Chapter 6

Multiparametric Interfaces

Overview

This chapter explains the background, purpose and construction of the user interface trials to test the hypothesis given in Chapter 5. It clarifies what is meant by multiparametric interfaces and gives details about their characteristics. Emphasis is given to the *mapping* between human gesture and the resulting changes in synthesis parameters.

The particular multiparametric interface used in the test is then described along with the other interfaces in the study. The musical task for the trials is then outlined and compared with some alternative tasks which may form the basis of further work. Finally there is a description of the MIDAS system which was used to build the test framework, the interfaces and the data gathering/analysis system.

6.1 Aim of the Study

The purpose of the tests was to study the effectiveness of different interfaces when used for a real-time musical control task. The data gathered was used to compare how a group of human test subjects performed in the exercise. Particular emphasis was given to comparing the results from different interfaces *over a period of time*. In other words the aim of the tests was to gather a set of data which measures how people respond to a range of interfaces and shows how that performance varies over time.

At least one of the interfaces chosen for the task needed to represent the commonly accepted way of controlling a system, so this used a mouse to select individual parameters and alter them. At least one of the other interfaces needed to be a more radical design which allowed the user to control multiple parameters at the same time in an explorative (holistic) manner.

6.2 Multiparametric Interfaces and Mapping Strategies

Section 5.4 stated that in order for a human to explore a device in 'Performance Mode' we would require that:

> The control mechanism is a physical and multiparametric device which must be learnt by the user until the actions become automatic.

and that:

> There is no exclusive 'set of options' (e.g. choices from a menu) but rather a set of continuous controls.

In other words for a device to permit Performance Mode to occur it needs to allow the user to have continuous control of several parameters at the same time.

Such multiparametric interfaces are rare in the computing world, but are abundant in the world of mechanical devices such as musical instruments and vehicles. Two particular concepts are now discussed which I believe are the key to the design and development of richer interfaces for computing systems.

- Multiple parameters should be coupled together
- The system should utilise the human operator's energy.

6.2.1 Parameter Mapping in conventional interactive systems

Consider a violin and ask the question “where is the volume control?”. There is no single control, rather a *combination of inputs* such as bow-speed, bow pressure, choice of string and even finger position. This is an example of a ‘many-to-one’ mapping, where several inputs are needed to control one parameter (see Figure 6.1). Rovan et al [1997] refer to this as *Convergent Mapping*.

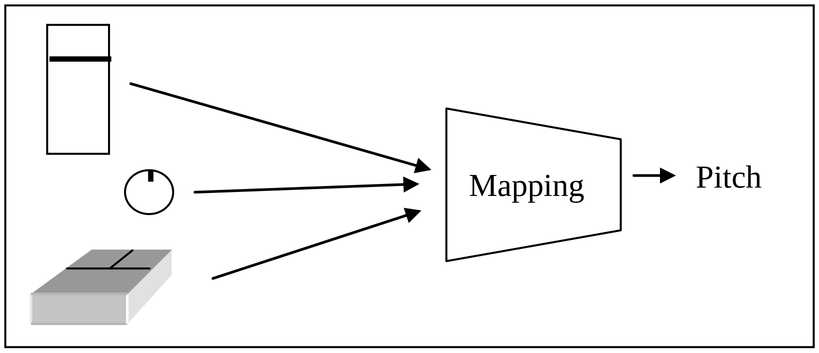

Figure 6.1: Convergent Mapping; Many controls operate one parameter

Again, considering the violin, ask the question “which sonic parameter does the bow control?” It actually influences *many* aspects of the sound such as volume, timbre, articulation and (to some extent) pitch. This is therefore an example of a ‘one-to-many’ mapping (see Figure 6.2). Rovan et al call this *Divergent Mapping.*

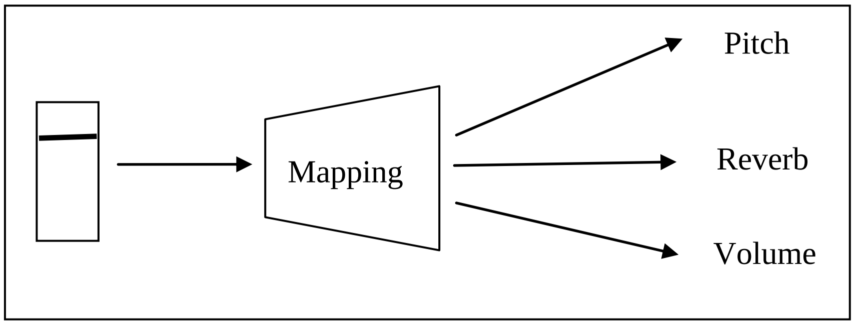

Figure 6.2: Divergent Mapping: One control operates many parameters

Human operators *expect* to encounter complex mappings, and yet so often engineers provide nothing but ‘one-to-one’ correspondences (for example a set of sliders, each controlling a single synthesis parameter).

6.2.2 Use of Energy in conventional interactive systems

In many real-time devices (for example a violin, a bicycle, a clarinet or a drum-kit) the human operator has to *inject energy* or ‘excite’ the system before it will operate, and must continue to supply energy to keep it going. Then, the energy is *steered* through the system or *damped* (dissipated) in order to achieve the task such as playing a note or climbing up a hill.

These two operations (inject/excite & steering/damping) are often carried out by different conscious body controls (e.g. bowing with one arm and fingering notes with another, pushing bicycle pedals with the legs and steering with the arms, blowing a clarinet and using the fingers to key the notes). Even in motorised systems (the car being the most common example) the concept of injecting energy with one limb and steering with another holds true. A motor actually generates the energy, but its injection and damping is controlled by the driver.

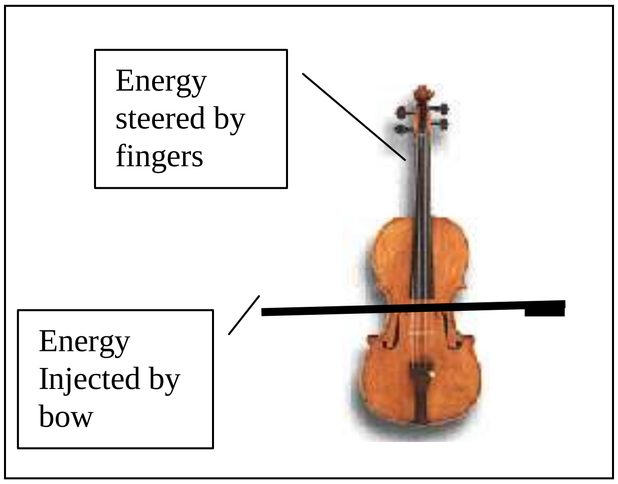

Figure 6.3: Human energy input and control

6.2.3 Further discussion of Mapping Strategies

The incoming parameters from the control device can be mapped in a variety of ways onto the synthesis variables. For example, two inputs can be *summed* or *averaged* to give a single output, suitable for the control of one synthesis variable. In this way both inputs have an effect on the resultant sound parameter. In the example of a violin, the pitch is controlled by a combination of the effects of finger position (on the string) and bow pressure. The finger position is the predominant effect, so we can also consider the *weighting* of each input parameter. The equation for the violin pitch might look like this:

Pitch = (finger position x (large weighting)) + (bow pressure x (small weighting))

Note how this is a convergent mapping (more than one input controls a sound parameter), but there are also *simultaneous* divergent mappings taking place in a violin. The bow pressure (which contributes to the pitch, as shown above) also has an effect on the volume and the timbre of the sound, each of which is affected by a range of other inputs. We can therefore see that the resulting mapping of input parameters to sound parameters in a traditional acoustic instrument resembles a ‘web’ of interconnections.

Rovan et al [1997] describe this process in more detail for an acoustic clarinet. They introduce a new form of gestural interaction known as *biasing*. This is more than just a summing function; it describes the situation where one parameter needs to be activated to a certain level before another one can even begin to have an effect. The example cited is of a clarinet’s embouchure which needs to be set to an appropriate level before the ‘air-flow’ parameter has any effect.

There are many possible strategies for mapping. Mappings may be arbitrary, perhaps done purely for the convenience of the technical designer or programmer. They may be experimental, involving user trials to see which combination of inputs gives the best sense of control for the user. To begin with, it is sensible to emulate some of the mappings that are common to acoustic musical instruments. These, at least, have stood the test of time and are well understood by many musicians.

Serafin et al [1999] describe the issues involved in controlling a physical model of a string by user input gestures. They use a stylus to allow the user to perform pen-strokes in a similar manner to the bowing action of a string player. The rate of change of the ‘bow’ position (i.e. the bow velocity) is used to determine the overall sound level, whereas the orthogonal bow position (up and down the string) is used to determine the tonal quality. Many-to-one mappings are introduced by using the downward bow force to inflect both the volume and timbre. Interestingly the designers use a second stylus to allow the performer to control pitch (continuously, as on one string) with the other hand. They comment on the effectiveness of these mappings. Essentially the user’s energy primarily controls volume, but also inflects timbre.

Some mappings are very subtle indeed. Wanderley et al [1999] describe how a clarinettist’s ‘non-obvious’ gestures (e.g. swaying with the instrument) contribute a great deal to the overall effect of the performance. They show that gestures such as these are not restricted to enhancing the visual effect of the live performance, but are integral to the quality of the sound. Displacements in the sound source (in this case, movement of the clarinet) give a continuous variation in early reflections from the room (which cause filtering effects and amplitude modulation). Such mappings are subtle because they concern gestures that are not directly related to sound generation, but nevertheless are present in nearly all live instrumental performances.

The total effect of all these convergent and divergent mappings, with various weightings and biasing, is to make a traditional acoustic instrument into a highly non-linear device. Such a device will necessarily take a substantial time to learn, but will give the user (and the listener) a rich and rewarding experience. Many computer interfaces concentrate on simple one-to-one mappings and, though easier to learn, can give an impoverished experience to the player and listener. The experimental work described in this thesis attempts to define a relatively simple interface that exhibits some of these complex mapping strategies.

6.2.4 Design of Multiparametric interfaces for the tests

The characteristics identified in this report for allowing Performance Mode are thus:

- Continuous control of many parameters in real time
- More than one conscious body control (or limb) is used
- Parameters are coupled together
- User's energy is required

With this type of operation each physical device (for example a slider) controls a variety of parameters. In other words there is no longer a one-to-one mapping between a control device and a system parameter. There will be more than one control device for the user to control simultaneously (for example, several sliders and a mouse). The parameters are **grouped** under each control device such that each control device has a distinct characteristic (for example, one pedal in a car "increases engine speed" whereas the other "slows down the wheels").

To illustrate this imagine a design where moving the mouse to the left decreases the pitch *and* softens the timbre, whilst moving the mouse upwards (i.e. pushing it forwards) increases the volume *and* decreases the reverb level. In this example a circular mouse movement would control pitch, timbre, volume and reverb level, all at the same time.

The way that the parameters are grouped will affect which areas of the system parameter space can be covered. In our imaginary example it is impossible for the user to have a very loud sound with a high reverb level.

The purpose of the user interface experiments is to see how human users react to having grouped parameters which steer them away from an analytical one-to-one control/parameter mapping to a more holistic performance exploration of the parameter space.

6.3 Choice of Interfaces for comparison

A range of different interfaces was originally envisaged for the tests. The intention was to allow the user to perform a real-time task on those interfaces which are commonplace in computer music and to compare these with a new multiparametric interface operating on the principles outlined in the above section.

The following three interfaces were chosen for the study:

- A set of on-screen sliders controlled by a mouse.
- A set of physical sliders moved by the user's fingers.
- A multiparametric interface which uses parameter coupling and the user's energy.

They represent a series of stages – from the most commonly accepted through to the most radical.

In order for the tests to be compared in a fair manner it was important to define what parameters were being controlled and to have exactly the same parameters in each of the interfaces. All the sounds were made up of the following **four** parameters:

- Pitch
- Volume
- Timbre
- Panning

Each of the chosen interfaces is now described. Following this is a discussion of those interfaces which were initially considered, but were *not* chosen for the final tests.

6.3.1 Mouse Interface

This interface consists of four sliders on a computer screen, one for each of the sonic parameters that can change. During the test, the player uses the mouse to move the sliders (see Figure 6.4).

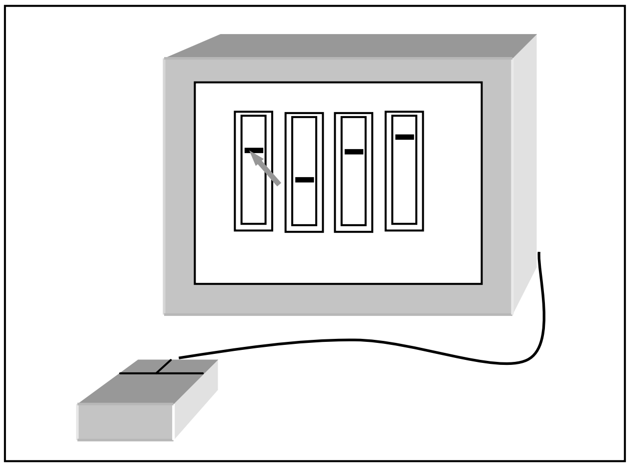
Figure 6.4: The Mouse & Sliders Interface

Each slider control can be ‘dragged’ to produce a trajectory or ‘clicked’ to produce a step change. The mouse can even be moved left and right *across* the bank of sliders and if the button is held down, each slider will 'snap' into position as the mouse is passed over it.

Some initial pre-trial studies of this interface showed that the only way it could be made viable was to ‘pre-set’ the starting positions of each slider control to correspond with the starting values of the sound in question. At least this way, the user had some chance of reproducing the sounds. Otherwise, too much time was spent trying to set each of the sliders into an appropriate starting position.

Note also that this interface is not a ‘true’ WIMP interface as it has no menus. The user does not have to *search* for the parameters, but simply move the mouse to the appropriate sliders. An interface with menus would actually slow down the whole process so as to make interactive continuous control impossible (see section 6.3.4 for further discussion).

This interface does not allow the user to change more than one parameter at a time, so it is not a multiparametric interface. The following section describes an interface which *does* allow the user to operate more than one of the parameters simultaneously, whilst remaining conceptually similar.

6.3.2 Sliders Interface

This interface uses four of the sliders on a Roland SC-155 sound module (see Figures 6.5 and 6.6). This was configured to send out MIDI information and thus control the sound algorithms on the MIDAS computer system.

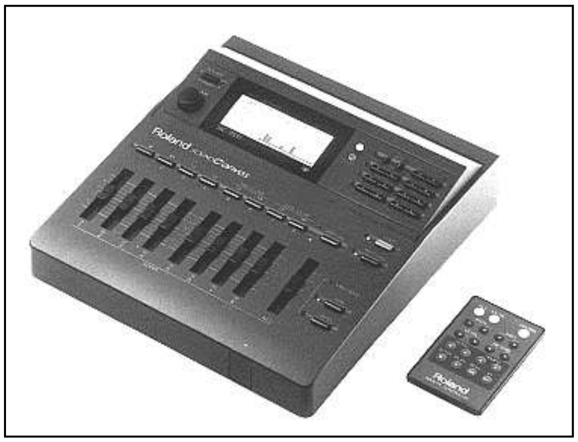

Figure 6.5: The Roland SC-155 Sound Module

Each slider controls a single sound parameter (i.e. a one-to-one mapping). The user can move each of the sliders independently and can thus simultaneously control all four sound parameters. The slider positions are also shown on the screen, but the user does not *need* to look at the screen in order to use this interface.

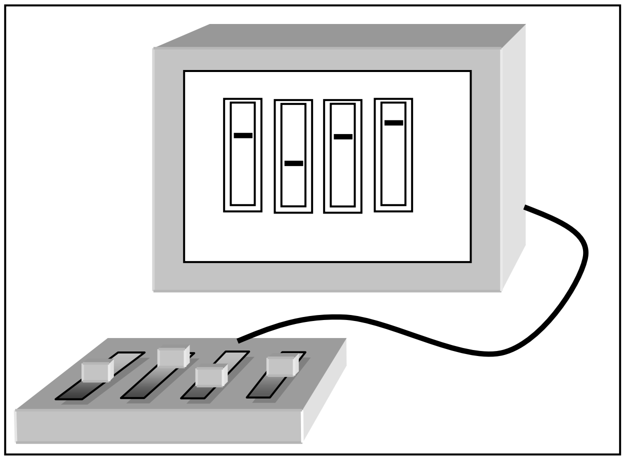

Figure 6.6: The Physical Sliders Interface

Note that this sliders-based interface fulfils the first two requirements of a multiparametric interface outlined in section 6.2.3 (i.e. 'many parameters' and 'more than one body control'), but not the final two ('coupled parameters' and 'use of energy'). Of course the user expends a small amount of energy in moving the sliders, but it is only the slider *position* that determines the state of the parameters. We therefore need to design an interface which fulfils all of the requirements in order to establish the effect of parameter coupling and energy use. The following section describes such an interface.

6.3.3 Multiparametric Interface

This interface uses the same hardware as interfaces 6.3.1 and 6.3.2 (the mouse and physical sliders on a sound module), but it uses them in two radically different ways. Firstly the system expects the user to expend some physical *energy* to continuously activate the system. Secondly, there is only one direct one-to-one correspondence (mapping) between a physical control and an internal sound parameter (panning). All other mappings are complex, as shown in Figure 6.8.

The multiparametric interface used in the study is shown in Figure 6.7. The user finds that the computer screen is blank (in contrast to the two previous interfaces where the screen shows a representation of four sliders). Sound is only made when the mouse is *moved*. The sound’s volume is proportional to the *speed* of mouse movement. This ensures that the user’s physical energy is needed for any sound to be made, and that the amount of energy has an effect on the quality of the sound.

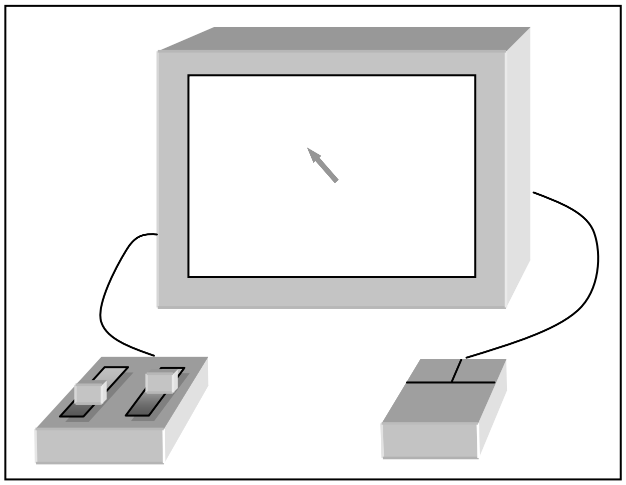

Figure 6.7: The Multiparametric Interface

The volume, pitch, timbre and panning are controlled by *combinations* of the mouse position and the position of two sliders, as shown here and in Figure 6.8:

- Volume = speed of mouse + mouse button pressed + average position of two sliders.
- Pitch = vertical position of the mouse + speed of movement of slider no. 2.
- Timbre = Horizontal position of the mouse + difference in the two slider positions.
- Panning = Position of slider no. 1.

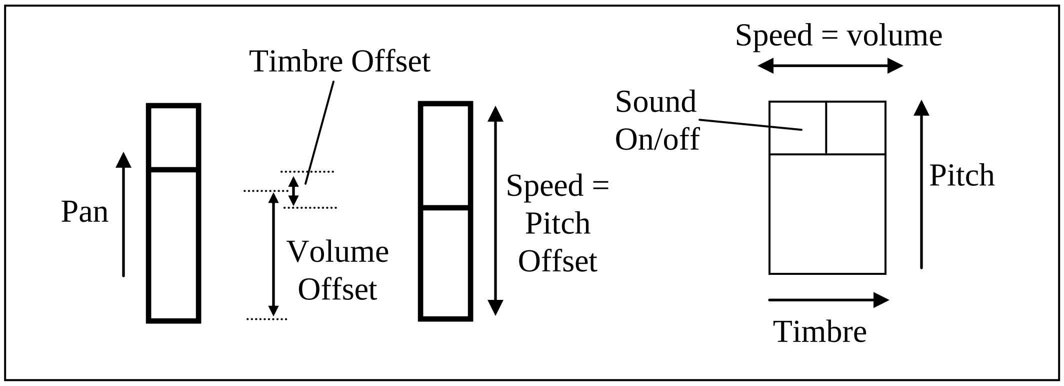

Figure 6.8: Multiparametric mappings

This ensures that there are several many-to-one mappings. Simultaneously there are various one-to-many mappings (e.g. slider 1 affects volume, timbre and panning). Two limbs are used, as the player has to use two hands – one on the mouse, one on the sliders.

There is no 'obvious' mapping of hand position to sound produced. The user must experiment. During the tests users tended to be somewhat baffled at first, because they could not find 'the volume control'. After a while (and a few verbal hints, such as 'try wobbling the mouse left and right, and listen to what happens') then they gradually developed a 'feel' for the interface. After more time most people finally began to think in terms of gestures and shapes, a holistic control, rather than a one-to-one analytical approach.

6.3.4 Rejected Interfaces

Several interface styles commonly used for computer music were rejected from the outset, as they do not allow any form of real-time interaction.

6.3.4.1 Non-real-time number lists

The available parameters are edited into a text file as numbers along with time-stamps (see Figure 6.9). The file is then compiled and played as audio. This type of interaction is very predominant in the computer music world and is the basis of the ‘score’ file concept for ubiquitous synthesis programs such as Csound (see section 3.1.1).

```
0.0     64
0.5     65
0.6     73
0.9     96
1.0     117
2.5     127
. .      . .
```

Figure 6.9: Text file of numbers and times

However there is no real-time feedback whatsoever. It is an off-line preparation system which involves thinking about the music as numbers in advance. It has nothing whatsoever to do with generating values in a performance.

While it might be technically possible to type numbers in interactively, the sheer amount of numbers needed to adequately define a parameter stream makes this impractical. Therefore it was deemed unnecessary to incorporate such an interface into those to be compared for real-time response.

6.3.4.2 Parameter selection and adjustment.

Section 3.1.4 describes a commonly encountered synthesiser interface consisting of an LCD screen with a set of Cursor Keys (see Figure 6.10). The ‘Left-Right’ keys are pressed to select the parameter being edited and then the value itself is changed by using the ‘Up-Down’ keys, or by typing in a number, or sometimes by adjusting a slider. The synthesis parameter is then updated and can be heard if a note is triggered (usually by playing a keyboard).

It was reasoned that the process of selecting a parameter and adjusting a number takes a minimum of one second, and usually a lot longer. This is clearly inappropriate for a musical task where up to four parameters need to be adjusted in a reasonably continuous manner. Therefore this interface was rejected.

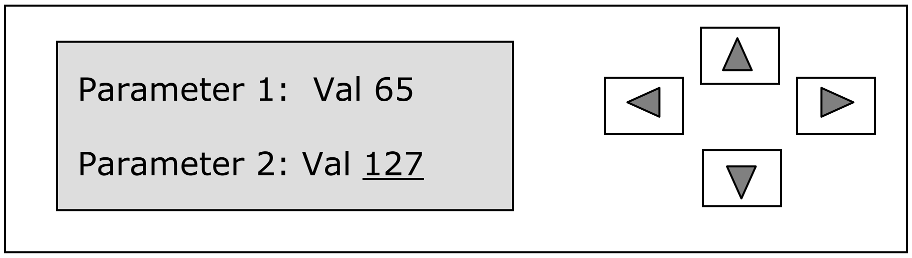

Figure 6.10 – LCD screen and cursors

6.4 Different types of Test Scenario

A number of audio tasks were considered as the vehicle for this study. All of these experiments required the user to control sounds in real time. The following sections describe the task that was finally chosen, followed by some alternatives which were not selected but which may form the basis of some future work.

6.4.1 Listen and Copy

This task requires the human test subject to listen to a musical phrase lasting no more than a few seconds and then attempt to reproduce it accurately on the interface to hand. In order to simplify the number of variables involved, the musical phrase consists of changes in up to four sonic variables (pitch, volume, timbre and stereo panning). The computer generates a 'sonic trajectory' - a moving set of the four parameters - and the user then aims to recreate that sound via the user interface.

The complexity of the phrases ranges from those involving a single change in one parameter to those containing simultaneous trajectories of all four parameters. More details of the sonic content of the tests are given in section 7.2.

6.4.2 Pitch-following

This task involves the user listening to a tone whose pitch is being controlled by the computer. The user has control of the pitch of another tone. The aim is to try to reproduce the pitch that the computer is producing.

There are several variations on this task:

i) The user has continuous control of pitch and follows a continuously changing signal. In this task we would always expect the user to be lagging behind the computer. This task could be termed *'pitch tracking'*.

ii) The user has *discrete* pitch steps (for example semitones) and follows a discrete set of pitches produced by the computer. This is another form of pitch tracking so again there will be a continual time lag.

iii) The computer changes the pitch (either to a new value in the continuous range or to one of the agreed discrete values) then the clock starts. The user has to match the pitch and then press a button when they feel they have done this. The button press stops the clock and so this could be termed a *'race against time'* task and may yield results about the relationship between 'time taken' and 'accuracy achieved'.

6.4.3 Holding a pitch steady in a dynamic system.

One of the first tasks that many teachers of 'bowed string' or 'wind' instruments ask their pupils to do is to hold a steady note. This is because it is quite a difficult feat to control a complex, unstable system and mastery of one note is needed before melodies can be attempted.

In this task, the user will have control over several parameters of a real-time simulation of an audio system (e.g. excitation pitch and amplitude, system pressure such as controlled by the covering of holes on a woodwind instrument). Other factors such as the effect of a change in air temperature on musical tuning would be simulated via the computer. The prerequisite for the simulation is that pitch cannot be controlled directly by a single input device. i.e. more than one parameter is required to change the pitch and the parameters interact in some way.

6.4.4 Reasons for selecting the 'Listen and Copy' task

The Listen and Copy task was chosen above the others because the sonic tests can be created in advance, stored on the computer and used in every session without change. This gives the facility of having a graded set of input sounds which is consistently played for all tests and for all subjects. Whilst this is also true for the Pitch-Following scenario, only the ‘Listen and Copy’ tests involve several simultaneous audio parameters.

6.5 Implementation on the MIDAS system

The University of York, UK’s MIDAS system, running on a Silicon Graphics Indy machine, was used to construct the audio algorithms, the user interfaces and the data monitoring systems. For some of the interfaces, an external piece of hardware was used. The hardware was linked to the MIDAS system via a MIDI connection.

This section describes the MIDAS system, and explains the role the author has had in its development. It then outlines how the user interfaces were constructed, before giving details on how the system was configured for each of the tasks for creating sound, gathering data and analysing results.

6.5.1 Description of MIDAS

MIDAS [Kirk, Hunt 1996] is an acronym for the Musical Instrument Digital Array Signal-processor. It was conceived by Ross Kirk in the early 1990s as a test-bed for real-time performance control of audio signal processing algorithms and for associated studies in Human-Computer Interaction. In this respect it was intended as a follow-on from MidiGrid (see chapter 3) to allow the construction of user interface experiments with real-time synthesis systems. It has since been under development at the University of York.

MIDAS allows users to manipulate a 'tool-kit' of audio-visual algorithms for constructing interactive systems. It also provides a means of distributing these algorithms over a network of connected computers of different types, so as to maximise the available processing power. A full description of the system is given Appendix F, but a summary is now provided so that those unfamiliar with the system will understand how it has been developed and used as part of the research described in this thesis.

MIDAS is based around the concept of the *Unit Generator Process (UGP)*. A UGP is a piece of code which handles an individual task, such as drawing a rectangle, establishing the current position of the mouse or generating a sound output. The concept will be familiar to computer musicians in the form of audio units, such as oscillators and filters (see section 1.7.1) in programs such as *Csound*, but MIDAS UGPs are designed to run in real-time on distributed processors. UGPs can be connected together into *networks* which together perform a more complex audio-visual processing task. MIDAS is therefore capable of being configured into performing any possible synthesis method.

Figure 6.11 is a graphical representation of a MIDAS network running on several different computing platforms.

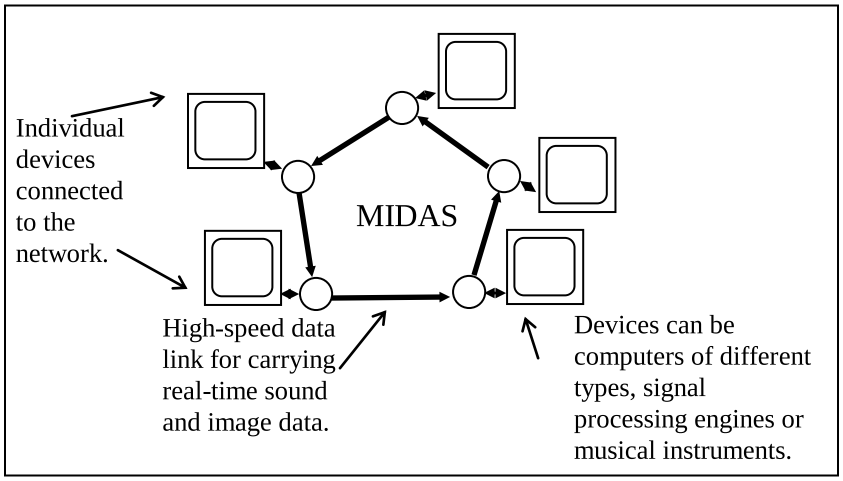
Individual devices connected to the network.
MIDAS
High-speed data link for carrying real-time sound and image data.
Devices can be computers of different types, signal processing engines or musical instruments.

Figure 6.11 – The MIDAS system

The entire network of UGPs is run from top to bottom within a single sample period. This ensures that data flows from the 'upper' UGPs to the lower ones and can be updated at a maximum rate equivalent to the audio sampling period. Each UGP is buffered from the one above and the one below by a 'data holder' (a variable). Therefore MIDAS does not need the concept of a separate 'control rate' and 'sample rate' since any lower rate data is just read repeatedly by a UGP which works at the audio sample rate.

The MIDAS web-site (*http://www-users.york.ac.uk/~adh2/midas/midas.html*) provides full details of the UGPs that are available. It divides them into a series of categories for easy reference and these are shown here:

Category	**Example UGPs**
Audio	('oscillator', 'stereo output')
Data	('integer variable', 'array')
Graphical & User Interface	('rectangle', 'slider')
Maths & Logical	('adder', 'inverter')
MIDI Handling	('extract note', 'filter channel')
Musical Structure	('sequencer', 'draw score')
Signal Manipulation & Flow Control	('rate-of-change', 'conditional subroutine')

6.5.2 Work done on MIDAS

The core of the original MIDAS system was written by Ross Kirk in 1992 and was tested on a network of transputer systems to prove the concept. My roles in the development of the system since 1993 were as follows:

- Port the C program code onto the Silicon Graphics (Indigo/Indy) machines.
- Write the audio and synthesis routines for real-time operation.
- Create the suite of UGPs which are now available (some work done in conjunction with student projects).
- Write the user C programming library which is used to control the system.
- Code the interface layer that takes the user's instructions and drives the kernel.
- Build up and maintain the MIDAS web-site which was begun by Owen Upton (MSc project 1994).
- Write specific UGPs for this DPhil work (see following sections).

The above tasks represent a substantial amount of work approximating one year's full-time programming and development. The benefits of producing the system are manifold. Not only do we now have a cross-platform architecture for audio-visual interaction, but it is easily open to development. It has allowed practical exercises in multimedia, MIDI systems and audio synthesis & control to be run on the Masters course at York. If anyone creates and publishes a UGP it can be employed by any other user in any network. The user interface tests described in this thesis are therefore not locked into a single configuration on a fixed computer platform, but rather can be used for future experiments by reconfiguring the unit generators.

For this study it was possible to use the some of the UGPs developed by others and to write new ones to help create the entire test environment. The following sections describe how MIDAS was used in different ways to implement the user interface study.

6.5.3 Network for Sound Generation

The first role of the MIDAS system in this study is to generate sound that is continuously controlled by the four parameters *pitch, volume, timbre and panning*. Figure 6.12 shows the network of UGPs which carries out this task.

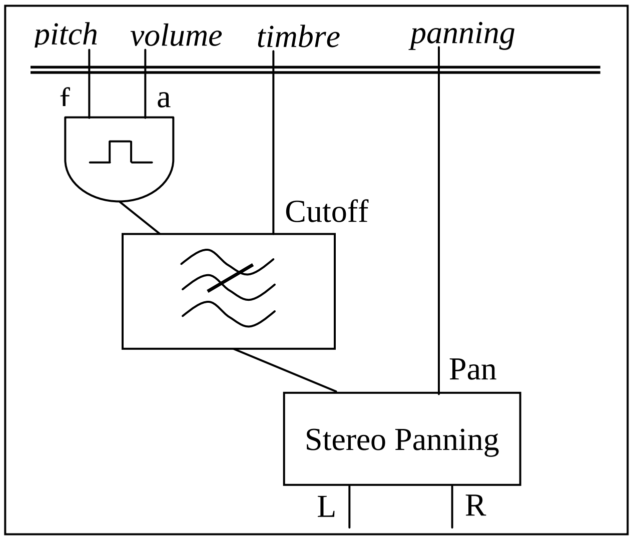

Figure 6.12: The MIDAS sound generation network

The sound source is kept extremely simple. It consists of a square-wave oscillator which has the two controls Frequency (*pitch* input) and Amplitude (*volume* input). The harmonically rich square-wave is passed through a low-pass filter whose cut-off frequency is controlled by the *timbre* input. Finally the filtered signal is panned in stereo according to the *panning* input.

When MIDAS is required to play a sound, data values are fed into the inputs at the appropriate time. When the computer is meant to play a sound for the user to listen to (see section 6.4.1) these data values are read from a set of data files by a C program. This program (listed in Appendix D) loads the data values onto each of the four sound parameter inputs then instructs the MIDAS network to run.

6.5.4 Network for Data Gathering

The same sound synthesis network is used when the human test subject performs on the interface. Data comes in from the interface, is stored for future reference and is also passed on to the appropriate input of the synthesis network. The storage is done by a set of four 'data-logger' UGPs. As the data on each input changes, the new value is stored in a text file along with the current time (measured in audio sample ticks) for later analysis. Since the whole network is run from top to bottom in a single sample frame, the data-loggers add no latency to the system, and do not affect the response of the user interface in any way.

Figure 6.13 shows how the sound synthesis network is controlled from the changes in the user interface.

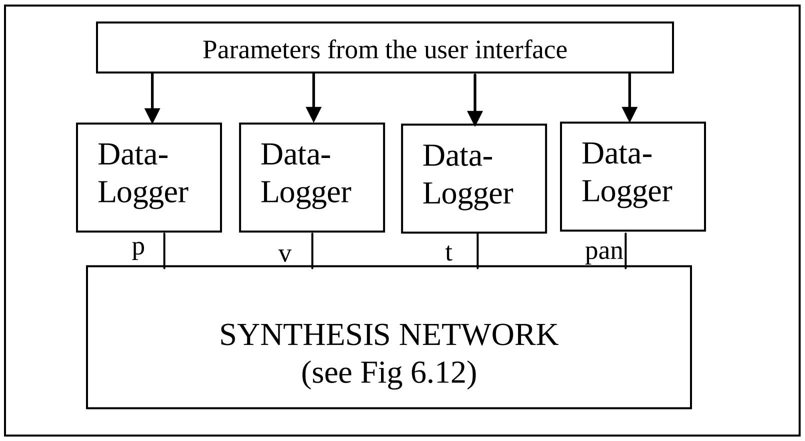

Figure 6.13 - The data gathering network

6.5.5 Networks for the User Interfaces

The interfaces themselves are made up from real physical devices which feed data into the computer. The data is processed by a set of UGPs specific to each interface before being fed into the data gathering and sound processing networks (as shown above in Figure 6.13).

6.5.5.1 The Mouse Interface

This consists of the standard Silicon Graphics mouse which is used to control four on-screen sliders (see section 6.3.1). These graphical sliders are in fact MIDAS slider UGPs. They allow the user to adjust the current value (between upper and lower limits) by moving the mouse pointer over the slider area when the button is held down. This means that the slider bar can be 'clicked' into position, or 'dragged' up and down. The current value of the slider is sent out of the UGP as a data value into the data-logger UGPs and on to the sound synthesis engine (see Figure 6.14).

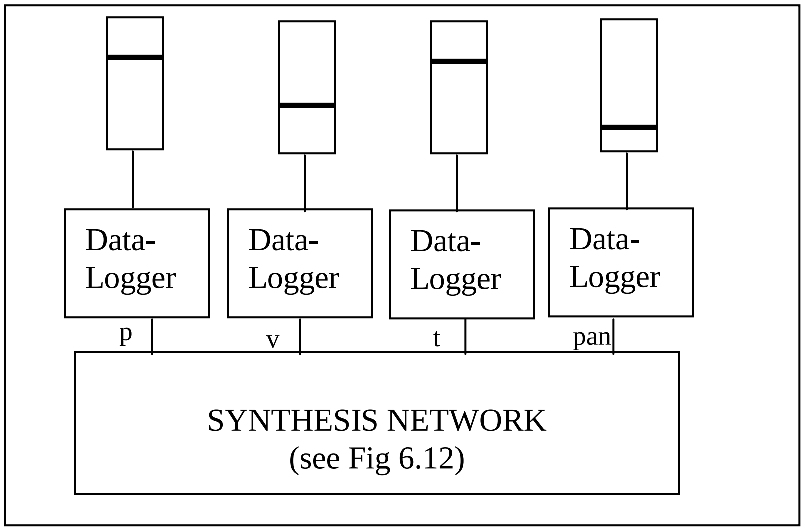

Figure 6.14: The 'mouse' interface network

6.5.5.2 The Sliders Interface

This is an extension of the above network which allows the on-screen sliders to be controlled from a bank of physical sliders on an external MIDI device (see section 6.3.2). This is made possible because the 'slider' UGP can be controlled not only by the mouse pointer, but by a data value injected at one of its inputs. Figure 6.15 shows the UGP network which creates this interface.

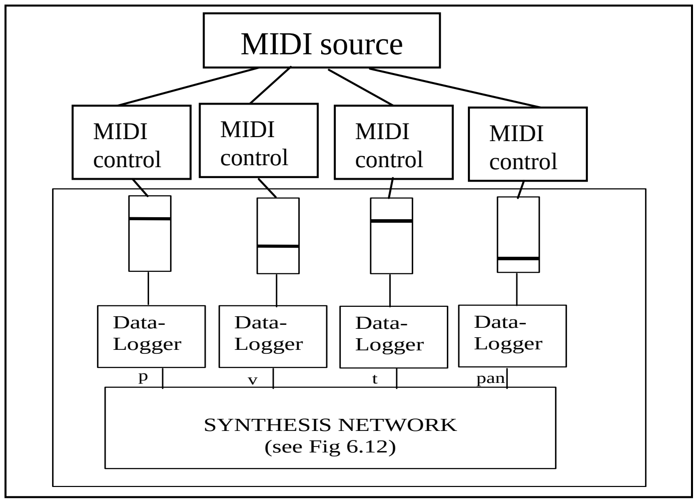

Figure 6.15: The 'sliders' interface network

The Roland Sound Canvas SC-155 has a built-in bank of physical sliders which can be configured to send out MIDI 'volume control' messages. Each slider sends out a message on a different MIDI channel so they are distinguishable when sent down a single MIDI cable. These volume messages are picked up by specially written 'MIDI control extract' UGPs and are fed into the slider UGPs. Thus any movement on a physical slider causes a corresponding slider on the screen to move to the same position. In addition the result of moving that slider is stored in a data file and causes a change of sound from the synthesis engine.

6.5.5.3 The Multiparametric Interface

As described in section 6.3.3 this interface consists of two physical devices - the Silicon Graphics mouse and two sliders of the SC-155 module. The module communicates with MIDAS via a MIDI cable as with the 'sliders' interface, outlined above. The mouse is read by the MIDAS 'window_manager' which makes available the current mouse position and button state to any UGP that requires it.

Where this interface really differs from the other two is in the *processing* of the inputs before they reach the sound engine (see Figure 6.16).

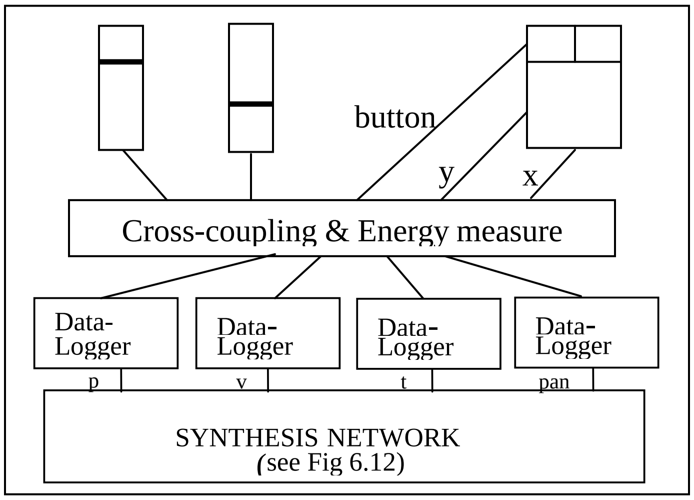

Figure 6.16 - The 'multiparametric' interface layout

There are no 'slider' UGPs and so there is no visual representation on the screen. All the user sees is a blank window containing the mouse pointer. A network of UGPs is used to cross-couple the user's inputs into the four sonic parameters required by the sound engine. This is the 'mapping' outlined in section 6.3.3. The user inputs are gathered from the mouse (x position, y position and left-button-state) and from the MIDI port (volume controllers on channels 1 and 2). Figure 6.17 shows the set of interconnections that implement the mapping function.   Two special UGPs were developed for this purpose.

The first is the 'speed' UGP which outputs the *rate of change* of the input variable. This is central to getting a measure of the user's energy.   In the network below it is mainly used to turn the speed of mouse movement back and forth in the x dimension into a volume control.

The second is the 'range' UGP which scales any input range into any desired output range. This is required, for example, when implementing the 'volume' input from a combination of slider position and speed of movement in the x direction.

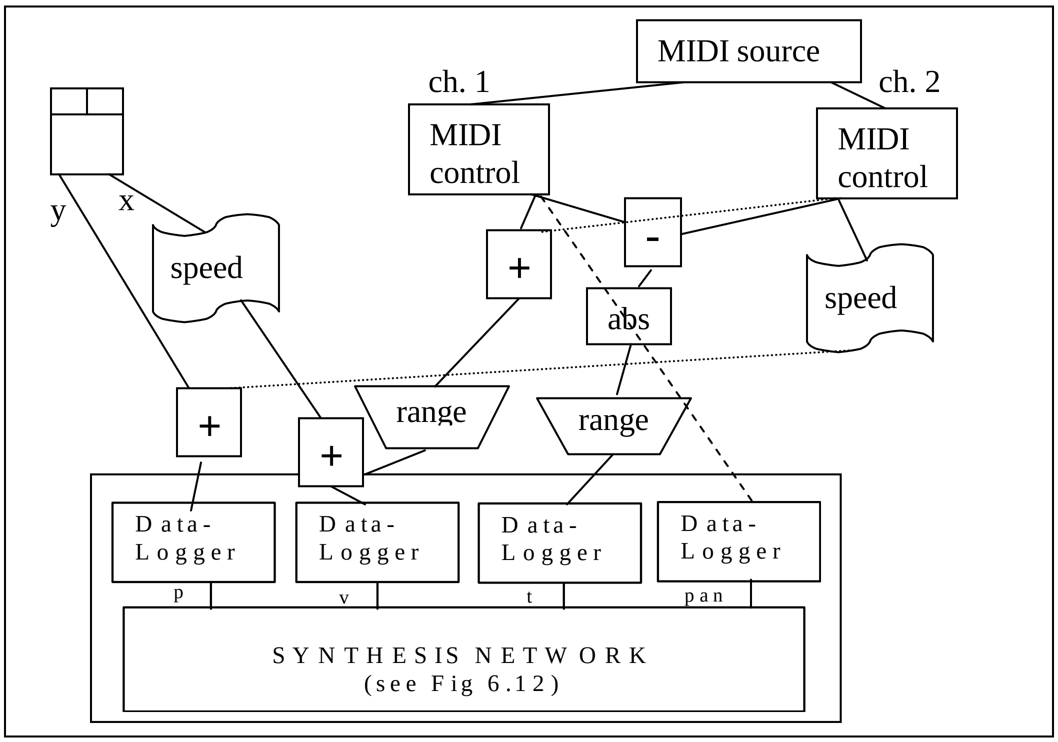

Figure 6.17 - The 'multiparametric' mapping network

6.5.6 Network for Analysis

MIDAS was also used to aid the processing of the test data. An identical synthesis network was used to play the original tests and then the files recorded from the user. In this way a human marker was able to compare each test with the 'ideal' sound played by the computer. Chapter 7 describes in detail how the tests were analysed.

6.6 Summary

This chapter has described how the MIDAS system was used to create three user interfaces for comparison in a series of user tests. These interfaces are based on the mouse, a set of physical sliders and a new multiparametric interface. Some thoughts about the general attributes of multiparametric interface have been given; particularly the interaction of parameters and the use of the human operator's energy.

The tests themselves took the form of a series of 'Listen and Copy' sound trajectories which were played by the computer and then recreated by the user on each of the interfaces. The data values generated by the user were stored in files for later analysis. Further details on the test environment, the human subjects, the musical nature of the tests and the analysis are given in Chapter 7.