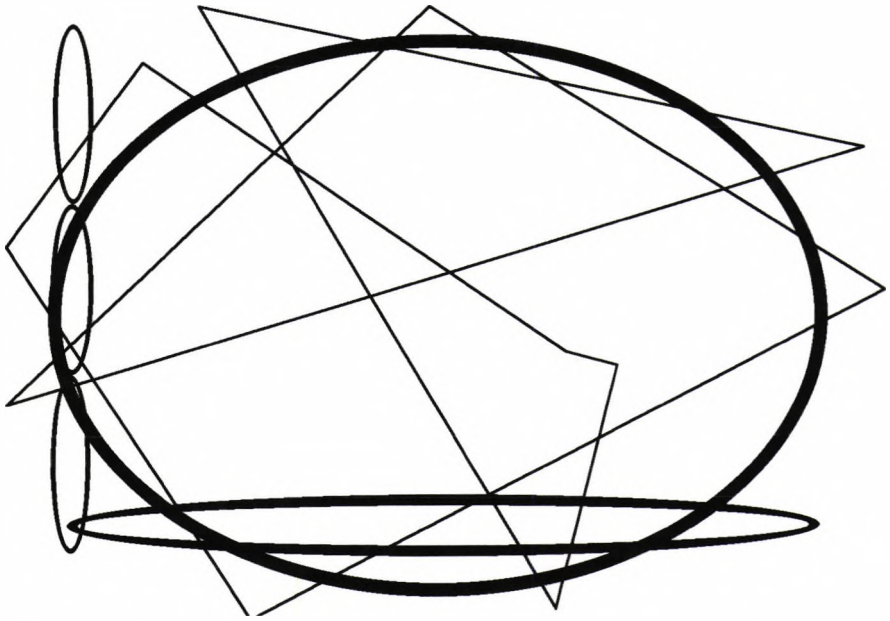


Figure 9.13 Trajectory Path for Coloured Ball Sequence

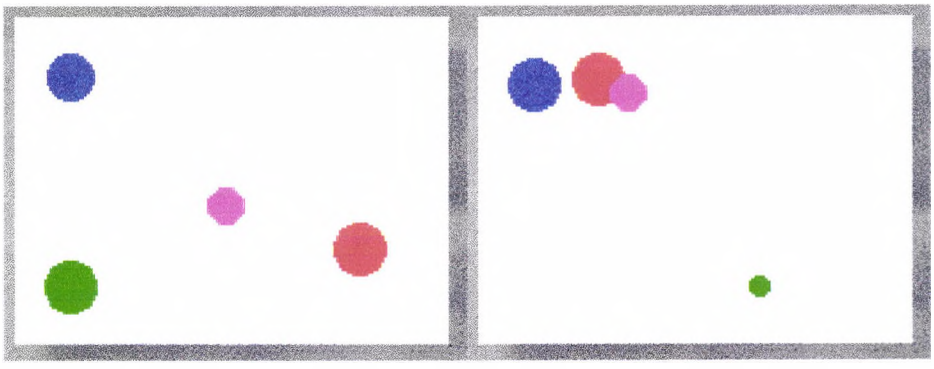


Figure 9.14 Movement Pattern of Coloured Ball Sequence

Setting a region definition area as detailed in the box numbered 1 in the Region Definition Screen seen in Figure 9.15, a

corresponding action list tracking the red ball as it circles its way around the active screen is detailed in Figure 9.16.

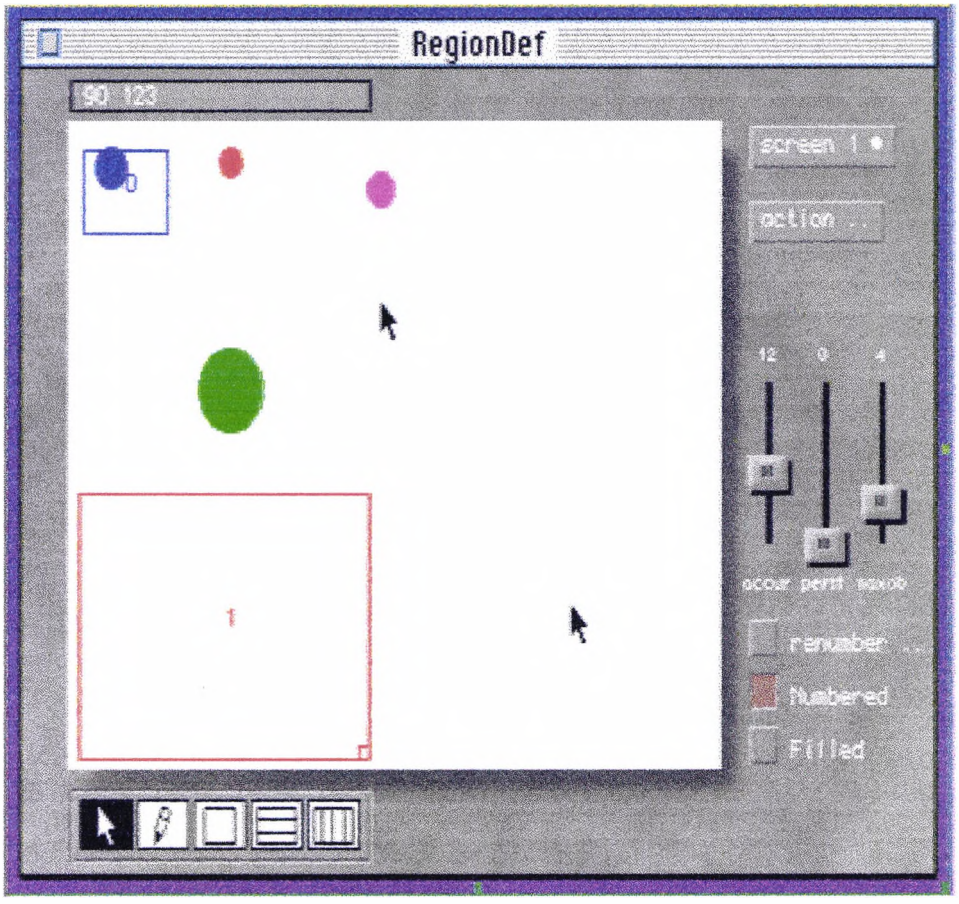



Figure 9.15 Region Definition Screen for Tracking Red Ball (rectangle number 1).

This action list describes the Midi data to be triggered for the three stages of tracking (entering the region (*new*), moving within the region, (*move*) and leaving the region (*gone*))

```
Action List for New
        non  1    80   100
        end
Action List for Move
        pbd  1    0    X
        end
Action List for End
        nof  1    80   0

        end
```

Figure 9.16 Action lists for tracking Red Ball

The action is set in Figure 9.16 above for a Midi note at pitch 80 to be triggered as the red ball enters the active area. As the ball moves through the area a pitch bend will occur, the amount of pitch bend being determined by the X co-ordinate of the red ball position. As the ball leaves the area of tracking, a note off will occur. Every action listing must be completed by the *end* command.

In a more complex working fashion, the user may employ a mode known as scripting, whereby in a programming environment, the user has much more control over the whole object tracking process in terms of what midi messages are to be triggered, and has control over a series of structured commands being passed to the BigEye image processor for Midi event creation and control. Figure 9.17 below shows as an example one section of a typical BigEye script, a procedure assigning an index table to Midi note values.

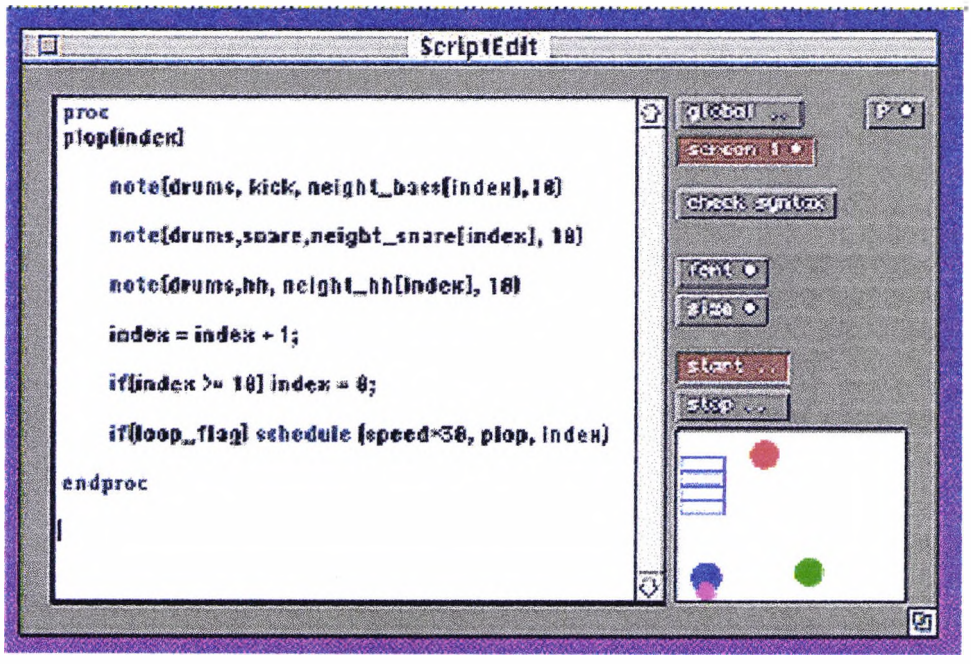



Fig. 9.17 BigEye Scripting Environment.

The user of BigEye who opts to work with scripting to improve and advance their control of the program would need basic computer programming knowledge. As the system designer tells us (Demeyer, 1995, pg. 24):

> "The scripting system is a programming language, and users inexperienced in computer programming will have to invest time on order to be able to use the BigEye scripting system to its full potential."

As a system which is both graphical in its control and real-time in its user feedback, the BigEye has interesting potential in the growing field of multi-media arts collaborations.

## 9.3 Systems Synopsis, Analysis and Evaluations.

To facilitate analysis and evaluation of the Steim systems introduced above, the Midi Conductor, Lick Machine and SensorLab are grouped together as an operational system, referred to as the Steim Midi system, with BigEye being evaluated as a stand alone interface.

### 9.3.1 Systems Synopsis Grids

The analysis of the Steim Midi system is based on several residencies at the Steim studios, during which time the author employed the Lick Machine, Midi Conductor and SensorLab as an integrated system configuration.

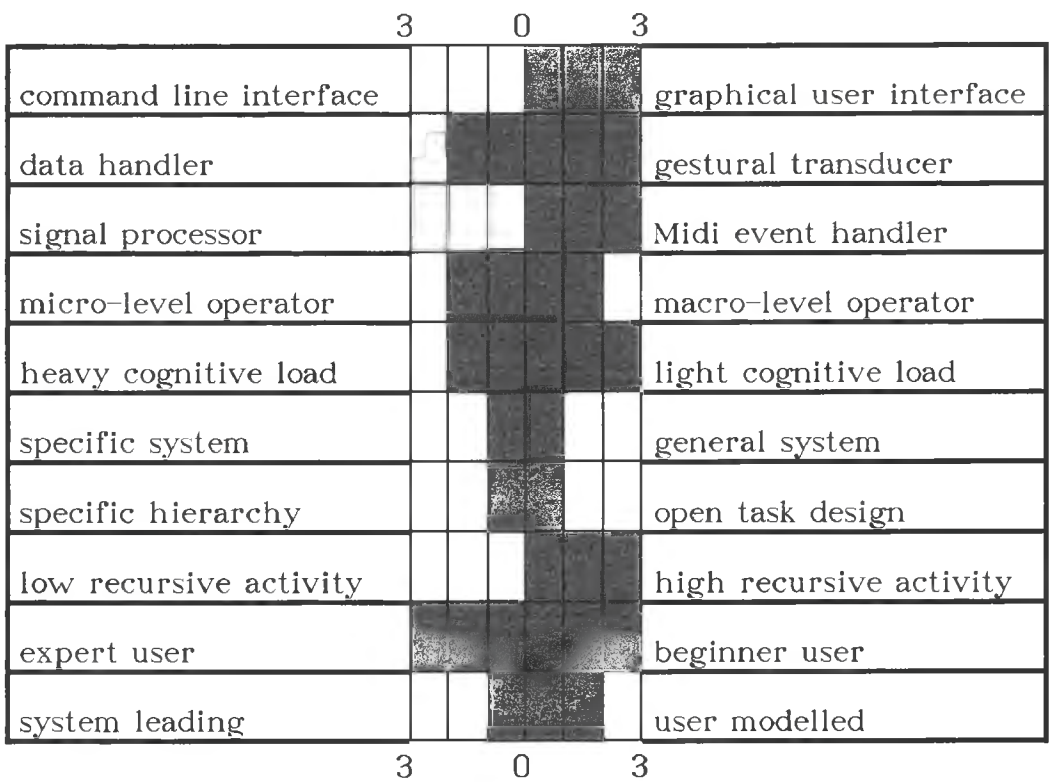



Figure 9.18 Steim Midi System Synopsis Grid

The Steim Midi system is summarised in the system synopsis grid in Figure 9.18 above. The system employs a graphical user interface, handling data arriving from a gestural physical interface. As a Midi data handler and manipulator, the system gives the user the opportunity to work at both micro-[1] and macro-levels, specifying single data bytes for Midi instructions, or handling 'licks', sets of Midi data for performance triggering. As a graphical interface, the Lick Machine software puts a light cognitive load on the user, but if the user employs the Spider language to re-configure the SensorLab, a good deal of concentration and programming skill is required. The physical operation of the Midi Conductor is straightforward only once the user has cognitively mapped the correct switches to the Midi parameter being controlled.

As a system which affords the user complex control of Midi data, no specific compositional approach or inflexible task hierarchy is imposed on the user, who even has the opportunity to compose in space if desired with the Midi Conductor. With the graphical user interface that the Lick Machine employs, a high level of recursive activity is possible with only single parameters needing to be altered in dialogue boxes on the screen.

As a Midi controlling system, all levels of users are equally supported, with the beginner getting immediate reaction in a gestural environment, and the more expert users having the opportunity to reconfigure the switching interfaces on the Midi Conductor via the SensorLab and Spider combinations. As far as

[1] While some 'single bytes' may trigger events (macro objects), many such as velocity and some controller data operate at the micro sound level.

user modelling is concerned, the entire system is modelled very strongly on the users expectations as far as employing Midi common format, and there is rarely a need to learn system specific task models, with the exception being the employment of the Spider language to re-configure the SensorLab, although this task itself is mapped very closely to conventional C programming procedures.

The analysis and evaluation of the BigEye system is based on work carried out by the author during two further residencies at the Steim studios. During the first visit, the BigEye was still at design stage and by the next visit the program had been officially released. The system synopsis grid in Figure 9.19 below summarises the details of the BigEye video-to-Midi system.

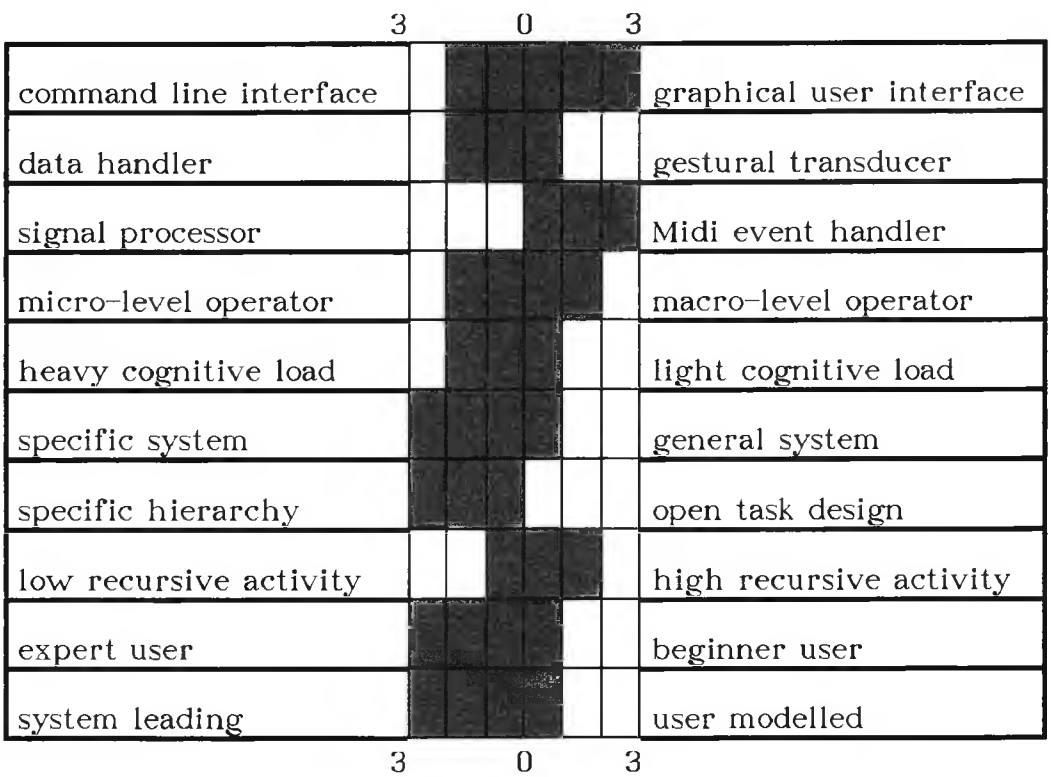



Figure 9.19 BigEye System Synopsis Grid

As a system which tracks objects on an active screen to trigger Midi events, the BigEye operates to a large extent in a graphical environment, although scripting for Midi events is carried out in a programming manner. The system is a gesture transducer only insofar as it tracks moving images on the screen, and all other input to the system is achieved by data input to dialogue boxes and Midi scripts, with a syntax checker built in. Completely a Midi event handler, the BigEye allows the user to specify the same range of macro– to micro–level Midi data as the Midi system described above.

As a system which requires the user to specify thresholds, masking filters, and to align channels of visual Midi tracking regions with scripts allocating Midi event data, the BigEye can place a large amount of cognitive stress on the user. Being a specific system relating to moving image triggering Midi data, the BigEye employs a very specific compositional hierarchy, not immediately approachable by a beginner. Composers conversant with Midi protocol and also C programming procedures are well accommodated by the system, but even they have to spend time modelling the task structure of filter setting channels and region definitions, which does not come naturally to the average composer.

Designed as a Midi data handler, the system operates with a high level of recursive activity possible, and users are consequently not afraid to experiment with filter settings and definitions for tracking regions. In a manner similar to the Steim

Midi system detailed above, the BigEye is user modelled insofar as standard Midi protocol is employed and modelled on users expectations of Midi use, but the composer has some mapping to do in order to comprehend the system task model of interacting filters, regions of definition and active areas on serial sets of active screens.

### 9.3.2 Analysis and Evaluation.

Guidelines from Chapter 5 are introduced in italics as start points for systems analysis and evaluations. Analysis and evaluations of both the Steim Midi system as described above and the BigEye system are introduced immediately following the appropriate guideline.

*"...Use natural sentences and natural language words....employ user models for language syntax and medium...Use consistent syntax across the interface..."*

With both the Lick Machine and the BigEye system, users operate within a familiar and very consistent syntactical environment so far as Midi protocol is concerned. One major problem with the BigEye system is the confusion concerning double meaning for the word channel. In the system, the user will identify channel with regard to Midi channel assignment. However, the word has another role, which becomes very confusing. As the user sets an image filter to a particular specification, this setting is known as a channel

setting. Channels in BigEye refer to different settings of an image by various selected filters. This double use of what a composer would regard as conventional syntax is very misleading, and causes unnecessary confusion. Although this wording may relate to filter settings in light analysis work, perhaps in this application a less leading name for filter settings on an image would have been more appropriate.

*"...Combine the presentation of text with graphics where applicable...keep graphics images clear, uncluttered, and not dense, thereby aiding memory retention....design a consistent well organised interface...always have the main menu available..."*

Within both the Lick Machine and the BigEye software interfaces, text and graphics are combined at all stages on each, resulting in interfaces which are immediately approachable and easily processed by the user visually. In both packages, the main menu and other windows are available at the touch of a mouse button in the drop-down menu bar at the top of the screen. All images are clear, and the working area is uncluttered and pleasant to the eye to work with. Screen designs are consistent throughout each package, and the user quickly memorises screen details as they are presented as far as possible in combination with suitable graphics.

Within the Lick Machine in particular, icons are used to excellent effect, making what is a powerful Midi data processor look very simple and approachable. One problem arises with the use of icons in this package however, when the user has to define the meaning of some icons before he can use them. As we can see

in Figure 9.20 below, in a selection of Lick Machine icons, some of the graphical representations are immediately self explanatory. Equally, some are less than immediately obvious with regard to what they represent. When the user has to 'translate' a graphical symbol in order to use it, then the symbol is somewhat redundant, and perhaps a word might serve a better purpose, or a more refined and specific symbol.


Figure 9.20 Lick Machine Icons Selection

In the Lick Machine, user conventions and expectations with regard to graphical symbols and their meaning are relied upon heavily. To a large extent, the use of graphical icons is very successful, and their meaning to the user is immediately apparent. Take for example the traditional icons for play, stop and record as seen above. Also, the icons for speed indications (mapped as a tortoise to a hare), and also the key pressure icon (mapped as a finger on a piano key) are very easily identified by the user. However, a less straightforward mapping occurs for the other symbols from the Recorder screen, seen in the top row of symbols

in the diagram above. The user is not immediately aware of the meaning of the green and yellow symbol with two arrows, which relates to a 'play from' cursor position, or the insert icon, the green arrow pointing into a broken line. Careful user modelling should always be carried out across a broad spectrum of users before symbols are adopted for a systems interface.

*"...Choose the input device carefully, considering compatibility with system and user, accuracy within the system, efficiency and ease of use..."*

As a physical controller, the Midi Conductor interface is at first foreign and seemingly complicated to the user. However, once the mental model of a glove controller for both hands has been accepted by the composer, the flexibility of the Midi Conductor, doubling as both a gesture transducer and also a performance interface becomes apparent.

The mouse as an input device for both the Lick Machine and the BigEye is suitably based on contemporary user expectations. However, the use of a Midi keyboard for input to the Lick Machine could perhaps be explored as a more flexible input device.

*"...Simplicity should be aimed for..."*

With the Lick Machine software, the composer has a very powerful and open-ended Midi data controller. If the user has a firm basic understanding of the Midi protocol, the Lick Machine is a very

simple control interface for this medium, with its reliance on graphical symbols and simple dialogue box data input style.

*"...Keep colour usage to a minimum – recommended to employ up to 4 grey scales in a monochrome design, and up to 8 colours in a colour interface design..."*

With both the Lick Machine and the BigEye system employing colour, the user is presented with interfaces which are made more accessible and more easily memorised with consistent colours used across each screen in both packages, and colour scaling being used sympathetically within each system. For example, in the BigEye system, when the user may be overloaded with colour already in the input image, all scripting is carried out in greyscale, and other inputs which are not directly on the image screen itself are greyed also. The Lick machine, with extensive use of icons retains a consistent colour code throughout the system, and again greyscales all data input and presentation to prevent visual overload.

*"...The virtual software created interface should shield the user from the computations happening at lower levels within the system..."*

Within both the BigEye and the Lick Machine, the user is defining Midi data, and as a result is continually working in a user-modelled and musically conventional fashion, not computationally intensive. However, the Spider language which is used to re-configure the SensorLab is at a lower level of processing, and is not approachable by the average musician who would not have

programming experience. Also, the scripting method for providing Midi data in BigEye presents the user with a computationally intensive communication format with the system. Some form of modular linking language format might leave the system accessible on all levels to composers who feel not completely at ease with programming scripts.

*"...Afford the user complete/complex error handling...Provide a fast retrace option in the system..."*

Both the BigEye and the Lick Machine have very comprehensive editing facilities and allow the user to retrace his steps with no difficulty. One very interesting and invaluable feature in the BigEye is the inclusion of a syntax checker for scripting, which helps the user refine and correct scripts before compiling them. This means that the user will not be afraid to experiment with strings of input in the script as he learns the system in an atmosphere of trial and error.

*"...Strike a balance between a system which is a gesture transducer and one which is an information handler..."*

In the system set-up referred to in this chapter as the Steim Midi system, the user has the opportunity to work with a hardware and software configuration which is an interesting balance between a gestural transducer and a information data handler. Using a physical interface in space the user can control Midi data across the spectrum from the micro- to the macro-levels.

The Steim ideal, summarised as an attempt to provide not "high technology" but "appropriate technology"[7] links composers to electronic devices as a means to an individual creative end, and not in an attempt to create industry standard interfaces. The composer is embraced in the main for all he is capable of achieving creatively, and not necessarily for all he is capable of learning technically.

[7] Michel Waisvisz (Krefeld, 1990, page 31).

# Chapter 10

# POD-X Polyphonic Computer Composition System.

## 10.1 Introduction to the System.

Currently available at the Department of Communications, Simon Fraser University, Burnaby, British Columbia, the POD-X polyphonic composition system consists of a complex set of programs which afford the user real-time synthesis possibilities in an interactive composition environment. Designed, developed and continually updated by the composer and programmer, Barry Truax, the POD-X is an example of the integration of sound and structure within a computer-based composition environment[1]. As Truax (1991: pg. 1) writes:

> "Central to the instrumental music approach, and coincidentally to most computer music systems, is the separation of sound and structure. Within the MUSIC V system and its many relatives, this separation is expressed as the "score" and "orchestra"; within the MIDI world it is embedded in the difference between the "note on, note off" data and the frequently arbitrary nature of the timbres used in the synthesiser. "

[1]The system described in this section is the POD-X family of programs which were used by the author during a composer-in-residency at Simon Fraser University in the Autumn term of 1991. The POD-X is based on an LSI 11/23+ micro 11 central processing unit, with a 31Mb hard disk, two 10Mb RL02 removable drives, 2 floppy drives, a Kennedy magnetic streaming tape drive with high speed (45ips) transport, 2 CRTs (for either in studio or separate room use), and a line printer for hard copy. The main synthesis and processing hardware is based on a DMX-1000 signal processor, with 2 output channels, and a DMA interface to the disk's streaming tape drive to allow sample transfer from storage to the DMA-1000's 16-bit digital-to-analog convertors for real-time feedback.

The POD-X system has evolved around a very distinct compositional working model illustrated in Figure 10.1.

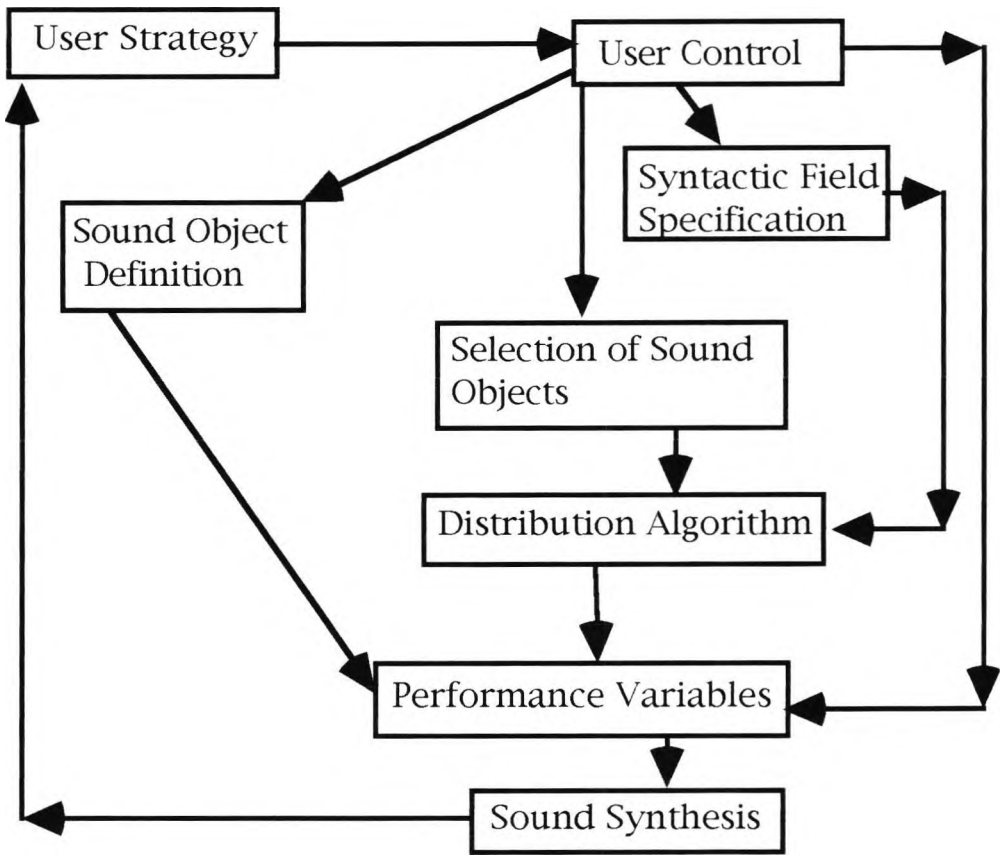



Figure 10.1 Compositional Model of the POD-X System (after Truax 1978, pg. 43)

Speaking of this model, Truax (1978, pg. 42) tells us the compositional strategy encompassed by the POD-X system allows the composer to set up:

> "... general patterns of frequency regions, amplitude ranges, and density of attack points (the "syntactic field specification" ...) and then determines how a specific set of defined "sound objects" (or timbral entities) is to be selected and displayed within the given structure...A further level of control is called "performance variables" and with

these variables the composer can achieve alternative versions of the same set of events, mainly through control of the time structure and the ratio of event duration to delay entry..."

Inherent in the user psychology of the POD-X system is the interaction by the user with the constant feedback supplied at all times during a normal working session. As Truax (1976: pg. 30) states:

"The user strategy, external to the program, works with the mental representation that the user has of a possible goal structure, and that of the present structure derived from hearing the sound synthesis result."

The real-time feedback loop is an extremely important factor in the success of the system in translating original notional outlines that the composer has into final sound textures in a completed composition.

The POD-X environment operates in two very defined modes, described by Truax basically as 'bottom-up', and 'top-down'. In this fashion the user can function from either the macro- or the micro-level, and have a complex set of controls which afford flexibility at every stage.

## 10.2 System Details and Mode of Operation

What makes the POD-X quite different from the other compositional environments discussed in earlier chapters is that the

POD-X is almost exclusively centred upon one synthesis technique, the granulation of sonic material. Developed from theories by Gabor (1947), researched and developed further by both Weiner (1964) and Xenakis (1971), the most recent influence responsible for the implementation by Truax for the POD-X has been the work of Curtis Roads. Roads (1988: pg. 11) writes that:

> "Granular synthesis involves generating thousands of very short *sonic grains* to form larger acoustic events. The technique can be classified as a form of *additive synthesis* since sounds result from the additive combination of thousands of grains."
> (italics as in original).

In Roads work, the grains are organised into events, which are characterised by 12 parameters, detailed in Figure 10.2:

| | |
|---|---|
| 1. Beginning time | 7. Bandwidth. |
| 2. Duration. | 8. Bandwidth slope. |
| 3. Initial waveform. | 9. Initial grain density. |
| 4. Waveform slope.[2] | 10. Grain density slope. |
| 5. Initial centre frequency. | 11. Initial amplitude. |
| 6. Frequency slope. | 12. Amplitude slope. |

**Figure 10.2 Parameter Listing for a Sonic Grain**

The POD-X implementation of granular synthesis employs a different set of parameters for the composition of an individual grain. These vary according to the actual model of granular synthesis being used. The system provides three individual models of synthesis, as detailed below, each employing a unique set of parameters, as shown:

---

[2] Roads (1988; pg. 12) defines waveform slope as the transition rate from a sine to a band-limited pulse wave.

## 1. Model 1. (A.S) Amplitude Synthesis

A simple oscillator with variable frequency, waveform and duration. The parameters required are as follows:

| | | |
|---|---|---|
| 1. Frequency. | 3. Duration. | 5. Delay (density). |
| 2. Frequency range. | 4. Duration range. | 6. Ramp value. |

## 2. Model 2. (F.M) Frequency Modulation

A basic F:M oscillator pair with specifiable carrier:modulator ratio, carrier frequency, duration and maximum modulation index.

| | | |
|---|---|---|
| 1. Frequency | 4. Duration range. | 7. Modulation index. |
| 2. Frequency range. | 5. Delay (density). | 8. Index range. |
| 3. Duration. | 6. Ramp value. | |

## 3. Model 3. (SAM) Sampling

Sampled sound with specifiable duration and offset time.

| | | |
|---|---|---|
| 1. Offset. | 3. Duration. | 5. Delay (density). |
| 2. Offset range. | 4. Duration range. | 6. Ramp value. |

This sampling granulation synthesis version exists in three variants:

- Model 3A

A fixed short sample is used, extracted from the recording on the disk and loaded in RAM (so length is restricted to a 4k on-board sample).

- Model 3B

Real-time direct granulation of the streamed samples read directly and continuously from the disk, with the 4k RAM acting as a short delay line.

- Model 3C

An interpolation between the above two methods which, by allowing the rate at which new samples are read into on-board memory to be varied, provides the option of variable rate granulation of the stored sound samples. The rate may be varied from real time to anything in the region of thousands of times slower than real-time sample granulation.

The parameter values are entered into the system as a string on a command line. The typical format of a command line string for control of a POD-X file in real-time is shown in Figure 10.3. In this example, values are being assigned respectively to an envelope, an increment value, a frequency, a frequency range, a duration, duration range, a density amount, a ramp value, and finally a number of voices. We shall see during the course of this chapter how the user can control and manipulate each of these parameter values.

| Env | Inc | Freq | FRange | Dur | DRange | Dens. | Ramp | N.Vces |
|---|---|---|---|---|---|---|---|---|
| 6 | 3 | 825 | 85 | 15 | 50 | 20 | 75 | 15 |

**Figure 10.3 Typical Command Line for POD-X control.**

Further to the three types of granular synthesis available, the POD-X system contains spatialisation and score programs which allow for large scale fully integrated development of sonic textures in real-time. Based, as we have seen, on fixed waveform and FM synthesis models, the POD-X system consists of

22 programs available to the user for interactive composition with real-time feedback.

Within each of the three types of granular synthesis available on the system, the user is afforded complete and comprehensive control over the shape of the actual envelopes within each sonic grain. Each grain has a 3-part attack, sustain, release (ASR) linear envelope, whose attack and release (decay) portions are set by default to 1/4 of the overall grain duration. However, at the far left of the command line, the user can alter the envelope parameters between the extremes of 1/2 to 1/16 of the grain total duration for the attack and decay. These vary from a triangular (1/2) to approximately a rectangular (1/16) waveform.[3] This envelope symmetry makes the POD-X grain envelope completely palindromic.

The POD-X system as used by the author comprised a set of 22 programs, facilitating comprehensive granulation techniques, together with editing facilities and some spatialisation programs. Appendix F contains a program listing, together with brief descriptions of each of the programs available, for reference purposes.

To allow the user full control of frequency when using the Additive Synthesis (Model 1) form of granular synthesis, the POD-X system includes a fully interactive program, *Wavex*, for

[3] The attack and decay values are not separately specifiable.

creating waveforms and also processing and testing them. Waveforms can be created in one of eight ways:[4]

- Formula Waveform.

  A choice of 8 pre-defined waveforms which may be stored.
- Harmonic Construction.

  Working on a previously stored waveform, the user can specify up to 32 harmonics.
- Filter Modification.

  Working on a previously stored waveform, the user has the choice of either low-pass filtering or resonance construction.
- Mix Waveforms.

  The user can mix any two previously stored waveforms with amplitude control over the second waveform.
- Rectify.

  The option for half or full wave rectification of previously stored waveforms.
- Random.

  Generation of random waveforms, with a choice between white noise, Gaussian or 1/F distribution equations.
- Copy from another File.

  Duplication of one waveform to another in the same waveform file.
- Chebychev Polynomials.

  These equations are used as distorting or shaping functions in waveshaping.

[4] POD-X system documentation available as on-line material.

As may be seen in Appendix F, the POD-X system itself envelopes a large set of programs which collectively afford both real-time and non-real time working modes, and several merge, spatialisation and complex algorithmic compositional techniques. In all of the system programs the data handling, in line with the compositional methodology supported by the POD-X, is efficiently hierarchical. As Truax (1978: pg. 44) tells us:

> "The data structure is hierarchical in the sense that variables at one level interpret data at a lower level, (e.g., performance variables interpret event data, and sound objects interpret acoustic data). Although the same range of output could be realised without such variables, it can be accessed much more quickly and efficiently with them, and hence a wider range of output (which reflects an increased generality) is simply more readily available to the user."

By supporting data transfer and access from and to all levels, the system actively encourages the notional compositional hierarchy (bottom up and top down with data interpreted between all levels at all times), which the designer has evolved and which allows the user the choice to operate at either the micro- or the macro-levels of the creative process.

From a very early stage of exposure to the POD-X, the user realises that compositional decisions are to be formulated at three distinct levels, listed here in a top down, macro > micro, fashion:

- the combination of scores level
- the score events level

- the sound object level

This three-tier compositional process is inherent in the system design and everything within the program structure is implemented to operate at one or more of these levels. The compositional hierarchy of the POD-X can be further defined by the following sketch, Figure 10.4, which details how the three tier process may be made to reflect a compositional process, reading either top-down or bottom-up.

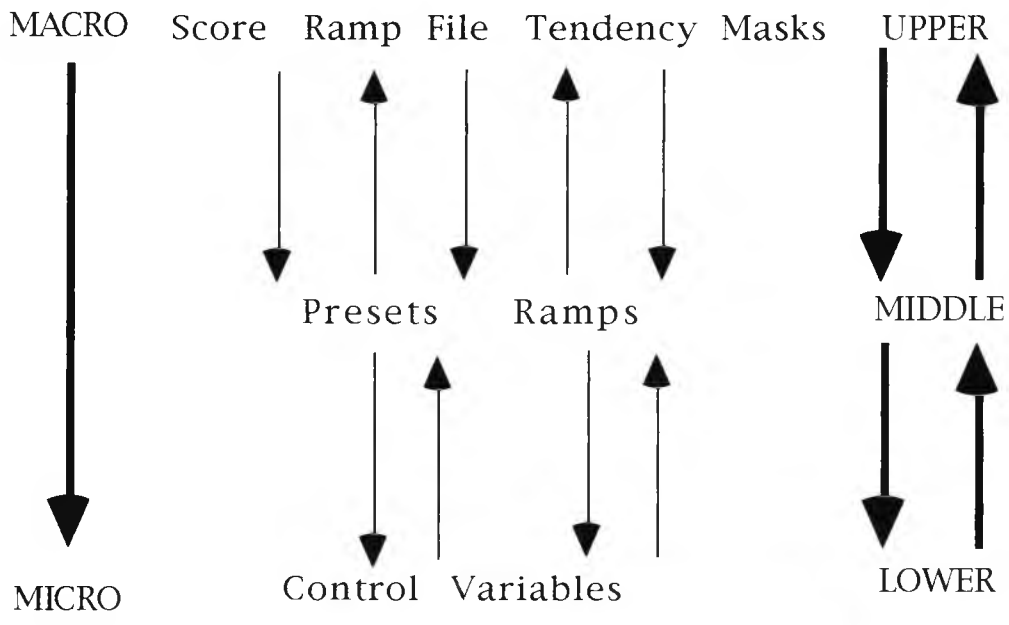



Figure 10.4 POD-X Inherent Composition Hierarchy

It is clear that the user has the choice of several working paths. This hierarchy, or the individual elements of it, will now be discussed in detail in order to reveal the basics of POD-X program operation.

Working from the micro level at the *control variable* stage, the user specifies exact values for several parameters,

controlling the physical make-up of a grain. The single command line for this control involves supplying parameter values for:

- frequency
- duration
- delay (in ms)
- number of voices
- panning position of outputs
- frequency range
- duration range
- ramp value
- envelope

Control of these parameters can be handled in real-time on-screen, as in the control line string seen in Figure 10.3. The user tabs between various numerical positions, and either types in a new value or holds down the + or - keys to effect an incremental change on the parameter value. An incremental index can be specified so that gradual incrementing does not have to be by a value of 1 at all times.

Moving one level up the hierarchy, all of these parameters can be stored as a collective *Preset* by the user to facilitate immediate change from one set of parameter values to another during real-time playback. These presets are stored as a letter of the alphabet and recall of any particular one is as simple as hitting the letter on the keyboard. By using presets, the composer can build up an extensive library of various textures, all able to be loaded and used in real-time. Attack and decay rates for presets can be controlled by the caps lock key. Presets can be initiated with a decay at a rate equal to a variable *inc* when the caps lock is off. The spacebar can be used to initiate an attack at that rate as well. If the caps lock is on, there is no variation in

amplitude and a continuous sound texture related to the loaded preset is heard.

The employment of *Ramping* is another very effective facility for the creation of continuously varying textures in real-time. Described by the designer as (Truax, 1989, pg. 7)) :

> "...patterns of continuous change in selected parameters at a specific rate (in contrast to the discrete changes implemented with presets)..."

By placing a + or - sign in front of any particular parameter value, and setting in motion any of a choice of four ramping styles, ascending, descending, cyclic or random, the selected parameter value is affected by an independent ramp generator. The ramping value is dictated by the set increment value (*Inc*), on the same command line. Large ramp steps can be set by placing a numeric value (X), after the + or - sign in front of the selected parameter. As a result, the parameter value is ramped in step by values to the order of X times Inc. An outline of a typical command-line with several ramps in operation is listed in Figure 10.5 below:

| Env | Inc | Freq | FRange | Dur | DRange | Dens. | Ramp | N.Vces |
|---|---|---|---|---|---|---|---|---|
| 7 | 2 | +2 800 | -3 100 | 25 | -4 50 | +5 20 | +4 75 | 20 |

Figure 10.5 Ramp Control on a Command Line String

In the above example, the envelope value position may be varied to show either envelope details or panning positions. The preset name which is loaded or the type of ramp which is to be used are depicted by a letter of the alphabet which appears just to the left

of the Inc position. It is clear that, with a command line system in which the user accesses only one line to control the entire system, then a great deal of information must be available at the same time.

Ramps may be controlled manually in real-time, being switched on and off by hitting R on the keyboard or, as an editor has been built in, ramps may be stored as an independent ramp file for recall at any time by hitting I on the keyboard to initiate the ramp run. Both maximum and minimum values can be specified in the ramp file. If a ramp file is initiated at any time when the value to be affected is outside the ramp file upper or lower limits, a reversal of synchronisation is carried out to bring the value back within the required range. Up to 18 sequential ramps may be stored on the system at any one time. Each ramp file includes parameters as follows:

- ramp type (A, D, Q)[5]
- ramp scaling, (values 1-10), for each synchronised parameter
- synchronisation (+, -, none)
- end condition (elapsed time or specific parameter)

Presets may be captured 'on the fly' during real-time performance of a ramp, which allows the user to store specific momentary parameter values during the run of a ramp file for future recall as a preset file. This is a very useful function, as it allows for saving of successful settings during a constantly changing texture in real time.

[5] Ramps may be ascending, A, descending, D, or random mode, Q.

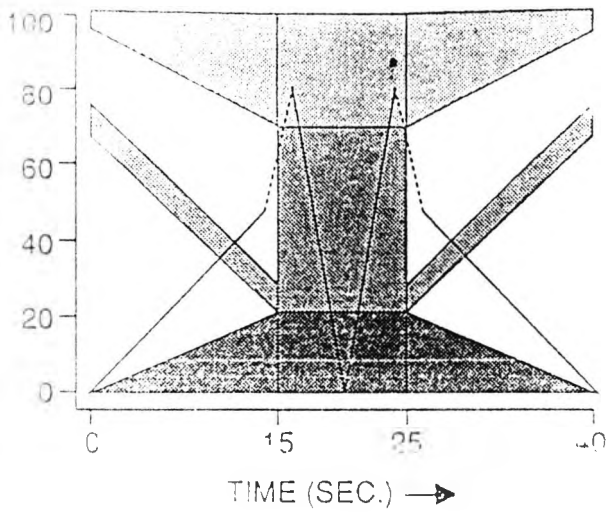
The discussion of ramp files has taken us to the middle level of the system compositional hierarchy. On the top level of the compositional hierarchy, the user accesses the macro-level programs in the POD-X system. When employing Tendency Masks, the user is thinking in terms of complex gestures as opposed to on-going processes as with presets and ramps. A tendency mask is defined by the designer (Truax, 1989: pg. 8), as:

> "...an area within which values may be chosen"

A tendency mask is created by numerically specifying the corners of the mask and its duration, or the end points of the specific envelope segment and its duration. In operation, the masks and envelopes are translated by the program into a series of presets and ramps, so it is clear to see that the use of masks is a level higher in the compositional hierarchy allowing the user to think in terms of large-scale gestures, as opposed to time-stamped control variables. A pictorial representation (Truax, 1989: pg. 9) of the use of tendency masks by the developer in one of his compositions, *The Wings of Nike* displays the ideas involved in tendency masking to produce areas of available parameter values for the system (Figure 10.6).

Within the program GRMSKX, which allows for the control of granular synthesis via tendency masks, the user defines average values and a range of values for each of the inclusive parameters to be effected as a set of tendency masks or, if only one value is involved, as an envelope. Tendency masks can be used to

vary frequency, duration or modulation index, and envelopes are used to vary amplitude or delay time values.




Tendency Masks used in The Wings of Nike.

Parameters shown are
frequency (light shading)
duration (medium shading)
mod.index (dark shading)
delay times (lines)

Figure 10.6 Tendency Masks Used in *The Wings Of Nike* (Truax, 1989: pg. 9)

Also on the upper level of control, the composer can operate within the *Score Creation* level to build large gestures from preset and ramp controls. Scores organise presets and ramps into particular performance sequences and are created in one of two ways:

- by storing a real-time alpha-numeric keyboard performance of the presets;
- by editing previous stored orders.

The program, PDFILG, a score editor for granular synthesis, can provide up to 160 presets in a single file together with exact data as to how these presets are to be performed in time. The data required by PDFILG includes:

- preset name;
- entry time in centiseconds;
- maximum amplitude.

Overriding values for frequency and duration may be set in the score file, which will override those saved in the current preset. Ramps may be used simultaneously with scores but the rule in this case is that any variable singled out for synchronisation is no longer controlled by the score file but by direct real-time commands from the user. In this way, modification is still an available option even during macro-level score control.

At any stage of the three-tier compositional hierarchy, at either the micro- or the macro- working levels, the user has the option to have real-time spatialisation of the sonic feedback. The system affords several spatialisation methods, which in real terms involve the control of the number of voices output to each of the two playback channels. Spatialisation is achieved in a number of ways:

- toggling the position variable with the up and down arrows on the keyboard. The current value is reported to the user in the leftmost position on the command line;

- ramping the output positions of the voices to the extreme left or the extreme right by pressing [ or ] on the keyboard respectively;

- applying a previously created and selected trajectory file to the sound output which results in spatialising the sound texture according to the path created by the user in the program

PLOTX, a program for spatial trajectory composition and file testing. This trajectory map is translated into numeric values which directly influence the number of voices sent to each of the two output channels, a form of automated panning control;

- applying a trajectory to a score file during performance.

With PLOTX, up to 36 points are user-specifiable in a two-dimensional space according to 16 radial positions and amplitude, relating to laterality and distance from the perceived position of the listener respectively. Figure 10.7 amd 10.8 below show the hard copy of a trajectory traced with breakpoints identified as A-Z then 1-10, if all available 36 points are used. If continuous textures are used, it is important to remember that moving from a very distant position to a very close position with no points between may be very abrupt, so it is useful to specify plenty of intermediate values to prevent either clicks or unnaturally abrupt sound movement.

```
.....7|6.....
P.....8|5.....RST
O.....9|4.....U
N.....A|3.....V
M.....B|2.....W
L.....C|1.....X
KJIHGFED|OZ.....Y
```

This trajectory has 43 points plotted and is not used in reverse order.

Figure 10.7 Trajectory Trace Screen Plot.

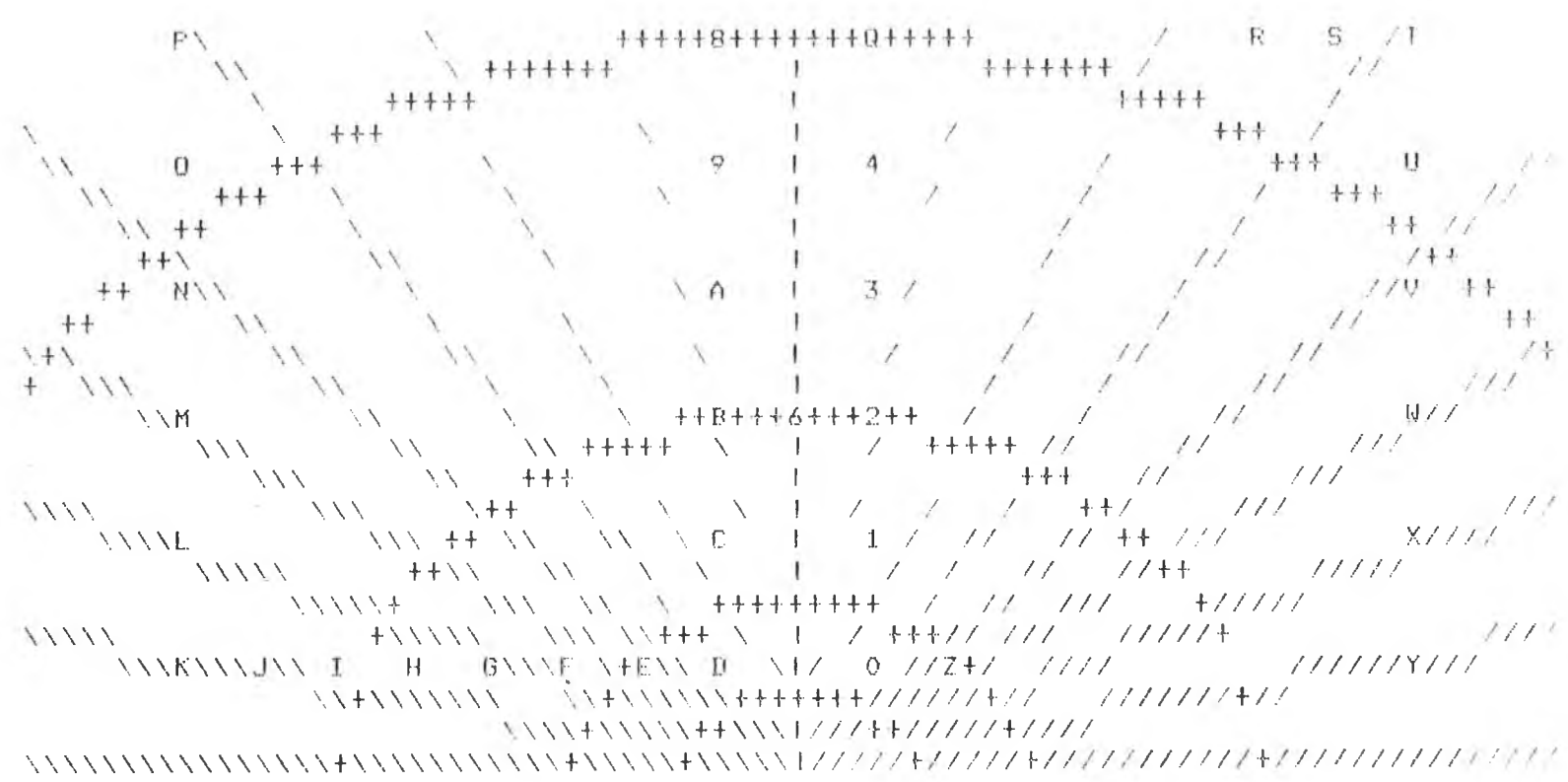


Figure 10.8 Trajectory Trace in Expanded form

THIS IS THE EXPANDED HARD COPY OF TRAJEVTORY MICTRA

THIS IS A DYNAMIC MOVING TRAJECTORY. REPEATED IN SPACE AND OF QUITE A SHORT

DURATION IN ITSELF.

EMPLOYS CONSTANT SPEED BETWEEN POINTS. (OPTION 2). AND THE DURATION OF THE TRAJEVTOR

IS 15.00 SECONDS. REPEATED A TOTAL OF 6 TIMES IN ANY SINGLE EMPLOYMENT.

The speed of a trajectory is given as a total duration and there are two options for 'reading' the trajectory in time:

- in constant time
- with the illusion of constant speed.

(near objects appear to be moving faster than distant ones).

The second option creates the illusion that, as is psychologically correct, near objects appear to be moving faster than distant ones. The speed can, however, be further manipulated by relating the trajectory to the *Inc* value on the command line, (placing + or − sign in front of the *Inc* value), negative values will speed up the trajectory movement and positive values slow down the reading of the trajectory, therefore moving the related sound output slower in space. The settings of either +1 or −1 on *Inc* indicate the use of the original speed value which was set when the trajectory was created in PLOTX. It is worth noting that, in this fully integrated real-time system, the eventual output trajectory speed is also determined by the complexity of all of the other parameters competing for real-time execution in the DMX-1000.

In addition to the real-time granular synthesis programs on the POD-X, the system also supports several other complex algorithms adapted for compositional use including implementation of the Karplus-Strong[6] algorithm, frequency modulation models, and also chaotic non-linear theories. The FM algorithms are fully implemented in the granular synthesis environments and a complex FM tutorial is also available on-line. The Karplus-Strong algorithm is afforded quite novel real-time access by literally 'playing' the alphanumeric keyboard, with octave

[6] Karplus Strong Algorithm (Karplus, K and Strong, A. (1983)).

specification on the numeric keys and particular notes within each octave mapped to the letter keys. Like the granular synthesis programs, various sampled waveforms can be applied to the algorithm, as well as a fixed waveform and FM grains. The algorithm dies away after each excitation in an exponential fashion.

As the POD-X system designer, programmer and its most prolific user, Barry Truax (1989: pg. 8) writes:

> "The current implementation, while just a beginning, has already gone some distance in establishing an appropriate hierarchy of levels of compositional control, ranging from the control levels at the grain level, through to groups of such variables (presets), rates of change (ramps) of the control variables, and macro-level tendency masks and scores to determine large-scale forms. In addition, the user has a choice of compositional processes ranging from improvisational to design-based, with each approach being able to incorporate significant elements of the other."

The POD-X is, in this collection of systems descriptions and evaluations, unique in that it incorporates a compositional methodology and actively encourages the composer, as a consequence of system design and programming strategies, to utilise this hierarchical methodology. Composition at the micro level goes far beyond the traditional concept of the single sound object, but involves the actual creation of a grain of sound. As Truax (1990: pg. 5) writes:

> "The conventional distinctions, found even in computer music systems, of score and orchestra, or in MIDI between note commands and arbitrary synthesiser patches, are obliterated in a more integrated, even organic process."

The POD-X, as a compositional system, presents itself to the user with an individualistic compositional methodology which must be embraced completely in order for the user to produce effective results.

## 10.3 System Synopsis, Analysis and Evaluation.

The analysis and evaluation of the POD-X system is based on research and composition carried out by the author as the Composer-in-Residence in the Department of Communications at Simon Fraser University, Vancouver in the Autumn term of 1991.

### 10.3.1 System Synopsis Grid.

The POD-X system is summarised in Figure 10.9 in the system synopsis grid as introduced in Chapter 5.

In the grid, the system is seen to be almost exclusively command line based, with the trajectory plotting for the PLOT-X program being the only graphical interfacing of any sort included in the system.

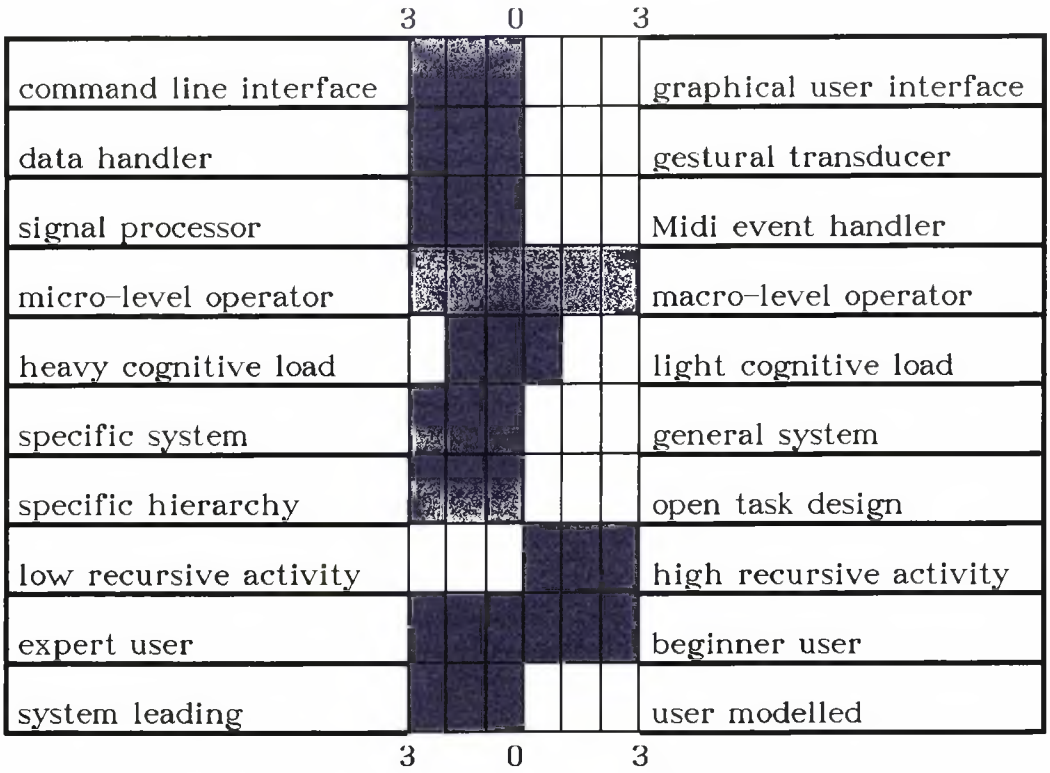



Figure 10.9 POD-X System Synopsis Grid

As a data handler for signal processing capabilities, the POD-X inherently embraces a micro- to macro-level of task operation. With a medium to heavy cognitive load, requiring the user to input numerical data to a single string at all times, in real-time, the POD-X has a specified compositional approach, with a very defined compositional hierarchy. Editing and error correction is well facilitated on the system, with data handling across a single string able to be altered at any stage as the user listens to the real-time granulation taking place as aural feedback. Suitable for beginners to experts across the user spectrum, with initial output being very easy to achieve, the POD-X imprints a task structure very heavily on the user.

### 10.3.2 System Analysis and Evaluation.

*"...Define and maintain a system balance between generality and strength....Confine the degree of system leading of the user purely to enhancing user friendliness and ease of operation..."*

There can be little doubt that the POD-X system leads the user very strongly, and is completely specific with regard to the compositional methodology inherent in the system. The POD-X actually provides no flexibility with compositional hierarchies for users with individual compositional approaches. One of the major points of contention with a system such as the POD-X is the fact that the output from the machine, regardless of the composer involved in the compositional process, is very strongly identified with the system. As a result of this, users find it very difficult to retain a sense of personal identity when composing with the POD-X. This degree of system leading and loss of user individuality becomes a serious issue when considering the values of a system with such a highly defined compositional hierarchy.

*"...Maintain a clear structure throughout the micro and macro levels of the system..."*

Although much may be said about the compositional hierarchy of the POD-X, and its adverse effect upon user creativity from the point of view of individuality, it must be agreed that the actual structure of the hierarchy has been very effectively implemented. For the user who embraces the compositional hierarchy and

flourishes creatively within it, the POD-X is a system with a very well implemented structure from micro- to macro-levels. The user can operate from the micro-level, handling single parameter values equally as easily as at the macro-level, controlling trajectories for sound spatialisation. Also the user has the ability to move from micro- to macro-level very smoothly and easily.

*"...A general release system should encourage user development, allowing access at all levels to accommodate all user communities, and to encourage and lead beginners to experiment with micro level tasks and functions in the system..."*

With the POD-X design encompassing a very organised compositional strategy from the micro- to the macro-levels, users approaching the machine can choose a level to operate at where they feel technically proficient and sufficiently in control of the interface to effect a creative output. With three very defined compositional levels, users from beginner to expert are accommodated within the compositional strategy, which is very strongly imprinted in the system.

*"...Choose the input device carefully, considering compatibility with system and user, accuracy within the system, efficiency and ease of use..."*

As the only controller for the POD-X being the alpha-numeric keyboard, the user is presented with an inflexible input method to the system, and one which maps poorly to the creative task at

hand. The user must effectively learn to be very keyboard literate and employ the keypad at high speed to input values to the command string. No use of a trackerball or mouse facility means that left–right–left orientation across the command string is controlled by the arrows on the keyboard and beginner users find the interfacing initially akin to early computer games in input style. Although the command line input is certainly efficient in use, the input and control methods for the POD–X do not help to reduce cognitive load and fatigue for the user. This is due to the fact that only one string is visible at any time, with previous granular settings impossible for the user to see in real time, even though all of the screen is blank with the exception of the command string across the very bottom.

*"...Strike a balance between a system which is a gesture transducer and one which is an information handler....Combine the presentation of text with graphics where applicable...."*

As a system, the POD–X requires the user to operate with numerical or keyboard data at all times. Even when the user is 'drawing' a trajectory shape on the screen, he is required to place an X at each breakpoint position, and has no opportunity to gesturalise the shape via mouse or light pen input. The POD–X acts purely as an information handler, and feeds back granular textures in real–time constantly to the user as he alters numerical values on the screen. This style of input maps very poorly upon the nature of the output received by the user. Some inclusion of a graphical input for trajectories would map more easily to the user's mental

model of sound objects moving in space, rather than a series of X placements on the command screen.

*"...Reduce user cognitive load to leave the musician user free to concentrate on the aural output, and not to have to unduly worry about system operation...."*

Although the style of input to the POD-X system is mapped poorly upon the musical nature of the task and also upon user mental models, the user who learns the 'performance' of the alpha-numeric keyboard to facilitate smooth control of and input to the command string can quickly become immersed in the effect on the aural feedback of parameter changes. The input becomes the secondary issue, as each data alteration is immediately noticeable aurally. In this fashion, the system facilitates a close relation of user action to sonic reaction, and allows the more experienced user who has mastered a fluid communication on the command line string to concentrate on the ever altering sound output.

*"...Allow the user to easily undo or reverse an action..".*

One of the benefits with the single line input style of the POD-X is that at all times the user can alter a parameter, and immediately undo that action easily if required. As an input process, although the user is required to be cognitively aware of several data inputs simultaneously, the editing of any of these inputs is very simple which increases the user's likehood to experiment with varying parameter settings.

The POD-X presents itself as a very defined system with regard to the compositional hierarchy embedded in it and the nature of the sound output from it. The user who embraces this hierarchy has the opportunity to work with a system which gives immediate aural results for even the smallest alteration of a parameter value.

# 11. Specialist Interfacing Considerations: KE:NX[1]

## 11.1 Introduction to the Environment.

In the present chapter, many of the ergonomic considerations, user issues, design methodologies and task definitions discussed previously will be seen to be put to use in the design of a user interface for music composition. The aim is to design a user-friendly interface while simultaneously creating a computer music system design methodology for this particular design task. This methodology will be seen to be an amalgamation of the most applicable aspects of the methodologies discussed in Chapter 4, together with the user and task analysis discussed in Chapter 3. The guidelines introduced in Chapter 5 will permeate the entire discussion and design process.

The user interface design introduced and discussed in this chapter details a very specific design case. The user for whom the interface is designed was at the time of writing a student of composition and performance attending workshops run by the Drake Music Project Ireland and is severely physically disabled. The interface being designed and implemented is intended to provide independent control by the user of the computer system with which he intends to compose.

[1] The programming control environment KE:NX is pronounced *kennex*.

### 11.1.1 The Drake Music Project.

As a registered charity, The Drake Music Project was set up in the UK in 1988. The objectives of the charity are explained in the Charity Trust Deed (1986)[2] as follows:

> "The advancement of education for physically disabled children and adults in the Arts through the use of new technology, the provision of facilities for recreation and other leisure time occupations with the objective of improving their conditions of life, the promotion of research into education and the therapeutic use of the arts for the benefit of the disabled and any other connected charitable purpose."

The overall aim of the charity's existence, in the words of the founder, Adele Drake, is:

> "in order to enable physically disabled children and adults to make music."[3]

In pursuit of this aim, the Project employs advanced music technology equipment, in combination with specialist switch access and adapted interfacing techniques in order to enhance any degree of physical control and voluntary movement which a disabled person may have. This voluntary movement, no matter how slight, is extended with the aid of switches and physical controllers mapped onto computer interfaces, to provide as independent a control environment as possible for the disabled musician.

---

[2] Drake Music Project Charity Deed, London, 1986
[3] Drake Music Project Charity Deed, London, 1986

The Drake Music Project Ireland came into existence in November 1992, with the author as the Project Manager, Senior tutor, and resident Interface Researcher. Workshops began in Belfast in November 1992, and in Dublin in April 1993. The volunteer student for the interface design under discussion in this chapter was a composition student in the Drake Dublin workshop attending on a weekly basis throughout this period.

### 11.1.2 Project Workshop Equipment.

The range of equipment in use in the Drake Ireland workshops at the time of this research included several computer-based workstations, all of which were designed to be controlled by means of switches, either single or multiple switches simultaneously, to replace the operation of a mouse as controller or the use of an alpha-numeric keyboard for input to software environments. The three computer-based systems used on a regular basis in weekly Project workshops in Ireland include:

- Atari Sequencer Systems with controller keyboards

As seen in Figure 11.1 below, this system typically would be used to run the Midigrid graphics sequencer package with Midi keyboards as input controllers and Midi sound modules.

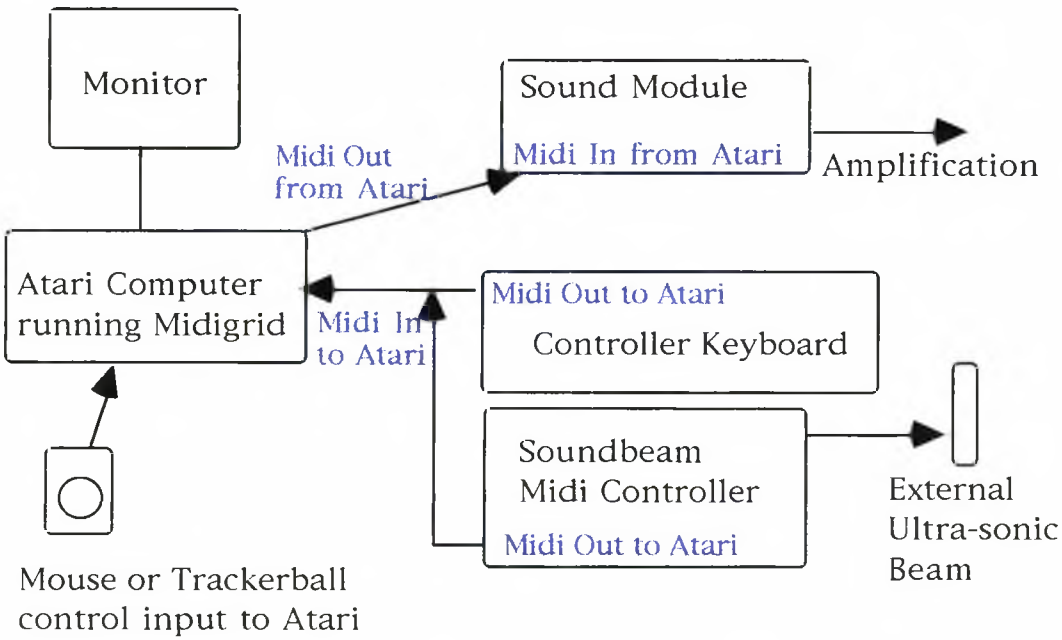



**Figure 11.1 Atari Composition Set-up using Midigrid sequencing software**

Data control of the sequencer package is achieved by using either a trackerball device (e.g.: Marconi[3] trackerball), or home made mouse port extension kits, which allow combined use of a mouse, together with additional extended switches to afford control of left and right clicks for software data entry and control.

- Atari Sequencer Systems with Soundbeam

Employing an Atari computer, Midigrid sequencer package and a sound module, this set-up includes the soundbeam as a controller and input device. The Soundbeam, as seen in Figure 11.2, is a Midi

[3] *Marconi* is a registered trade mark for a trackerball device in use in the workshops. The trackerball is manufactured by Marcus Inc. USA.

controller, operating by sending out a conical beam of ultrasound, the length of which is controllable by the user. The ultrasonic pulses are reflected back into the beam's sensor by interruptions of, and movements within, the active beam area. Information about the speed and direction of this movement is translated into Midi data.


Figure 11.2 Atari Sequencing Set-up with Soundbeam

As a controllable device the Soundbeam allows the user to set three parameters:

- Mode

  Within this setting, the user dictates which type of note combinations the device will create in Midi, as it perceives

length, the choice on the Mode switch relates to the type of scale these 16 notes will create, what the relationship of one note to the next will be (an arpeggio or a chord), and whether pitch bend or modulation will occur in conjunction with movement in the beam.

- Transposition

In order to be able to 'tune' the beam for playing within an ensemble with other sound modules in use, or with acoustic musicians, the transposition dial allows for semitonal transposition of the notes within the beam relative to each other for each of the modal settings.

- Range

To allow for extensive movement or just the smallest range of physical movement to trigger all 16 notes, the Range dial allows the user to set the length of the active beam from the smallest setting of 0.25 metres to the most extended beam range of up to 6 metres. For use with people with physical disability and limited movement capability, this setting is very useful. A short beam will afford a user with limited range of movement to access all 16 notes within the physical space of the beam, while being able to set a very long beam active range means that several users can share the active sound space, or perhaps a user who is wheelchair-bound can 'dance' within the active beam area, creating music as a direct result of their movement.

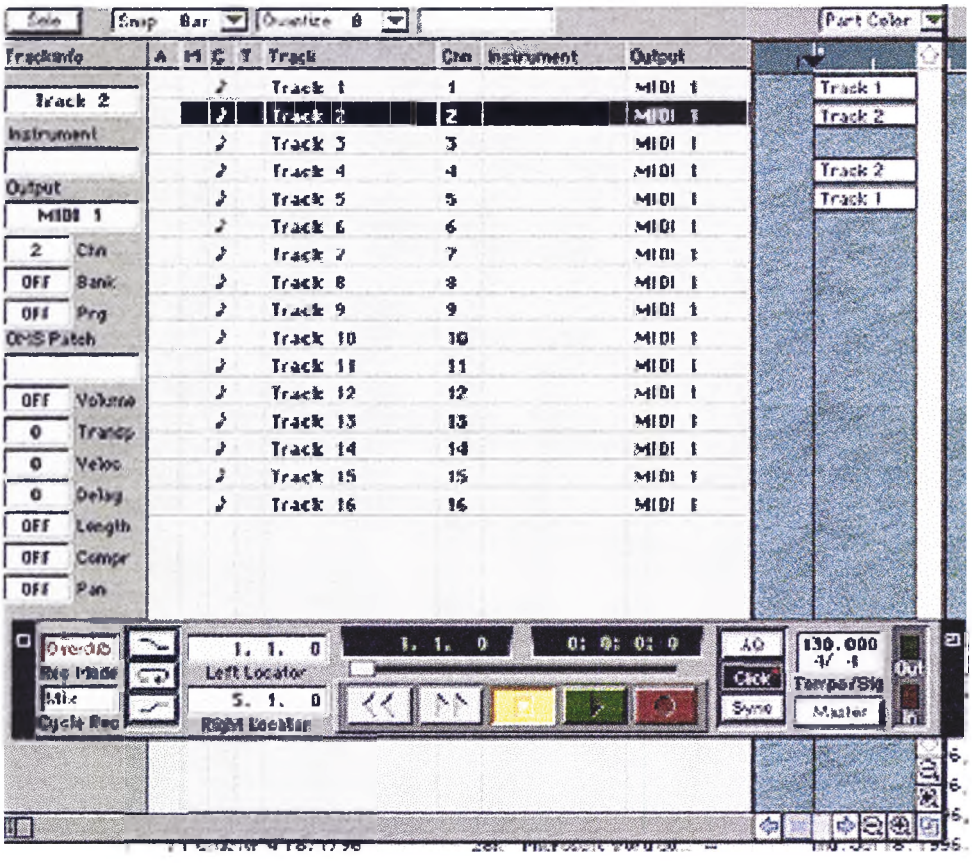



Figure 11.3 Cubase Track Screen

This Macintosh computer based system set–up involves running the Cubase sequencer package with third party joystick devices as input controllers to the software, either a Midi controller keyboard, Soundbeam, or a synthesiser as Midi input device, and a sound module. Cubase, from the Steinberg software house, is a multi–channel sequencer package allowing up to 64 tracks of Midi musical data to be created per song file. Figure 11.3 details the main Track screen of the program. As can be seen from the actual amount of graphical detail provided on the screen there are a large number of active areas which the user can interact with and facilitate changes in a multitude of parameters; the amount of

detail in the track information panel alone (left side of the screen), where the user can input track and instrument name, channel bank and program numbers, together with a wealth of data inputs relating to volume, transposition, velocity, delay etc, indicate that a user employing a mouse to control the Cubase sequencing environment needs to have controlled and steady movement in order to successfully access the various active sections of the screen.

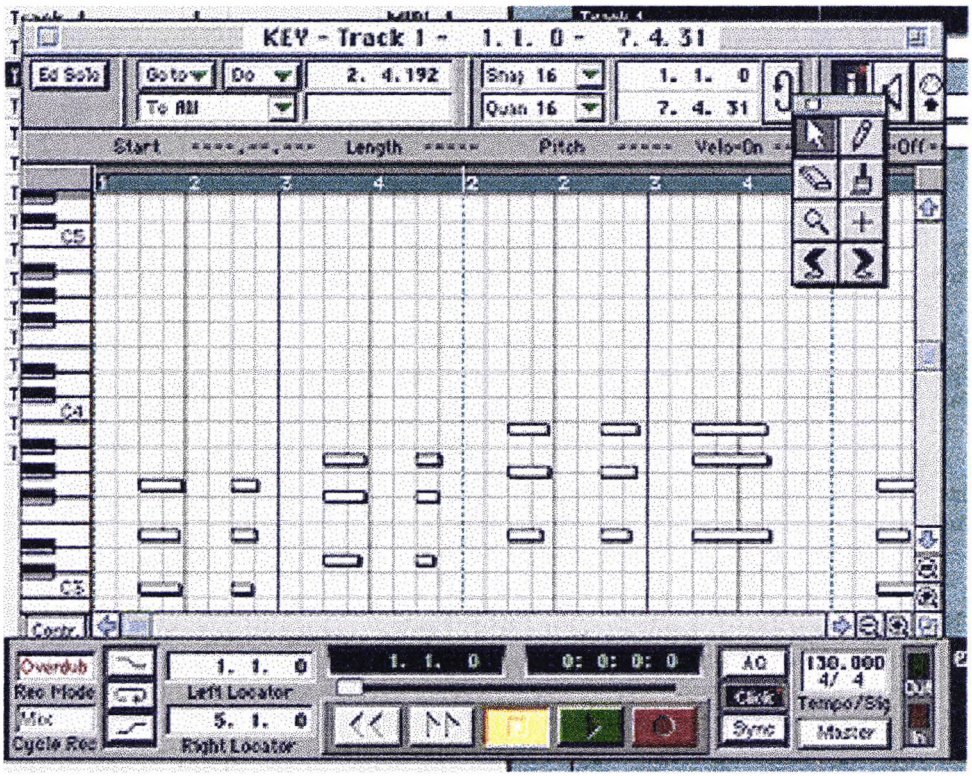



Figure 11.4 Cubase Key Edit Screen

Full editing functions are available in the program, most of which a user will want to have access to and be able to control within sometimes minute physical screen space. Figure 11.4 displays the *Key edit* screen of the program, and Figure 11.5 shows the *Score edit* screen. The details and parameter inputs

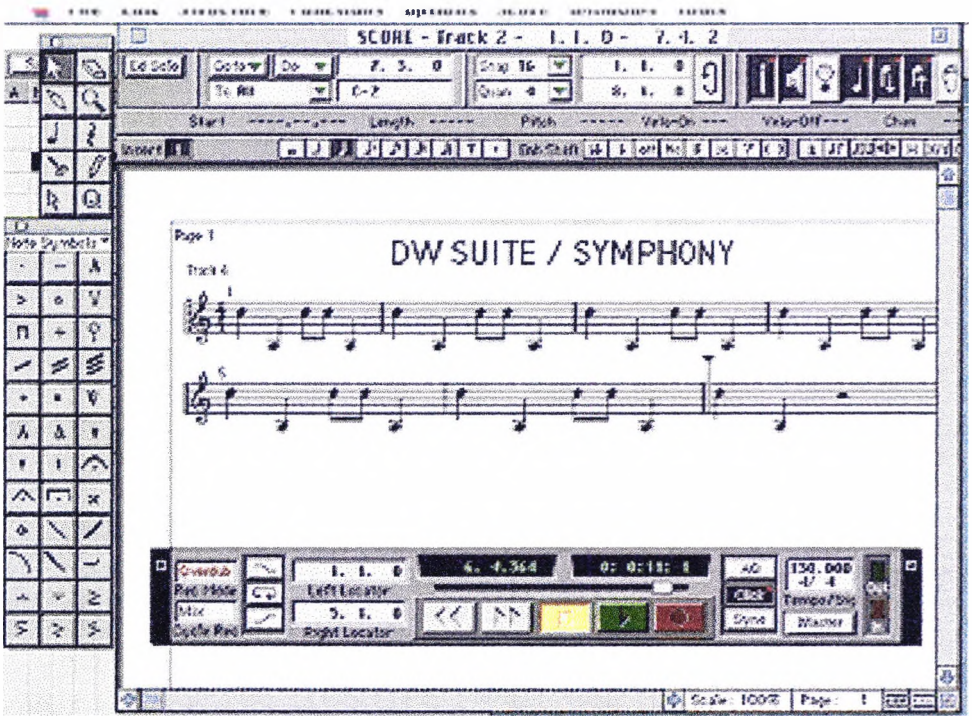



Figure 11.5 Cubase Score Edit Screen

Taking a moment to consider the complexity of the detail on each of these screens, and the fact that editing of single details requires controlled movement and use of the mouse, and also perhaps a combination of mouse and alpha-numeric keyboard use, we begin to understand the problems faced by a physically disabled user attempting to control this environment with any degree of success, in an environment which is to facilitate creative and physical independence.

### 11.1.3 The Pilot User

The Drake Music Project Ireland student who volunteered to take part in this research was 16 year old Stephen. Stephen had been with the project during a pilot course in Dublin in 1992, and had

attended the Project workshops regularly until the time of writing. He had always expressed a very keen interest in music from a very early age as noticed by his family and monitored by his teachers. However, until his participation in Drake music workshops, his musical activities were purely passive in the face of a lack of physical means to express his musical abilities.

Stephen has cerebral palsy which makes co-ordinated and controlled movements of parts of his body impossible. He has voluntary control only of his head movement and of the lower part of his left leg. Stephen is unspeeched, and communication takes place by means of a letter board which he indicates choices on or, with people more familiar to him, Stephen will listen while the person recites the alphabet in a series of three lines (Upper A - J, Middle L - R, or Lower S - Z); he first indicates which line to recite, then waits until the letter he wants is reached, and then indicates to stop. Communication can be a slow process but this scanning method proves very satisfactory for Stephen and also allows for eye contact during conversation which is very important to him.

At the start of the project with KE:NX, Stephen had some basic knowledge of musical conventions but had a great deal of experience making music using the Atari and MIDIGRID system in improvisatory and controlled composition sessions in workshops. As most of his work was carried out using the graphics based MIDIGRID sequencing package, Stephen had little knowledge of musical score notation and conventions. His reading ability was

sufficient to understand Cubase screen information and Macintosh screens and command boxes.

From very early in his attendance at workshops, it became apparent that Stephen had a keen musical sense, and needed only the right interfacing to express his musicality. To date, he has toured England, Iceland and the United States with Drake Project teams of tutors, composers and performers, and has had several public performances of his work including a broadcast in April 1995 on BBC Radio 3, of *Fire and Ice*, a composition co-composed with the author after the Project performances and tour of Iceland.

## 11.2 System Details

The role of a Drake tutor is to hand on the tools and knowledge required in order to enable a student to become skilled in the art of composition and performance. In teaching these skills, it is imperative that the tutor interferes with the composition activity of the student as little as possible. 'Interference', in the shape of a tutor becoming a students' 'hands' to access equipment front-ends and software interface management, is only required and necessary when the composer has not got the physical ability to operate interfaces and equipment. The task of the Drake Music Project researchers is to continually strive to improve the control environment of the students attending workshops in order to make their working conditions as independent as possible.

As both a researcher and music tutor with the Drake Music Project Ireland, the author had the opportunity to take delivery of a set of specialist software, KE:NX, which is designed to provide keyboard and mouse alternatives for physical access to the Macintosh for physically disabled people. Developed by Don Johnson Developmental Equipment Inc, the software appeared to be an exciting way forward to develop single and multiple switch access to and control of the Cubase sequencer package for some of the Projects' more severely physically disabled students such as Stephen.

KE:NX is a transparent control environment which claims to be able to operate over any software which a disabled person wants to access and control on the Macintosh. The system configuration for the purposes of this research and interface environment design can be seen in Figure 11.6.

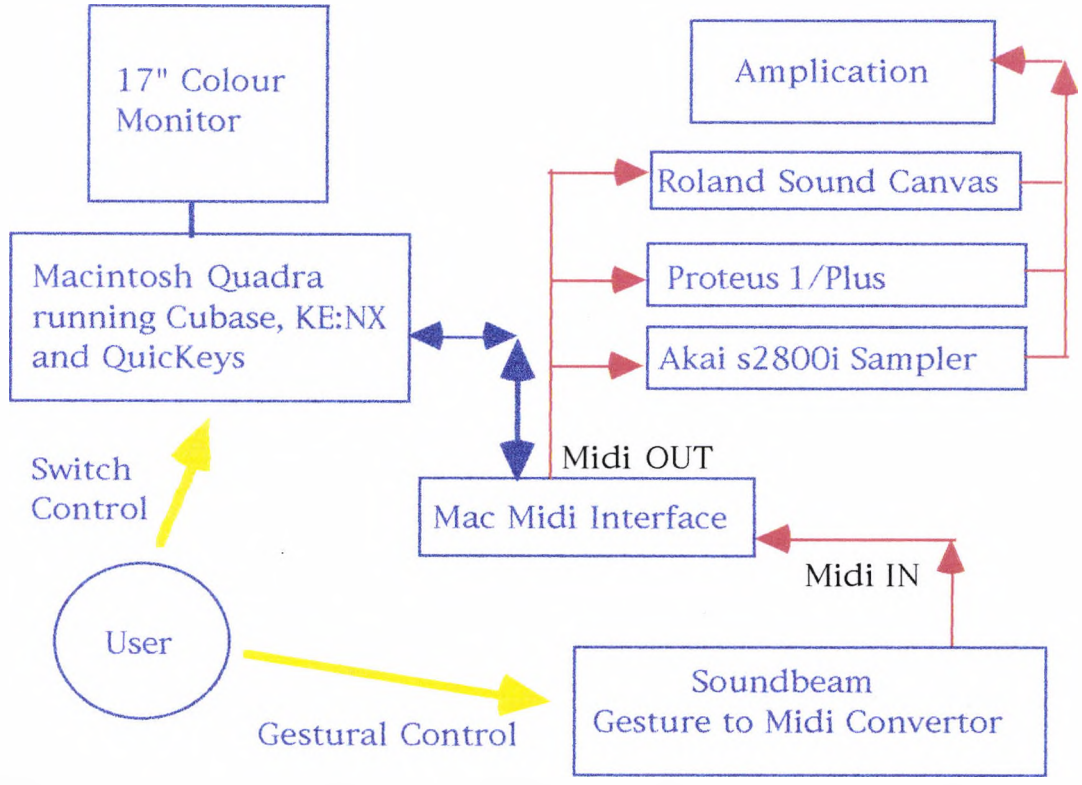



Figure 11.6 KE:NX – Macintosh System Employed

The workstation used consisted of a Macintosh Quadra with a 17" colour monitor, together with a Midi set-up using a SoundBeam as the physical controller. The KE:NX software was set up as an overlay control environment for Cubase sequencing package, with QuicKeys[1] also loaded as a macro command controller. Different sound modules, including a Roland Sound Canvas and a Proteus 1Plus were employed. An Akai sampler was also used on several occasions as part of the workstation set-up.

The KE:NX software package presents several alternative control methods for physically disabled people. These include:

- KE:NX Morse Code

As its name suggests, this set-up allows a user to control the system by entering Morse Code signals via a switch into the software environment.

- KE:NX On Screen

For users with particularly defined mouse or trackerball control, the KE:NX On Screen set-up operates by imaging the alpha-numeric keyboard to the screen. This set-up allows people who experience difficulty accessing a keyboard to control and access all its keys from this image. Two sizes of on-screen keyboards are available for people with greater or lesser refined mouse or trackerball control.

[1] QuicKeys, macro function utility software package, CE Software Inc. 1993.

- KE:NX Alternative Keyboard

This set-up allows the user to connect a different keyboard to the Macintosh in place of the standard alpha-numeric keyboard. The software will sense the type of keyboard connected and will interface successfully with it. Keyboard alternatives include enhanced size keys, or keyboards which have larger keys placed to the left side or the right which facilitates users with enhanced movement left or right. Other examples include keyboards which do not comply with the qwerty traditional key layout, facilitating people with single hand movement or similar limited physical control to have more commonly used keys available in better physical positions on the keyboard layout.

- KE:NX Scanning

With this control method the user activates a switch into the computer system. As a result of the click message received a palette of buttons is displayed on the screen, constantly scanned through at a user determined speed. When the scanning reaches the button the user wants, he hits the switch again and the function assigned to that button is carried out. These functions will have been set up in advance by the user or the system programmer within the KE:NX Create environment in which individual user palettes can be created by assigning functions and corresponding icons or words to KE:NX palette buttons. Each button may contain one function or a series of functions in a 'batch'. In order to be able to 'batch' commands, the software program *QuicKeys* was also used in the system research and design. This software allows for comprehensive batching of commands and also

for the user to record sequences of commands to subsequently assign to one key stroke. As limited physical movement relates directly to the speed at which several commands may be executed serially, a batch function of this type proves very useful to speed up operations and to decrease user frustration with his access time and execution time on a system.

There is also the possibility within the KE:NX scanning environment of 'nesting' palettes, to allow the user to move to extended sets of buttons or functions within the system, to provide palette control for other programs and also control over communication palettes and numerical palettes to speed up specific data input to dialogue boxes etc..

In order to get a user started with the scanning environment and to give him an idea of the operation of the scanning screens and the possibilities for personalisation, KE:NX contains a scanning demo screen, seen in Figure 11.7 *as the user sees it* on screen.

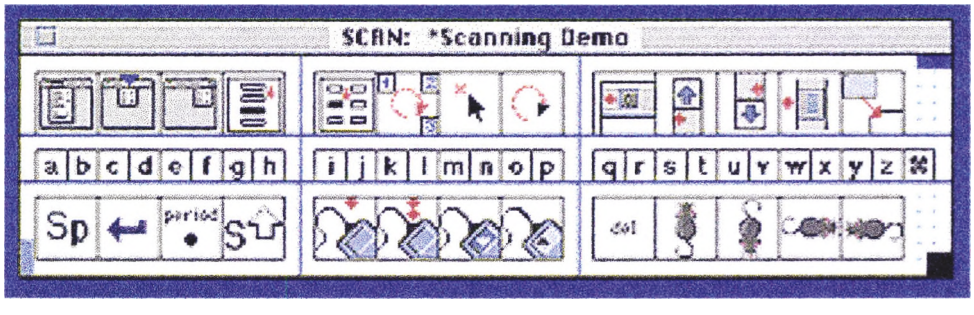



Figure 11.7 KE:NX Scanning Demo Palette *as the user sees it.*

The details in this scanning demo palette become clear when we view the palette *as the user hears it* being scanned, as in Figure 11.8:

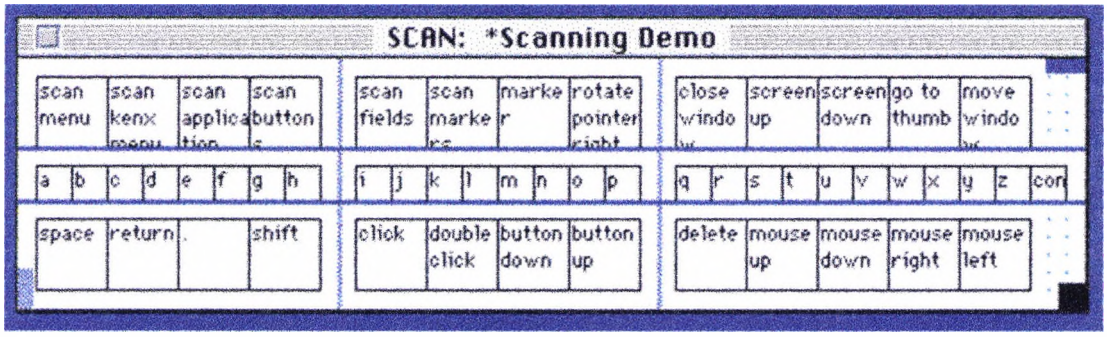



Figure 11.8 KE:NX Scanning Demo Palette
*as the user hears it.*

KE:NX Scanning will accept several types of functions to be assigned to a single button on the user's individual palette. In the words of the program literature:

> "By pressing a switch, the computer user activates an array of functions displayed as words and/or icons. Computer functions such as keyboard, computer functions and mouse scripts are represented in the display by icons, icons with words, words or letters."[2]

The types of functions which are assigned to buttons in a palette, either graphically, in words, or a combination of both, may include:

- Keystrokes

This may be a single letter or number, to be the equivalent to hitting that key on the alpha-numeric keyboard, or a combination of keystrokes, such as ⌘B, ⌘X, Shift T, etc.

---

[2] KE:NX Reference Manual, USA 1992, pg. xiv.

- KE:NX Scripts

These are a set of mouse and window related functions, replacing the physical use of the mouse on a surface to interact with the Macintosh screen area. Scripts such as 'mouse up', 'mouse right', 'rotate mouse 90 degrees', 'close window' etc. are all programmed into a palette in the form of scripts. There is a library of graphical icons to choose from to represent these scripts, or if the user is feeling artistic, he can design any graphic he wishes which he may feel better represents any script for himself.

- Branch Commands

In order to allow for nested palettes, a branching facility is provided. This allows the user to tell KE:NX to close the currently active palette, and open another one, pre-specified by the user or the programmer in the branch command built into the original palette.

KE:NX affords the user the opportunity to actually 'hear' the scan as it is carried out. This facility is invaluable for users with a visual disability. Speech options are provided in one of two ways:

- Digitised Speech

This is a set of sentences provided in the software environment such as 'mouse up', 'double click', or 'close window'. These can be very useful not only for people with a visual disability but also for someone perhaps working with a user on a personalised palette

who may not recognise or be familiar with the icons the user has loaded on his palette.

- Text to Speech

This facility converts text segments selected by the user into sounding words.

One very useful feature of the KE:NX control environment is that, with enough time and patience, a disabled user should be able to create his own personalised palette of control buttons to scan over any software program he wishes to access and control. In the design currently being discussed however, Stephen had limited access to computer time and wanted to gain control of the Cubase software as soon as possible in order to get on with the task of composition. The author acted as a KE:NX 'programmer', together with a fellow researcher volunteering with the Project who also investigated the apparent handshaking and multi-tasking problems presented by the Cubase software when operated in conjunction with KE:NX for any prolonged period of time.

## 11.3 User Analysis.

Over a period of some five months, working together with Stephen only for a short time once a week, a personalised control environment was designed to allow him to independently control first a word processing program and then the Cubase sequencing package.

In order to begin to understand the type of controls which Stephen would want to have access to via a scanning palette, it was first necessary to carry out user and task analyses to assess the nature of the interface required. For the purposes of this design task, the author employed the user analysis models discussed in Chapter 3 of this thesis, adapted suitably for the task in hand. Issues including user cognitive response, cognitive load, reaction time, fatigue levels and intellectual ability were evaluated prior to beginning to evaluate the tasks the user wished to carry out with the system.

### 11.3.1 Physical Input Method

From over two years working knowledge of Stephen's' physical working space, the issue of what type of switch input to the KE:NX environment to employ was easily answered. As he had used a single switch for almost all his life to operate a communications board, Stephen was keen that the KE:NX scanning input should also be single switched. He operated a single switch very comfortably and also very adeptly behind his head, to the left side just behind the ear. The issue of how many switches were to be employed to control KE:NX was one of the first questions which had to be satisfactorily settled, as all subsequent switching and programming information related to this number.

From a physical control point of view, the issue of Stephen's eyesight had to be considered due to the physical

distance he would be away from the computer monitor, as his physical shape, involuntary arm movements and size of wheelchair dictated that he could not actually be positioned under a table of any sort. Stephen wears glasses to correct short sightedness, and with his glasses on, he can be positioned comfortably distant from a monitor no less than 15" in screen size and still control the screen without undue stress on his eyes.

As a musician used to working with monitor loudspeakers and never headphones which cause physical discomfort, Stephen has very acute hearing and has no difficulties monitoring the speaker outputs even when this output is set quite low.

### 11.3.2 User Cognitive Ability.

From a working knowledge of the user in question and from family reports and school records, it was clear that Stephen's intelligence is not affected in any way by his physical disability. However, in this case, some cognitive limitations had to be taken into account as a result of the limited opportunities which Stephen had to participate in main-stream education and standard communication practices. As a designer who had worked with Stephen for several years, it was clear to me that cognitive limitations in relation to word recognition particularly of unfamiliar musical terms and of programming terms needed to be taken into account. These were only considered in the early stages of the design task, as exposure to these terms on a regular basis removed the cognitive load on

Stephen very quickly. In order to minimise cognitive load in general, it was agreed to attempt to aim for consistency across all the palettes within the KE:NX environment personalised for Stephen, as this would decrease the slope of the necessary learning curve and the constant cognitive load on him as he concentrated on Cubase operated through the 'transparent' KE:NX control palette.

From a cultural and social perspective Stephen had a lot of experience of computer games, so icon design and differentiation would prove to be both straightforward and also one of the design tasks which Stephen himself would undertake very willingly.

### 11.3.3 User Action - Reaction Time.

In an evaluation of the 'action-reaction' communication flow between user and machine within the Macintosh front end and the Cubase software environment, Stephen had no delayed reaction time to system prompts or to perceiving changes in system state. The delay time in extending a response physically to the system is set by the time taken for the actual scanning to occur to the button the user wishes to select. Also, some delay resulted from Stephen becoming frustrated with the wait for scanning to happen across the palette and subsequently selecting the wrong button as a fatigue-related error. It became apparent from this finding that the use of 'batch' commands lying under single buttons would speed up operations for Stephen, and also that careful placing of buttons

with a high use priority on the upper left of the scanning palette would also speed up software access.

The nature of Stephen's disability, and the likelihood of involuntary body spasms increased the possibilities of accidental button selections from switch presses at undesired times. As a result of this, it was decided to make retracing steps from error selections as simple and as fast as possible in order to decrease user frustration. Also it was decided to keep the number of button presses to a minimum in order to reduce the probability of errors and also to maximise the action-reaction time in performance. Consistency was aimed for across the palettes with regard to colour, icon choice and placing of icons common to subsequent palettes, in order to reduce cognitive load as much as possible as Stephen moved either deeper into or further out of nested palettes.

## 11.4 Task Analysis

As a Drake Music Project composition and performance student, Stephen had a set of tasks in relation to the Cubase sequencing package which a composition tutor familiar with the Cubase environment could map and serialise for analysis and KE:NX task assistance. In Figure 11.9 one global task which Stephen wished to be able to achieve is described in terms of a Jackson Structure. From this diagram, the author and Stephen were able to decide at what stages the KE:NX environment could provide independent control and at what stages Stephen would still require assistance

from someone else. Within the Jackson Structure detailed in Figure 11.9 two stages could not feasibly be controlled by Stephen, the initial power-up of the workstation and the physical adjustment of the Soundbeam – the beam length setting and tuning. These tasks remained with the workshop tutor and will not be referred to again in further task analysis in this section.

At this stage it is important to point out that only a small percentage of the tasks to be achieved can be described in detail, and that palette design and creation depended on the amalgamation of several task analyses, Jackson Structure definitions and data flow presentations of tasks within tasks, together with constant prototype palette testing. The task chosen for description in detail in this research involves Stephen creating a Cubase sequenced track of recorded improvisation using the SoundBeam as the Midi controller.

From the global Jackson Structure seen in Figure 11.9 several data flow diagrams were drawn up to begin to sectionalise the macro task into a series of smaller subtasks. The screen dumps of the Cubase track screen to be controlled at this stage were studied in detail and several options for physical orientation around the screen to achieve some of the subtasks were considered; these were then taken and mapped onto a KE:NX related data flow of suggested button selections to carry out these tasks. Figure 11.10 and 11.11 are examples of the series of data flow diagrams constructed at this stage, together with the suggested KE:NX series of buttons set to achieve the task.

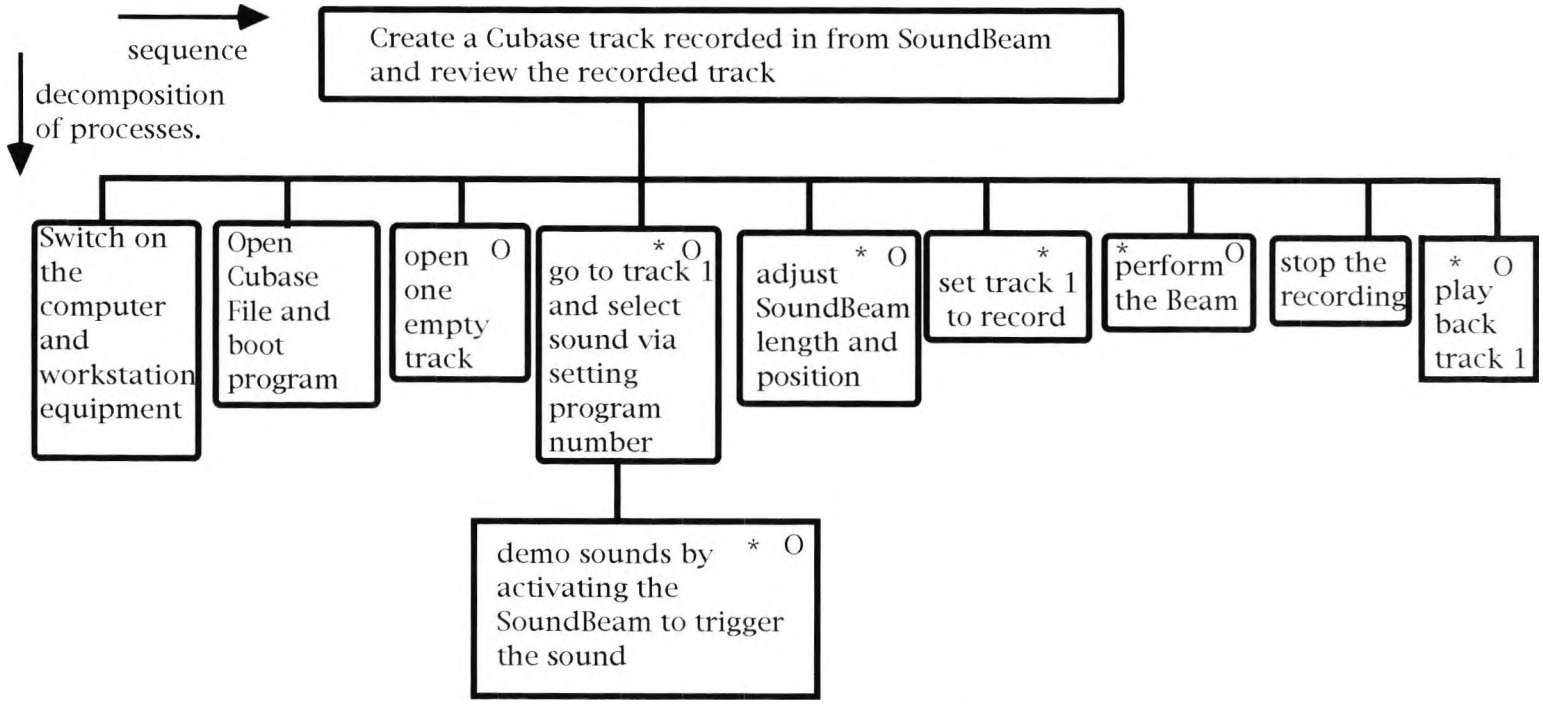



Figure 11.9 Jackson Structure of Test Task.

Taking the second stage of the Jackson Structure in Figure 11.9, – *Open Cubase File and boot program* – and breaking this task into a series of subtasks in a flow diagram, we get a clear understanding of the accuracy of the movements required by a switch user in order to achieve the superficially simple action of opening the appropriate file on screen and executing the Cubase sequencing program.

As the flow diagram in Figure 11.10 shows, the user has a great deal of recursive activity to carry out, continually re-activating the KE:NX scanning palette to select the next action on screen. Together with this repetitive action, the user has also to cope with inevitable error in pinpointing selection of items on screen. Although this error factor decreases with persistent practise, the user has to develop a 'rhythm' in the use of KE:NX, with the speed of scanning carefully selected and the user practising a selection speed in conjunction with this. Errors in selection of palette icons in the early tests with Stephen proved very frustrating for him, and only time and practice helped to reduce this error factor, and create a flow of action – reaction, relating screen scanning across the palette to user selection of icon within the palette.

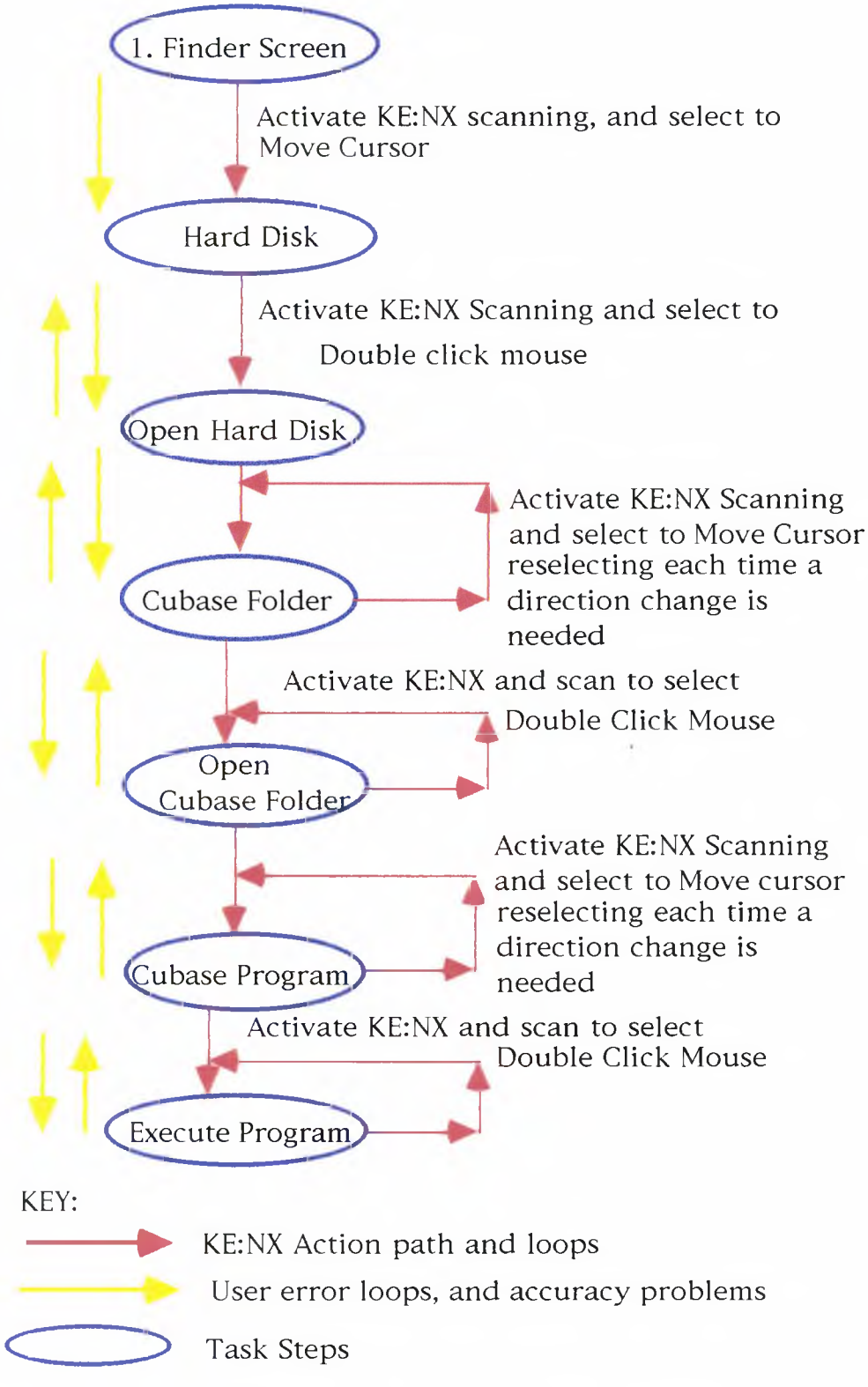


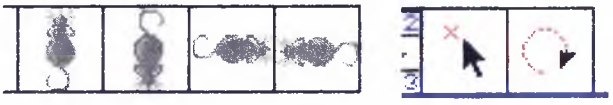

Figure 11.10 Flow Diagram to execute Cubase from Macintosh Power-up.

In Figure 11.11, in a KE:NX-related flow diagram, we can see the series of KE:NX common buttons which may be required to facilitate the task of opening a Macintosh folder and executing a program within that folder.

Finder Screen - move cursor to Hard Disk icon

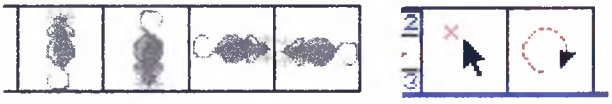
Double click to open hard disk icon, or scan menu, then click

Resize or move window on screen as required



Move cursor to Cubase file left right up or down, or rotate mouse,

click to open Cubase file



Move cursor to select Cubase program

Double click to execute Cubase program

**Figure 11.11 KE:NX Buttons for Program Execution Task**

From the list of buttons in Figure 11.11, it becomes clear that a straightforward task such as executing a program becomes, for a switch user, a long series of serial button selection from a scanning palette. The task of designing a palette suited to the user's requirements and matched closely to his cognitive skills and switching speed is one which requires much planning and testing.

## 11.5 KE:NX Palette Design.

The creation of a personalised palette with nested palettes was carried out in the KE:NX Create application, where a user and/or programmer can edit already existing palettes from the KE:NX library of icons, or create entirely new ones. Other features within Create include creating new icons, recording sounds for speech feedback and setting additional input options.

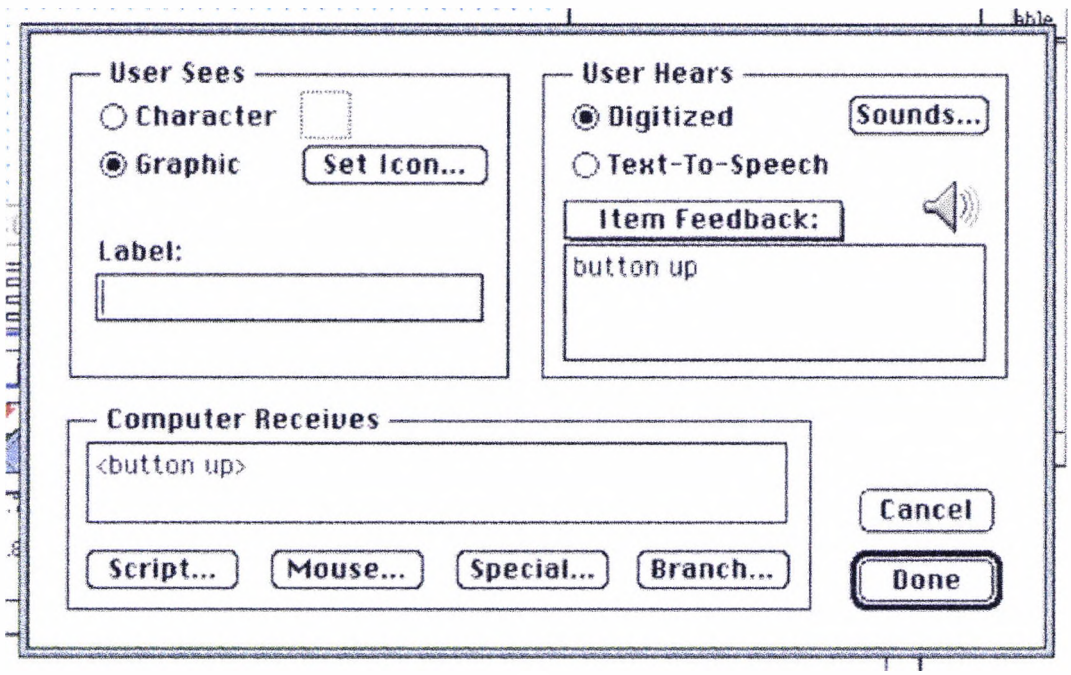



Figure 11.12 KE:NX Create Palette Design Area.

In Figure 11.12, the main working area within KE:NX Create that the user and/or designer employs during palette creation is illustrated. Within this data input box, the user specifies or chooses to create an icon (which he will later position on the KE:NX palette) together with its associated computer script, and also the aural feedback for the icon which the user hears during scanning. Based on the user and task analysis already carried out, the process of palette design for global control, communications and letter selection, numeric selection and also Cubase-specific control was started. Based on analysis, the following objectives were adhered to in all palette design and prototyping:

- Keep the number of scanning selections required to achieve an action to a minimum – this decreases possibility of error and frustration.

- Keep the cognitive load on the user as low as possible, to reduce the likelihood of error and also to decrease the likelihood of detracting from the compositional objectives at hand.

- Keep the palette designs consistent, in order to provide a working area which is familiar to the user and requires little interpretation between palettes.

- Retain as much similarity between icons employed on the palette and the icons employed within the software to be scan controlled. This will aid recognition by the user and decrease cognitive load by allowing the user to work within a palette

which mirrors the corresponding icon in the software package which it is scanning over.

- In early prototypes, as much flexibility was provided as was feasible, in order to allow the user to make an informed choice with regard to function control and mouse orientation, before a particular mouse and click combination of icons and controls was finally decided upon and carried across all the nested palettes consistently.

In order for Stephen to participate in the palette design process, and also to allow him to test simple procedures, a global control palette was required for him to start any session with, and to return to during palette prototype updates. Figure 11.13 details the scanning palette created for Stephen to use when beginning a work session, and for global screen control and access.

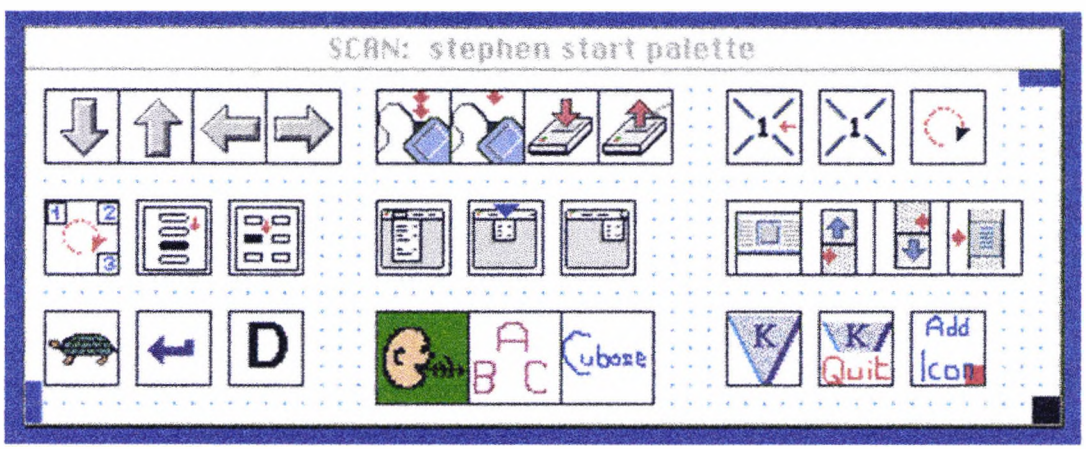



Figure 11.13 User's Global KE:NX Palette

Using the palette, the KE:NX scanning occurs row by row from top to bottom. The user clicks on a row when it is highlighted, then the scanning goes across that row in sets of icons as they are

grouped. Once the user has selected a group of icons as it is highlighted, then scanning is restricted to that group, scanning a single icon at a time until the user selects a particular individual icon. In this global palette working in sections as the scanning would occur, and labelling the rows A, B and C, and the icons 1, 2 etc, Stephen has scanning control as follows:

A1 – A4 – mouse direction control

A5 – A6 – mouse double and single click control

A7 – A8 – mouse button hold and release control

A9 – A12– set and go to screen markers, an 'invisible' tag defined on-screen for the mouse to go to a particular place quickly, useful when the user knows a particular function is always at the same position on the screen.

B1 controls to scan markers

B2 – B3 control to scan buttons and fields on screen for Macintosh dialogue box control and text input

B4 – B6 global menu scanning control

B7 – B10 window moving and sizing control

C1 – C3 mouse slow speed control, return and delete controls

C4 – C6 branching commands to specifically designed communications palette, spelling palette and Cubase palette

C7 – C9 controls to activate KE:NX Create, to Quit KE:NX Create, and to add an icon to the palette within KE:NX Create.

The placing of each icon with regard to row position was decided by Stephen together with the author based on the user analysis contained earlier in this chapter, but also an assessement of the user agility for click switch control in conjunction with the speed of KE:NX scanning.

The main Cubase control palette which was designed together with Stephen and which branches from the global palette can be seen in Figure 11.14. This palette was designed as far as possible employing common Cubase symbols and also maintaining consistency with the other KE:NX palettes used by Stephen (KE:NX global palette, Figure 11.13, and Stephen's KE:NX communications and spelling palettes).

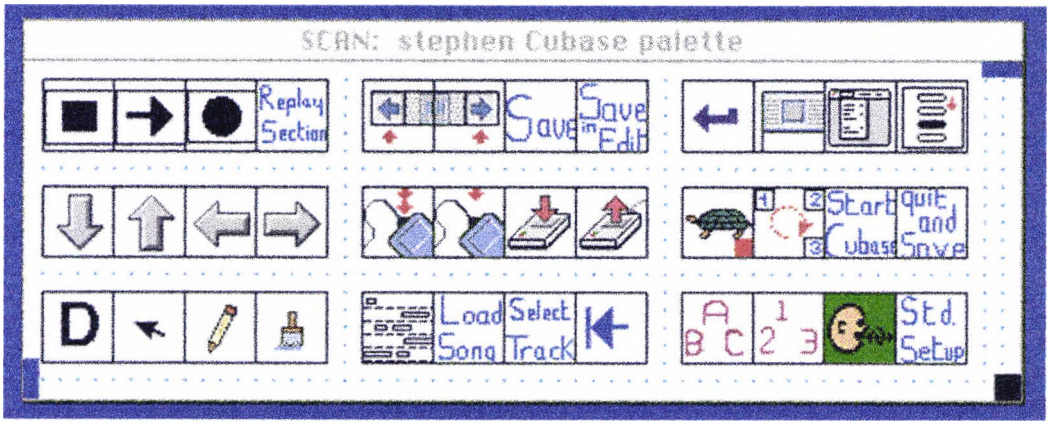



Figure 11.14 Stephen's Cubase Scanning Palette.

Within this palette, as many icons as possible were designed to bear a similarity to Cubase icons for a particular task. In the palette, in A1 – A3, for example, the author attempted to create icons for stop, play and record as they appear on the Cubase screen (see Figure 11.3). The palette icon contents and controls are as follows:

| | |
|---|---|
| A1 – A3 | Stop, Play and Record sequencing controls |
| A4 | Control to replay a section marked by cues |
| A5 – A6 | Screen scroll bar controls |
| A7 – A8 | Cubase Save and 'Save in Edit' control |
| A9 – A12 | Return, close window, scan menu and scan markers control |
| B1 – B4 | Mouse direction controls |
| B5 – B8 | Mouse click hold and release controls |
| B9 – B10 | Slow mouse and scan field control |
| B11 – B12 | Cubase functions to execute program and quit and save control |
| C1 – C4 | Delete, rotate cursor up left, pen and paintbrush control relating to Cubase |
| C5 – C8 | Open Cubase edit screen, load song, select a track and rewind to start of track |
| C9 – C12 | Icon access to branching palettes: spelling palette, numbers palette, communications palette and the standard global palette (Figure 11.13) |

The Cubase related icons in this palette have been created using QuicKeys[1] in conjunction with KE:NX. Using this program, a task such as quitting and saving could all be set under one palette icon, set up as a macro function. The reason for using QuicKeys was in order to facilitate single switch clicking to activate what would normally require a series of several mouse clicks to achieve. Setting macros under single icons where possible greatly decreased

[1] QuicKeys - the utility program for creating macros.

the time scale for task completion, and also decreased the frustration factor in using KE:NX and having to access the scanning palette to achieve every single step in a task serially. However, several problems arose in the handshaking between KE:NX, Cubase and QuicKeys which hindered the palette design process. Macros which had been created as a series of mouse movements on the screen (mouse movements which are saved as a macro, and then 'replayed' in real time when the macro is activated) were also less effective than originally hoped, as their success depended completely on the placing of buttons on the screen, and the continual use of exactly the same screen scaling and position every time the Cubase program was executed.

The main criteria for the process of individual palette design for the communications level, the global control palette level, or the actual Cubase control palette stage continually addressed the user profile built up during the user evaluations, and also was closely mapped to the task analysis already carried out. A long process of prototyping and re-drafting the palettes designed resulted in the creation of a set of control palettes which are being used currently at the time of writing by Stephen for Macintosh control of several software packages, including Cubase[2].

[2] Stephen is, at the time of writing, employing Cubase with KE:NX control with a moderate degree of success, due to continued problems with multi-tasking KE:NX, QuicKeys and Cubase.

## 11.6 Design Methodology.

The design methodology employed in this chapter is an amalgamation of the user and task analysis methods introduced in Chapter 3, together with the relevant sections of the design and analysis methodologies introduced and discussed in Chapter 4. Figure 11.15 details the interaction of all these methodologies in the design of the user interface for independent control detailed earlier in this chapter.

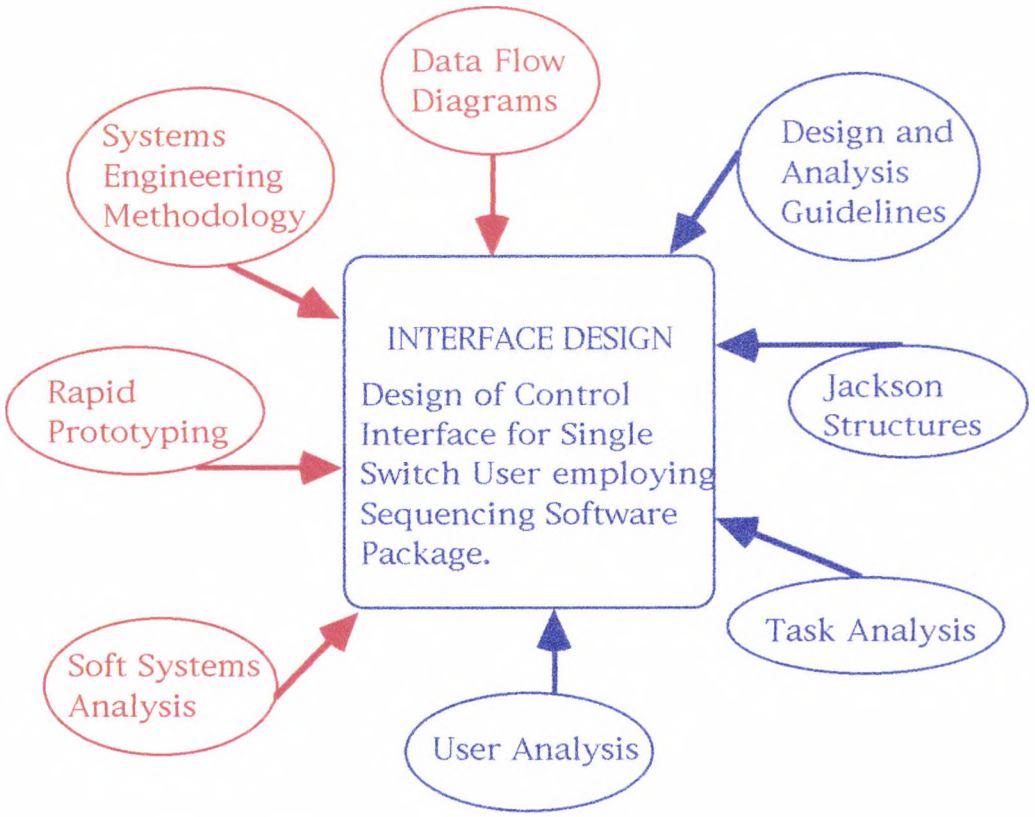



Figure 11.15 Design Methods Employed.

In this particular design case, certain elements from each methodology were found to be useful and applicable to the task. Together with applying Jackson Structure layouts to task sections, and using variations of data flow diagrams, simplified in order to involve the end user in their application, the following sections from the remaining methodologies were used.

Systems Engineering Methodology:

- formulation of the problem

*to produce an independent control environment for a single switch user wanting to employ a sequencing package for composition.*

- objectives of the system

*independent manipulation of Cubase parameters to control Midi data; independent control of Macintosh screen environment.*

- retrospective appraisal in an iterative design process

*variations of several KE:NX palettes were presented, tested, discussed and adapted by common consent in an iterative loop.*

- forecasting using accrued knowledge of user community

*the designer made several initial assumptions based on prior knowledge of the user, his action-reaction time, his task order preferences, etc. in order to start the design process by having an initial global palette in place to start the iterative design process.*

Rapid Prototyping:

- Rapid Functional Specification

*The functions that the user wished to be able to complete were specified early on in the design process, and expressed as Jackson Structures and flow diagrams for detailed analysis and breakdown into achievable steps.*

- Creation of Functions

*The functions which the user wanted to independently achieve were created in KE:NX button steps, (see Figure 11.11). and broken down into flow diagrams to facilitate future palette design.*

- Prototype Demonstration

*The KE:NX palettes created during the design process were constantly presented to the user in a prototyping test, evaluated and submitted to further alteration based on the results of the testing, by common consent.*[3]

- User Approval

*The palette designs were continually evaluated by the user, who was involved at all times in the design process. All palettes were subjected to user testing and approval prior to being used in the final system set-up.*

[3] Common consent refers to the fact that no changes were made to the palette without consent between the designer/author, the user after extensive testing and also the Drake volunteers who were assisting in the project.

Soft Systems Analysis:

- The Human Factor

The most prevalent impact of the soft systems approach was the incorporation of the sharing of perceptions and human interaction which permeated the entire design process. Once the problem had been defined, which is where the systems engineering and the soft systems approaches combined, the soft system method was incorporated in debate about particular icons or specific design considerations. Agreed changes were constantly carried out, by common consent.

No single design methodology, user or task analysis or design guideline would, in isolation, have been sufficient to complete the design discussed in this chapter. Several elements extracted from each of the methodologies combine with the design guidelines and user and task analysis methods to successfully refine the design method employed.

Due to the nature of musical tasks, it is apparent that a computer music systems designer should have a firm generic base of common design methodologies from which he can extract ideas best related to the specific system design he is realising.

# 12. Conclusion.

> "There is no dream of a machine. I do not dream of a machine. There is no need to dream of a machine. Dreaming of tools, defining tools in a visionary way tells more about the dreamer than the dreamed-about."
>
> (Goebel, J. 1991: pg. 47)

In a presentation and discussion entitled "Open Forum on the Contemporary Nature and Needs of the Computer Music Community" presented at the Keele Computer Music Conference in 1991[1], the author presented the following points as a summary to the panel led discussion.

> " Defining an 'ideal model' for a computer music system, the ideal system should:
>
> - encompass pluralities
> - allow for contradictions
> - call for minimum translation of original *action* before providing the intuitive *reaction*
> - provide a balance between the seemingly diverse systems worlds of generality and strength
>
> (Truax, 1985)

[1] International Computer Music Weekend, a Conference at the University of Keele, 26-27 January, 1991.

- provide an interpretative hierarchy so reducing mental load, allowing for high level of interaction, and encouraging optimisation
- encompass a high level of in-built intelligence, with a high degree of resultant automation, thereby reducing the user's need to handle low-level information and data.
- provide a balance between mental and conceptual models, so that the interface is optimised for intuitive user-friendliness."

(Mc Cormack. 1991).

This presentation and panel discussion had marked the beginnings of research and investigation by the author into the development cycle of a computer music environment. Some five years later, upon completion of this research to date, it has become apparent that the aim for a 'Dream Machine', which was the subtitle of the Keele paper, has been usurped by the realistic aim towards the supply of a selection of interfaces which can be available to suit a wide variety of users. In the design of a music system, it is apparent that no single design will fulfil all the requirements of all musician users.

From the preceding chapters dealing with systems analysis, each system discussed had some unique ways of accomplishing musical tasks. Each system had merits for various elements and presented problems to the user for other reasons. No single system from a collection of six of the more renowned

computer systems in this decade presented itself to the user as the single solution to any and all tasks the user may need to carry out. Each had a defined area of application within the wide field of computer music composition. The clear indication is that only a continuing development of a range of options for the computer music systems user would be the optimum path ahead.

Although we can state that variety is the best route to support, we can nonetheless draw some global conclusions regarding the design of computer systems in a general sense. Based on user analysis and task definition, systems for music composition application will be the optimum combination of :

- The User's Mental Model
  what is inside the user's head

with

- The Conceptual Model
  what the designer thinks is inside the user's head

Knowledge on the designer's behalf regarding the nature of the user community he is designing for, together with the user expressing his mental model as clearly as possible will result in a fertile discourse between designer and user. Neither the designer nor the user should exist in isolation. One of the outstanding issues to arise from this research is the increased need for communication. All of the design methods which may be employed, in part or in combination, call for communication between the designer and the end-user community throughout the design

process from inception to iterative prototyping and evaluative testing.

Reviewing the statements made to conclude the Keele paper, we can derive some suggestions for systems design in a generalised way, not however relating these to the outmoded goal of an ideal model.

In a global sense, a computer music system design should aim to:

- *encompass pluralities.*

A system designed even for the most specific of tasks should encompass plurality with regard to the task completion methods encoded in the interface; it should allow for user choice within the task path and not dictate artistic direction by virtue of its limited functional application.

- *allow for contradictions.*

Users should be able to alter task paths, and successive users on the same system should be able to work in individual ways, not being controlled or artistically restrained as a result of the interface design. Successive composers output from a particular environment should not necessarily be identifiable as being realised on the same system.

- *call for minimum translation of original action before providing the intuitive reaction.*

Users should be able to work with systems using as much natural language as possible, and adhering as closely as possible to the model of an action as the user perceives it without requiring extensive translation before execution.

- *provide a balance between the seemingly diverse systems worlds of generality and strength.*

A powerful interface should not automatically mean a strong tendency towards a particular compositional method which 'style stamps' the composition. Limitation of function in an interface should not automatically result in dictation of musical direction.

- *provide an interpretative hierarchy so reducing mental load, allowing for high level of interaction, and encouraging optimisation.*

Systems should be accessible for users without detracting too much working attention from the aural output in order to produce that output. Fast turn-around time of the action-reaction discourse will result in an open working environment, optimised for experimentation.

- *encompass a high level of in-built intelligence, with a high degree of resultant automation, thereby reducing*

*the user's need to handle low-level information and data.*

With the emphasis at all times on the musical output, the user should be required to divert as little attention from the acoustic feedback as possible, to be asked to concentrate on system details and data not relevant to the musical success of the immediate task at hand. Users (beginners and intermediate) who do not wish to become advanced systems specialists should be able to produce the intended aural outcome without extensive machine-based knowledge.

- *provide a balance between mental and conceptual models, so that the interface is optimised for intuitive user-friendliness.*

Systems designed with a continual high level of discourse between designers and users, with continual iterative testing and prototyping, will consequently be more friendly to the user community. The designer, through constant discourse, user and task analysis, and application of guidelines, will have a realistic understanding of the user's mental model. There will be few, if any, discrepancies between the user's mental model and the designer's conceptual model, as the two models will merge as a result of continual discourse, analysis and testing.

From the inception of this research to the time of writing, little has changed with regard to musicians' user interfaces. Although the hardware has become physically smaller,

less expensive, and real-time is almost always supported in signal processing applications, the actual interfacing with the user has altered very little. Sequencers have shown no marked improvement in interfacing techniques. The mouse and the alpha-numeric keyboard remain the primary sources of input and control for the user. Although large advances have taken place in the availability and control of video processing which has resulted in better graphics being available on the interfaces, advances in audio interface processing techniques do not seem to have kept pace.

Looking at the availability of 'home user' environments which have dominated the market for some time, there appears to have been a decisive move by larger company systems designers to concentrate their efforts on producing related packages which users can buy into at the entry level, and upgrade to successively more advanced levels, simultaneously gaining access to more and more features. This step-design method seems successful in securing a user as a potential customer throughout his working life, as upgrading means that the user retains a degree of familiarity with the package at all times[2], stepping up to new advanced features in measured and controllable stages. However, this step design seems simultaneously to restrict what may be more advantageous interfacing improvements as the designers strive to retain familiarity across the platform from one upgrade to the next. As a result of the upgrade-led design industry, the market is

[2] Designers of the Cubase suite of programs aim to access all users from the entry level to the most advanced computer composer and sound analyst, designing programs such as Cubase Lite through to more advanced programs with extensive features such as Cubase Audio.

restricted to evolution within interfaces, and rarely witnesses revolution.

The stimulus for this research was first and foremost an attempt to improve the working environment of the computer music composer. Several environments exist which superficially seem to make the task of composition easier, but this ease of task completion should always be measured against the quality and unique nature of the compositional output. It is hoped that by defining a design process as discussed throughout this thesis, users and designers might realise the benefits of discourse and collaboration. By applying design guidelines, employing extensive user and task analysis, and refining a suitable design methodology from an available library of techniques to suit the design task at hand, the process of systems design for musicians will be more effectively achieveable.

The availability of a comprehensive and friendly working environment will enhance the mechanics of the job of composition as it now exists. The creative impulse remains with the composer at all times, employing technology as a tool.

> "....as a musician, I do not long for machines that could work creatively in autonomy; I want to co-operate with the machine, to work in symbiosis with it, and this attitude is shared by most artists...Indeed, would it not be nice if musicians could state their desires so as to enable good designers to propose music machines that would fulfill them?.
>
> (Jean-Claude Risset, 1991, pg. 32)

Appendices and Bibliography.

# APPENDIX A – CSOUND Synthesis Methods.

- Direct Synthesis in CSOUND

This synthesis method, one of the simplest provided for by CSOUND, involves the generation of a waveform by sampling a function table which is stored in memory and is representative of one single cycle of the waveform required. The unit generators in CSOUND which may be used to implement this straightforward type of digital synthesis include, in its most elementary form, the *oscil* generator. This signal generator accesses values by sampling once through the function stored in memory. The rate at which the oscillator unit reads through the function table is determined by the duration stated on the control line of the instrument. A delay value may be provided on this line, whereby the oscillator will not begin reading through the table until the delay time has passed. It will then read through the function table at a constant rate, over the time stated for the duration value. When this time has elapsed, the oscillator will remain pointing at the last location in the function table. The output is obtained when the table output value is multiplied by the user specified value for amplitude, given on the same command line as the duration and optional delay values.

Another signal generating unit for direct synthesis within CSOUND is the *table* unit. Unlike oscil, which is restricted to use only as a control line variable, table can be used on behalf of initialisation, audio or control indices. Used on any of these lines, table will invoke a table lookup, also allowing for a wrap around value to be specified, and a start point other than at the

first table entry. The additional signal generators of *oscili* and *tablei* are interpolating versions of the originals detailed above. The interpolating generators are computationally very intensive, although they produce a much cleaner output signal.

- Additive Synthesis in CSOUND

Within the CSOUND environment, additive synthesis is provided by the *adsyn* signal generator. As the CSOUND Users Manual (1989) states:

> "...any number of sinusoids, each individually controlled in frequency and amplitude, can be summed by high speed arithmetic to produce a high fidelity result."

The audio line on which the unit generator is specified requires an in-file number that specifies both frequency and amplitude traces in breakpoint fashion. This file is itself created using the *adsyn.prg* provided in the system. At compilation, the simultaneous frequency and amplitude values are used, interpolatively, by an internal fixed point oscillator that adds each active partial to the accumulated output signal. Two other parameters on the command line allow the user to modify the amplitude and frequency of each contributing signal. Alternatively, these can be specified as control signals, allowing for amplitude modulation or frequency shifting at the specified control rate.

- Subtractive Synthesis in CSOUND

The signal generators, *buzz* and *gbuzz,* both of which operate on the audio line, output a set of harmonically related cosine partials. Arguments which must be specified on the command line for the generation of these complex signals include the fundamental frequency of a summed set of consecutively numbered cosine partials. Further manipulation is afforded in that the signals produced from either *buzz* or *gbuzz* may be either amplitude and/or frequency modulated, by either control or audio signals.

The CSOUND environment provides a complex set of filters for use in subtractive synthesis. The signal modifier, *port,* the only one of a set of five which operates on the control line, applies **portamento** to a step value control signal. The input value here is low-pass filtered at each new step value. On the audio line of an instrument design, *tone* and *reson* respectively implement a first order recursive and second order recursive low pass filter. The opposite filtering (high passing) techniques are provided by the units *atone* and *areson. Atone* is a form of high pass filter and *areson* a notch filter whose transfer functions represent the filtered out aspects of *tone* and *reson.*

- Sampling Synthesis in CSOUND

Within the CDP system, there are several programs for sampling and sample manipulation. In CSOUND, the *soundin* unit is a signal handler which derives its signal from a pre-existing sound file.

This unit opens the soundfile each time the host instrument is initialised, and closes it when the host instrument is turned off.

- Granular Synthesis in CSOUND

At a very simple level, granular synthesis can be achieved within CSOUND by the construction of very short score events, representing sonic grains, which are copied many times to produce dense textures. These types of scores are often computationally intensive.

- Non Linear Synthesis in CSOUND

Frequency modulation is achieved in CSOUND by the patching together of two or more signal generators. Also, a dedicated CSOUND unit generator, *foscil* is provided. Designed as a composite unit banking two oscillators, *foscil* produces FM synthesis results in classic Chowning fashion. An interpolating unit is also available, *foscili,* but computation time with this unit is considerably longer than with the non-interpolating one.

Waveshaping is possible within CSOUND with envelope generators such as *line, expon, linseg,* and *expseg,* and modifying units such as *linen* , applying a straight line rise and decay to an input amplitude signal, and *envlpx,* which applies a three part envelope, rise, steady state, and exponential decay to an input amplitude signal.

- FOF Synthesis in CSOUND

Formant waveform function synthesis, as developed by Xavier Rodet, in the program Chant, and implemented in CSOUND by Micheal Clarke, is available to composers with the *fof* generator.

Another synthesis technique supported by CSOUND is incorporated in a unit generator based on the Karplus Strong[1] algorithm. This *pluck* generator provides a method for producing plucked string and drum-like timbres. Two unit generators, *rand* and *randi*, (interpolating), produce random noise, and are best used as the sound sources to be fed to the *pluck* generator when creating string plucked sounds, as the sounds will be rich in harmonics, and will produce an effective plucked sound at the output.

[1] Karplus, K. and Strong, A. (1983).

# APPENDIX B – CDP CSOUND
## Signal Generators And Modifiers.

### CSOUND Signal Generators.

| | |
|---|---|
| LINE<br>EXPON<br>LINSEG<br>EXPSEG | Control or audio signal generation, tracing a straight line or exponential curve or a series of line segments between user specified points. |
| TABLE<br>TABLEI | Unit invokes a table lookup action on behalf of initialisation, control or audio indices. Tablei interpolates between adjacent table entries. |
| OSCIL | Oscillator outputs periodic control or audio signals dependant on the result returned from the specified amplitude value multiplied by the control or audio rate sampling from the function table. |
| OSCIL1<br>OSCIL1I | Oscillators which sample once through the stored table at a determined rate set in the command line. The oscili unit interpolates between adjacent values. |
| ADSYN | Additive synthesis signal generator. Any number of sine waves, each completely individually controlled (frequency and amplitude), may be added together. |
| BUZZ,<br>GBUZZ | Sound source generators used in subtractive synthesis. The output from these generators is a set of harmonically related cosine partials. |
| PLUCK | Signal generator based on the Karplus–Strong algorithms. The output consists of naturally decaying plucked string or drum sounds. |
| RAND | Random white noise generator. |
| RANDH<br>RANDI | Band limited noise generators. |

CSOUND Signal Modifiers.

| | |
|---|---|
| LINEN | Applies a straight line rise & decay to a control or audio amp signal. |
| ENVLPX | Applies a 3 part envelope to an incoming control or audio signal. |
| PORT | Control signal modification by low-pass or band pass recursive filters with a variable frequency response |
| TONE<br>ATONE<br>RESON<br>ARESON | Audio signal modification by low-pass or band-pass recursive filters with a variable frequency response. |
| LPREAD<br>LPRESON<br>LPFRESON | Read as a read-reson pair, these units modify the spectrum of an input control and audio signal. (input to lpread is a control signal, then passed to the reson unit as an audio signal). |
| GAIN | Amplitude modification of an input audio signal. |
| BALANCE | Amplitude modification of an audio signal so that its value is equal to that of a comparator signal given on command line |
| DOWNSAMP | signal conversion from an audio to a control signal by downsampling. |
| UPSAMP,<br>INTERP | Signal conversion from a control to an audio signal by upsampling. Interp uses linear interpolation between adjacent control values. |
| INTEG DIFF | Integration or differentiation of control or audio signals. |
| SAMPHOLD | Provides a sample and hold facility dependant on the value of gate, a type of on/off switch which controls the flow of samples to the output. |
| DELAYR<br>DELAYW<br>DELAY<br>DELAY1 | Highly manipulable delay paths, with specifiable delay time intervals available. Operative on audio signals only. Delayr and delayw work as a read/write pair. |
| COMB<br>ALPASS<br>REVERB | Reverberation of an input audio signal with either a coloured (comb), flat (alpass), or natural room (reverb) frequency response. |
| DELTAP<br>DELTAPI | Units to tap into delay lines at various offset times. Used in conjunction with a delayr and delayw pair. |

APPENDIX C – GROUCHO

## Sample Editing Program Suite

Program available in the CDP Groucho Editing set are listed, together with the command lines necessary for operation.

| Set 1 – Editing and Mixing Programs | |
|---|---|
| CUT | extract a portion of a soundfile<br>CUT I/F O/F START END |
| ENVFOLL | envelope superposition<br>ENVFOLL I/F1 I/F2 O/F |
| ENVEL | envelope sounds according to breakpoint function<br>ENVEL I/F O/F FUNCFILE |
| MIXSF | soundfile mixing with pan placement<br>MIXSF [-S] [-H] O/F MIXFILE |
| MIXTWO | simple mixing of two soundfiles<br>MIXTWO I/F1 I/F2 O/F |
| SPLICE | chain soundfiles together with variable splice<br>SPLICE [FLAGS] O/F I/F1 I/F2 ...ETC |

| Set 2 – Filtering programs | |
|---|---|
| ALLPASS | allpass filter.<br>ALLPASS [-F] I/F O/F TYPE FREQ C/B [BW] [PRESCALE] |
| EQ | equaliser of mixing desk type.<br>EQ [-F] I/F O/F TYPE FREQ C/B [BE] [PRESCALE] |
| FLTBANK | filter bank.<br>FLTBANK I/F O/F |
| FSTATVAR | function controlled state variable filter.<br>FSTATVAR [-F] I/F O/F [FUNCFILE] |
| LPHP | low or high pass filtering.<br>LPHP I/F O/F |

| Set 3 – Sample Manipulation Programs | |
|---|---|
| DELAY | delay line with feedback.<br>DELAY [-F] I/F O/F |
| FTRANS | transpose soundfile according to function.<br>FTRANS [-F] I/F O/F [FUNCFILE] |
| LOOP | loop portion of sf, brasssage and time manipulation.<br>LOOP [-F] I/F O/F O/FLENGTH LOOPSTART LOOPLENGTH [LOOPSTEP] [SEARCHFIELD] |
| PAN | pan according to function.<br>PAN I/F O/F FUNCFILE [PRESCALE] |
| REVERSE | reverse soundfile.<br>REVERSE I/F O/F |

| Set 4 – Utilities Programs | |
|---|---|
| CHANNEL | extract channel/s from multi-channel file.<br>CHANNEL [-F] I/F CHAN_NO [CHAN-NO....] |
| CONVERT | convert sample type (floats or shorts).<br>CONVERT I/F |
| GAIN | scale amplitude by fixed factor.<br>GAIN I/F O/F GAIN |
| GLITCH | extract single sample glitches.<br>GLITCH I/F O/F THRESHOLD |
| MAXSAMP | find maximum amplitude value in soundfile.<br>MAXSAMP I/F |
| SIGNAL | generator of test signal soundfile.<br>SIGNAL [-F] O/F |
| SPECT | apply FFT to a sample stream.<br>SPECT [-F] I/F [O/F] |
| STOM | mixes stereo file to mono.<br>STOM [-F] I/F O/F |
| VIEWSF | graphical display of sf, controllable definition.<br>VIEWSF S/F |

| | |
|---|---|
| SPECE | stretches the spectrum of an input sound sample. Does not shift the spectrum. |
| SPECSTR | time-variable time stretching of analysis data, employing a breakpoint file. |
| SPECBLUR | time averages the spectra in an analysis file. |
| SPECMAX | takes max value in each channel from input PVOC analysis files. |
| SPECTRAC | retains only the N loudest partials in a file. |
| SPECCALC | relates channels to frequencies. |
| SPECCROS | replaces spectral envelope of one analysis file with another. Degree of replacement may be time-varyingly controlled by a breakpoint file. |
| SPECLIFN | interleave N spectra in analysis file. |
| VOCINTE | interpolates between the values of two analysis files. |

Phase Vocoder SFPVOC Program Flags

| | |
|---|---|
| -N | Twice the number of channels wanted in the analysis windows |
| -F | Analysis sampling rate, which is the equivalent of the sample rate of the input sound divided by twice the number of channels you want to use ie: the -N value. It follows that -N and -F cannot be used together. |
| -b, -e | If not all the original sound file is to be analysed, then the user can specify beginning and end time values on the command line. |
| -A | Use of this flag generates an analysis data file which subsequent sound manipulation programs can access. |
| -S | This flag acts as the reverse to -A, resynthesising a sound file from a data analysis file. |
| -E | Generates data on the spectral envelope of a sound, window by window. |
| -T | This flag will result in a time stretch or contraction of the original sound without altering its pitch. |
| -P | This flag will cause a pitch transposition of the original sound file without actually altering the duration of the file. |

# APPENDIX E Syter Public Instrument Library.

| | |
|---|---|
| ACCHAR | Accumulator with 4 harmonisers |
| ACCHARNEW | Accumulator with harmonisation variation |
| BRASSALQD | Aleatoric brassage |
| DELAIST | 2 second stereo delay with feedback |
| DOPPLER | Doppler effect on 4 outputs |
| ENVERS | Inversion by threshold detection |
| EQUAL8 | 8 band equaliser, from -50 to +12 dB |
| ETIRPHST | Stretching by phase detection |
| ETIR4 | Stretching and contraction (mono) |
| FLT8 | Bank of 8 resonant filters. |
| FMCL6 | 6 voice polyphonic FM module. |
| FORMONDE | Harmonic spectrum synthesiser (mono). |
| GEL4 | Placing a sound sample in four loops. |
| GEL4QDCL | As above, with independant control+MIDI usage. |
| GLISS | Mono infinite glissando |
| HAR7ST | 7 parallel harmonisers |
| MAG | Empty, for recording/listening to sample. |
| MODFORST | Graphic amplitude modulator. |
| OSC8 | Bank of 8 mono oscillators. |
| PEIGNE | Comb filter bank. |
| PEIGNECL | As above, with independant control. |
| REVGLISS | Reverberater, with frequency variation. |
| REVST | Reverberater. |
| RINGLFOST | Ring modulator with LFO. |
| RINGST | Ring modulator. |
| SPECTRE | FFT visualisation. |
| TESTVU | Test tone. |
| TRAME | Bank of 24 FM generators. |
| TRAMECL | As above, with envelopes. |
| VAR | Speed variation. |
| VAROSC | As above, with oscillator control. |
| VIDE | Invisible system initialisation instrument |
| KEY: ST = STEREO QD = QUAD OUTS CL = MIDI CONTROLLABLE | |

# APPENDIX F POD-X System Program Library.

| | |
|---|---|
| GSX, GSAMX, GRMSKX, PDFILG. | Granular Synthesis Programs, real-time command-line controlled. Three synthesis models are provided. Within these granulation models, the user has the further choice of controlling either the density of the granulated texture in real-time or the delay time between successive grains in the output texture. This choice relates directly to the setting, during program initialisation, of either the quasi-synchronous or the asynchronous method of granulation. A delay value of 0 means maximum density in the output texture. |
| PLAYDK | Synthesis of sound samples on disk via DMX with signal processing. |
| WAVEX | Waveform generation, display and storage. User toggles on CRT to adjust parameters. Stored waves tested with a variety of synthesis models. |
| POD6X | Compositional program with extensive sound definition and a spectrum of Aleatoric to Deterministic controls to create and modify files. |
| SCORE | Hard copy POD6 file listing in Coded Standard Music Notation. |
| MERGE | Mixes files, with the added option to create choral effects. This program is also used to translate POD6 files to MERGE files. |
| POD7X | Allows for synthesis of single sounds, file objects and MERGE files in mono or binaural stereo modes. Real-time object testing is also included in the options available within this program. |
| PDFILX | Copying, editing playback and listing of MERGE, POD6, TRAJ and TUNING files. Useful for more general compositional needs. If necessary, the user can begin composition within this program by creating a dummy MERGE file. |
| POD7 | Non-real-time calculation of MERGE files; samples stored on disk. |
| POD7F | Non-real-time calculation of FUNC(tion) files; samples stored on disk. |
| FUNC | 8-voice additive synthesis, amp and freq controlled by user defined functions. |
| CONDUC | Real-time synthesis of 1-6 files with user controlled performance variables. |
| PLOTX | Plots a spatial trajectory on CRT |

| | |
|---|---|
| PDMSKX | Generates files of vertical density scores from frequency time masks. Four co-ordinates and a duration value are specified to create a mask. The first two coordinates define the upper and lower frequency limits at the start of the mask, and the third and fourth values specify the upper and lower values for the end of the mask. The duration specifies the overall length of the mask. In operation, scores are created from the time-point values read from the mask. |
| FMTUT | On-line tutorial on FM synthesis with exercises and quizzes. |
| CFM, CFM7X | Configurable FM programs (Oscillator/sound object versions). |
| KSX | Karplus-Strong algorithm performance program. |
| GSKX | Chaotic non-linear systems of real-time granulation with DMX-1000. |

## Books

BACCHETTI, A. and FAURE, C. (1989). Vers Une Culture de l'Interactivité. (Paris, C.S.I).

BALABAN, EBCIOGLU, LASKE, (Editors). (1992). Understanding Music with AI. (California, AAAI Press).

BATEMAN, W. (1980). Introduction to Computer Music. (New York, Wiley).

BEAUCHAMP and FOERSTER, (Editors). (1969). Music by Computers. (New York, John Wiley).

BORWICK, J. (1987) Sound Recording Practice. (Oxford, Oxford University Press, 3rd Edition).

BUTLER, D. (1992). The Musician's Guide to Perception and Cognition. (New York, Schirmer Books).

DANIELS, A. and YEATES, D. (1988). Basic Systems Analysis. (UK. Pitman).

CARD, S.K, MORAN, T. P. and NEWELL, A. (Editors). (1983). The Psychology of Human Computer Interaction. (Erlbaum).

CARROLL, J. M. (1988). Cognitive Aspects of Human-Computer Interaction. (Mass., MIT Press).

CHAMBERLAIN, H. (1985). Musical Applications of Microprocessors. (USA, Hayden).

CHECKLAND, P. B. (1981). Systems Practice, Systems Theory. (London, Wiley).

CHECKLAND, P. B. (1990). Soft Systems Methodology in Action. (Chichester, Wiley).

COLLINS, N., RYAN, J. and WAISWISZ, M. (Editors). (1993). Steim. (Amsterdam, Steim).

DAVIES, J. (1978). The Psychology of Music. (London, Hutchinson and Co.).

DODGE, C. and JERSE, T.A. (1985). Computer Music - Synthesis, Composition and Performance. (New York, Schirmer Books).

FALZON, P. (1990). Cognitive Ergonomics. (London, Academic Press).

GUNTON, T. (1990). Inside Information Technology. London, Prentice Hall).

HAUS, G. (Editor). (1993). Music Processing. (Oxford, Oxford University Press).

HUNT, R. and SHELLEY, J. (1988). Computers and Common Sense. (London, Prentice Hall. 4th Edition).

JACKSON, M.A. (1983). Systems Design. (London, Prentice Hall).

KELLEY, A and POHL, I. (1984). A Book on C. (California: The Benjamin/ Cummings Publishing Company, Inc).

LAYZELL, P. and LOUCOPOULOS, P. (1989). Systems Analysis and Development. (London, Chartwell-Bratt).

LINDSAY, P. H. (1977). Human Information Processing. (London, Academic Press).

LONG, J. and WHITEFIELD, A. (1989). Cognitive Ergonomics and Human- Computer Interaction - An Introduction. (Cambridge, Cambridge University Press).

McCORMICK, E. and SANDERS, M. (1982). Human Factors in Engineering and Design. (New York, McGraw-Hill).

MOORE, F. R. (1990). Elements of Computer Music. (London, Prentice-Hall).

NAUGHTON, J. (1984). Soft Systems Analysis. (London, Open University Press).

NORDOFF, P. (1971). Therapy in Music for Handicapped Children. (London, Gollancz).

OPEN UNIVERSITY (1989). Open University Press Module T301 -Block IV. Soft Systems Analysis: An Introductory Guide. (Open University Press).

PATCHING, D. (1990). Practical Soft Systems Analysis. (London, Pitman).

PELLMAN, S. (1994). An Introduction to the Creation of Electroacoustic Music. (California, Wadsworth).

POPE, S.T. (1991). The Well-Tempered Object. (Mass. MIT Press).

PREECE, J. AND KELLER, L. (1990). Human-Computer Interaction. (Prentice Hall in association with the Open University).

PRIESTLEY, M. (1985). Music Therapy in Action. (St. Louis, USA, MMB Music).

ROWE, R. (1993). Interactive Music Systems. (Mass., MIT Press).

SALVENDY, G. (Editor). (1987). Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems. Human Factors / Ergonomics Series. (Amsterdam: Elsevier Science Publishers B.V ).

SANDERS, M and McCORMICK, E. (1993). Human Factors in Engineering and Design. (Mc-Graw Hill International Editions).

SHNEIDERMAN, B. (1980). Software Psychology: Human Factors in Computer and Information Systems. (Boston, Brown).

SUTCLIFFE, A. (1988). Human-Computer Interface Design. (London: Macmillan Education Ltd).

## Articles

AMES, C. (1991). Introduction to COMPOSE: An Editor and Interpreter for Automated Score Generation and Score Processing. In Interface, Journal of New Musical Research 20 (3-4): 181 - 196.

ANDERTON, C. (1994). STEIM. In Keyboard, August 1994: 55 - 62.

ASHLEY, R. (1991). Two Responses. In Computer Music Journal, Vol. 15 (4): 55 - 61.

BAIRD, B., BLEVINS, D and ZAHLER, N. (1993). Artificial Intelligence and Music: Implementing an Interactive Computer Performer. In Computer Music Journal, Vol. 17 (2): 73 - 79.

BERNARDINI, N. (1991). Should Musical Instruments Be Dreams? In Computer Music Journal, Vol. 15 (4): 78 - 81.

BOULEZ, P. and GERZSO, A. (1988). Computers in Music. In Scientific American, Vol. 258 (4): 44 - 50.

BUXTON, W. (1985). In PENNYCOOK, B. (1985). Computer Music Interfaces: A Survey. In ACM Computing Surveys: Special Issue on Computer Music. Vol. 17, (2). June 1985.

CARD et Al, (1988). In Sutcliffe (1988). Human-Computer Interface Design. (London: Macmillan Education Ltd).

C.D.P. (1989). Advertising Material and Promotional Literature. (York, C.D.P).

C.D.P. (1989). Composer's Desktop Project Workstation Documentation. (York, C.D.P).

CHAFE, C. (1991). Dream Machine 1990. In Computer Music Journal, Vol. 15 (4): 62 - 64.

CHOWNING, J. (1973). The Synthesis of Complex Audio Spectra By Means of Frequency Modulation. In Journal of the Audio Engineering Society Vol 21 (7), 1973. Reprinted in Computer Music Journal Vol 1 (2), 1977.

CHOWNING, J. (1981). Computer Synthesis of the Singing Voice. In Proceedings, International Conference on Music and Technology, Melbourne.

DEMEYER. T. (1995). BigEye Reference Manual. (Amsterdam, Steim).

DEPRES. A. (1989). Le SYSTEM UPIC. In Vers une Culture l'Interactive. Cite de Science et de l'Industrie, La Parc de la Villette, Paris.

DOLDEN, P. (1987). PDFILX Documentation. In Online System Documentation, SFU, Vancouver.

GOEBEL, J. (1991). My Dream (Machine?). In Computer Music Journal 15 (4) : 47 - 50.

HARRIS. C.R. (1987). A Composer's Computer Music System: Practical Considerations. In Computer Music Journal, Vol. 11 (3): 36 - 43.

JENKINS. (1969). In Open University Press module T301 - Soft Systems Engineering.

JONES, D.L., and PARKS, T.W. (1988). Generation and Combination of Grains for Music Synthesis. In Computer Music Journal, Vol. 12 (2): 27 - 34.

KARPEN, R. (1994). In POPE, S.T. Touched by Machine? - Composition and Performance in the Digital Age. In Computer Music Journal 19 (3) : 13 - 17.

KARPLUS, K. and STRONG, A. (1983). Digital Synthesis of Plucked-String and Drum Timbres. In Computer Music Journal 7 (3) : 43.

KAY, A. (1976). Personal Dynamic Media. IEEE Computer March 1977: 31 - 41. In PREECE, J. AND KELLER, L. Ed. (1990). Human-Computer

Interaction. (Chapter 11). (Prentice Hall in association with the Open University).

KIERAS, D. and POLSON, P. G. (1985). An Approach to the Formal Analysis of User Complexity. In International Journal of Man-Machine Studies, Vol. 22, (4): 365 - 394.

KIRAKOWSKI, J. (XXXX). Human Computer Interaction - From Voltage to Knowledge. In

KREFELD, V. (1990). The Hand in The Web: An Interview with Michel Waisvisz. In Computer Music Journal, Vol. 14 (2): 28 - 33.

LANDAUER, T. (1988). Relations Between Cognitive Psychology and Computer System Design. In: Human- Computer Interaction. Ed. by J. Preece and L. Keller. (Prentice Hall in association with the Open University).

LASKE, O. (1991). Composition Theory: Introduction to the Issue. In Interface, Journal of New Musical Research 20 (3-4) : 125 - 136.

LOHNER, H. (1987). The UPIC System - A User's Report. In Computer Music Journal, Vol. 11, (3).

McCORMACK. M. (1991). Towards Tomorrow - The Future for Interface Design and Implementation for Computer Music Composition. Presented at Keele Computer Music Conference, 1991.

MARINO. G., RACZINSKI, J-M, and SERRA, M-H, (1992). A Description of the UPIC System. In Proceedings, International Computer Music Conference and Festival, Delphi, 1992.

MATHEWS, M., MOORE, F., and RISSET, J-C. (1974). Computers and Future Music. In Science, 183: 263 - 268.

NIELSEN, J. (1990). Traditional Dialogue Design Applied to Modern User Interfaces. In Communications of the ACM Vol. 33: 109 - 118.

NORMAN, D. A (1988). Cognitive Engineering - Cognitive Science. In: Human- Computer Interaction. Ed. by J. Preece and L. Keller. (Prentice Hall in association with the Open University).

PAPE, G. (1992). Some Musical Possibilities of the New UPIC System. In Proceedings, International Computer Music Conference and Festival, Delphi, 1992.

PAPE, G. (1992). Studio Report - Les Ateliers UPIC 1992. In Proceedings, International Computer Music Conference and Festival, Delphi, 1992.

PENNYCOOK, B. (1985). Computer Music Interfaces: A Survey. In ACM Computing Surveys: Special Issue on Computer Music. Vol. 17, (2). June 1985.

PIERCE, J. R. (1991). Final Comments. In Computer Music Journal, Vol. 15 (4): 87.

POPE, S. (1993). Music Composition and Editing by Computer. In HAUS, G. Ed. (1993). Music Processing. Oxford University Press.

POPE, S. T. (1995). Touched by Machine?- Composition and Performance in the Digital Age. In Computer Music Journal, Vol. 19 (3): 13 - 17..

PUCKETTE, M. (1991). Something Digital. In Computer Music Journal, Vol. 15 (4): 65 - 69.
RISSET, J-C. (1991). Some Comments About Future Music Machines. In Computer Music Journal, Vol. 15 (4): 32 - 36.

ROADS, C. (1988). Introduction to Granular Synthesis. In Computer Music Journal, Vol. 12 (2): 11 - 13.

RODET, X. (1991). What Would We Like to See Our Music Machines Capable of Doing? In Computer Music Journal, Vol. 15 (4): 51 - 54.

SCHMIDT, B.L. (1987). A Natural Language System for Music. In Computer Music Journal, Vol. 11 (2): 25 - 33.

SCHOTTSTAEDT, B. (1977). The Simulation of Natural Instrument Tones using Frequency Modulation with a Complex Modulating Wave. In Computer Music Journal, November 1977: 46 - 50.

SENSORLAB. (1993). Steim Advertising Material. (Amsterdam, Steim).

TRUAX, B. (1976). A Communicational Approach to Computer Sound Programs. In Journal of Music Theory, Vol. 20 (2).

TRUAX, B. (1977). The POD System of Interactive Composition Programs. In Computer Music Journal, Vol 1 (3), 1977; excerpt from A Communicational Approach to Computer Sound Programs (1976), In Journal of Music Theory, Vol 20 (2) 1976.

TRUAX, B. (1977). Organisational Techniques for C:M Ratios In Frequency Modulation. In Computer Music Journal, Vol. 1 (4): 39 - 45.

TRUAX, B. (1978). Computer Music Composition: The Polyphonic POD System. In IEEE Computer, Vol. 1 (8): 40 - 49.

TRUAX, B. (1978). Polyphonic Timbral Construction in Androgny. In Proceedings, International Computer Music Conference, Evanston.

TRUAX, B. (1982). Timbral Construction in Arras as a Stochastic Process. In Computer Music Journal, Vol. 6 (3): 72 - 77.

TRUAX, B. (1983). The Compositional Organisation of Timbre in a Binaural Space. In Proceedings, International Computer Music Conference, Fastman School of Music, 1983.

TRUAX, B. (1985). The POD-X System: Interactive Compositional Software for the DMX-1000. In Computer Music Journal, Vol. 9 (1): 29 - 39.

TRUAX, B. (1986). A Case Study of Computer Music Composition: Spatial Timbral Control in Solar Ellipse. In Proceedings, The Wired Society, The Music Gallery, Toronto, 1986.

TRUAX, B. (1987). Information Regarding the Polyphonic System and It's Use. In Online System Documentation, SFU, Vancouver.

TRUAX, B. (1988). Real-Time Granular Synthesis with a Digital Signal Processor. In Computer Music Journal, Vol. 12 (2): 14 - 26.

TRUAX, B. (1989). Composing with Real-Time Granular Sound. In Department Documentation, SFU, Vancouver.

TRUAX, B. (1990). Real-Time Granular Synthesis with the DMX-1000. In Software Documentation for POD-X System, Simon Fraser University.

TRUAX, B. (1990). Time-Shifting of Sampled Sound with a Real-time Granulation Technique. In Department Documentation, SFU, Vancouver.

TRUAX, B. (1991). Capturing Musical Knowledge in Software Systems. In Interface, Journal of New Musical Research 20 (3-4) : 217 - 233.

TRUAX, B. (1994). Discovering Inner Complexity: Time Shifting and Transposition with a Real-Time Granulation Technique. In Computer Music Journal, Vol. 18 (2): 38 - 48.