WORKING NOTE AC102

Alan Creak
1996 September 23

MIDIGRID NOTES

I looked at Andy Hunt's MidiGrid document[1] with a view to seeing whether there was anything in it which I could address with my system specification approach[2]. These notes record my observations and conclusions. They are illuminated in places by my recollections of a meeting with Andy[3], at which he very effectively demonstrated MidiGrid. Page references are to Andy's paper[1].

SUMMARY OF MIDIGRID.

MidiGrid is centred on the grid, which is a rectangular array of cells displayed on a computer screen. Each cell can be set up so that when it is selected a predefined sound is generated. A cell's definition may also include a specified successor cell. The cell contents are not predefined, so to use the system it is necessary both to store sounds and other initial values in a set of cells and to play the instrument by selecting cells.

Several sorts of musical material can be stored in a cell. The simplest is a single note of a predefined timbre, which can be anything provided by the connected MIDI synthesiser. Any combination of notes may also be stored in a cell, so chords can be constructed; the notes need not be of the same timbre. Finally, any MIDI sequence can be stored in a cell (page 6), so that one cell can contain a musical passage of considerable duration. The input is provided from a MIDI keyboard (or, presumably, any other MIDI instrument), and the contents of cells can be combined with further input within MidiGrid to compose more complex musical sequences (page 7).

Cells can be selected in any of three ways :

- by direct selection, using a mouse or some equivalent selection device;
- by activating some external MIDI device (for example, a MIDI instrument, or a switch set up to generate MIDI signals) which is "mapped" to the cell (page 5);
- by "step-time" operation (page 6); a switch which acts as a trigger to select the current cell's successor is operated. (The mouse button is suggested as a possible trigger, but there is no discussion of the interaction of this function of the button with its normal function of selecting grid cells.)

There appears to be no provision in the screen selection method for selecting two or more cells simultaneously, though there is no constraint on the rate of selection. Presumably multiple simultaneous selections are possible if an external keyboard mapped to the grid is used as the selecting device.

SETTING UP MIDIGRID.

Not much is said about setting up the system. There seems to be little to do except select a cell and place the desired material in it, and perhaps establish mappings from external MIDI signals to grid cells or other functions. A picture of the grid (page 6) shows various symbols round the edge labelled "Icons for program control"; presumably these identify various functions used in setting up the system, but they are not described.

There is no restriction on what can be associated with what, or on the ordering of associations. This is deliberate, as it opens the way to constructing instruments in which sound combinations which are physically impossible to play on conventional instruments can be made accessible (page 11). Notes can therefore be stored in the grid at any point, irrespective of the order of pitch, and the mapping from an external MIDI instrument to the cells and the notes, or other material, which they contain is similarly arbitrary.

PLAYING MIDIGRID.

Playing the instrument, once set up, is a matter of selecting grid cells. In the simplest application, individual cells are selected one by one with mouse clicks, with obvious consequences.

Other styles of input are possible, and are mentioned once or twice in the paper. The effect of using gestures, by holding down the mouse button and sweeping the pointer across the grid in arcs or circles, is an example.

So far as I can see, there's *only* cell selection. I didn't notice any mention of other controls, such as volume, sustain, etc., which might be used during performance.

THE INTERFACE.

There seems to be curiously little to say about the interface, perhaps because all you can do is select a cell. Because of the mapping function, it can be any MIDI instrument interface, or practically anything else which can produce a MIDI signal; I'll ignore this complication and concentrate on the screen grid.

There is some feedback from the interface. Cells are distinguished by single dots for single notes, multiple dots for chords, and lines for sequences. Chords are also named, and the sequence given is supplied with a title; it isn't clear whether these extra bits are supplied automatically, or whether any other help in composing them is provided. The cells do not seem to be very easy to distinguish, but then neither are harp strings nor piano keys. From Andy's facility at performance it's clear that an expert can overcome any such problems, but clearer distinctions between the cells might be helpful for learners.

SUMMARY.

MidiGrid is a collection of facilities for acquiring, storing, and replaying musical objects. There are few constraints on what can be done with the facilities, and no intrinsic structure to encourage one sort of use over another. There are a few suggestions in the paper of things that might be done with MidiGrid, but no suggestion that anything in particular *ought* to be done. There isn't much evidence of detailed design; the paper reads rather as though it were the elaboration of a single idea (selection of a musical note using a mouse – here I acknowledge that I'm cheating by including a comment from our meeting with Andy), with a few obvious generalisations (chords, sequences, mapping) which looked as though they might be useful.

In short, then, MidiGrid is a kit of parts. It is not so much a musical instrument as a workshop in which you can build a musical instrument. (The phrase "configurable instrument" is used in the paper (page 9), but I prefer the "kit of parts" metaphor. Would you regard an organ with all its pipes detached as an instrument ? I suppose I should formulate a definition of a musical instrument if I want to justify my preference, but I also suppose that it isn't that important !) That isn't a negative criticism; it classes MidiGrid with assembly languages, Meccano sets, and Unix, all of which are extremely useful tools.

It also means that MidiGrid has no intrinsic purpose. Again, that's not a negative comment, and is entirely appropriate for a kit of parts, but from a user-interface point of view it knocks away the props, for the first question is "What are you trying to do ?". User interfaces are not just designed; they are designed to accomplish some task, and if the task isn't specified there's not much for the interface designer to do.

COMMENTS ON INTERFACE DESIGN.

Well, there isn't *quite* no specification. At the very least, the interface is there to provide a means of selecting cells from the grid, and one can comment a little on that level of the system. Such comment might centre on the identification of the required cell. One would also like to comment on the positioning of the cells, but without some further specification of how the interface will be used, I don't think that there's anything useful to say on that point.

There are three parts to identifying a cell : distinguishing cells from other things, finding the cell you want, and selecting it.

Ideally, nothing but the active cells and perhaps a few necessary switches for changing modes – such as an "off" switch – should appear when the grid is in use. (In line with my earlier comment, I'm

assuming that there are no additional controls which affect the performance.) Anything that isn't a cell should look very different from a cell, or should be in some separate area of the screen – so a menu bar, or the thick boundary line shown on page 6, would probably not cause confusion. Cells for which no function is defined should be invisible.

To identify the cell required, one can use the position of the cell and its appearance. There is a remark (page 5) that each cell "can contain graphical 'icons' which represent musical material", which sounds like a good idea in principle, but it isn't clear where the icons come from. The picture on page 6 is not enlightening. If the intention is to use little pictures, as in desktop interfaces, then there might be difficulties; it takes some little time to distinguish one icon from another, and in a system where quite rapid selection is likely to be important the recognition time might be a hindrance. Colour codes, perhaps associated with very simple patterns, could be better, as they are much easier to recognise, while for colourblind people clearly distinguishable patterns would be preferable. It is also noteworthy that, apart from the ease of recognition, simple colours and patterns are much easier to design than are evocative pictures.

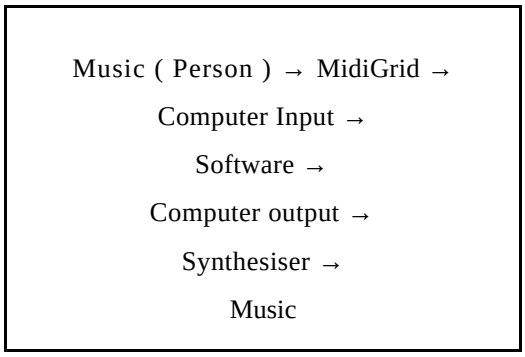
Cells are selected using a mouse. The only good point that I can see about that is that it comes with the computer. It's true that, with practice, one can become very proficient with mouse selection, but it's a very artificial way to do things and wastes a lot of the bandwidth available from a hand. A hand can convey information through (at least) two channels, position and pose; a mouse is completely insensitive to pose, which is the wider channel. (Consider sign languages.) Notice that, among conventional instruments, keyboards and strings exploit both the position and pose of hands in defining the pitch of a note, while brass (excepting trombones) and woodwind (excepting swanee whistles) predominantly use pose alone. Only percussion relies heavily on position alone. I'd have preferred some form of touch-sensitive input – a touch screen, or a sensitive pad, provided that it could detect several simultaneous contacts.

There are some obvious general comments about mapping. While arbitrary allocation of anything to anything is all right for the grid, which has no customary semantic associations, arbitrary mappings of the elements of – say – a keyboard, for which people have well-defined expectations, might be more confusing than helpful. For example, one could map the middle C of a keyboard otherwise mapped normally to the signal which advances the selection to its successor. If nothing else, that suggests that common sense is likely to avoid the more bizarre possibilities, but, once again, much depends on just how the interface will be configured.

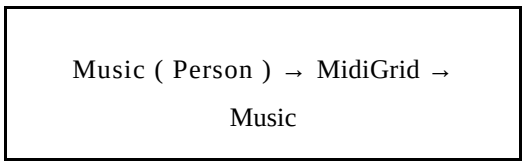
There are some hints of further structure in some of the suggestions. For example, the possibility of using characteristic gestures, such as "moving the mouse in a circular motion" (page 5, page 9) and "sweeps of the hand" (page 9), is mentioned; partitioning the screen into separate areas (page 9) is also suggested for a special type of application. If these are seen as generally significant, some interesting questions open up. For example, how would one design a grid to take advantage of these possibilities ? – and how would one address the problem of mapping gestures to – say – switches used by disabled people ? Yet again, though, I'm not sure that there's much to be said without some richer definition of the purpose of the operation.

WHAT ABOUT MY METHOD ?

Well, not much, really. I can start off well enough by identifying the overall process to be studied :



and, as the computer music end of the system is already done, I can abbreviate that to :



(though I reserve the right to expand it again if circumstances warrant further probing). I can also identify the input vocabulary of MidiGrid satisfactorily at the purely physical level of selecting cells.

But I can't get much further, except to reach conclusions of the sort I mentioned in the previous paragraph. They are, more or less, the result of assuming that the player plans the music note by note, and selects it note by note. (More precisely, for "note" read "cell".) I can do a bit better if I expand the nature of the plan a little, including attributes such as volume, attack, etc. which might in practice be determined during the performance. Having done so, though, I always find them missing from the MidiGrid vocabulary, so the answer is always "can't be done". (– or "design different variants into the MidiGrid configuration" : see below.)

I think that to get any further I'd have to assume some specific MidiGrid configuration, which is in effect a means of extending the input vocabulary by adding more semantic content to certain actions. I'd also want to know how people plan their actions during a musical performance at a level higher than single notes. Given those, I'd then go on to think about how the musician would encode his plan in terms of the available vocabulary, and how easy it would be to perform the corresponding action sequence.

One further possibility is raised by the parenthesised comment at the end of the paragraph before last : perhaps it would be possible to use the technique in defining configurations which are suitable for certain sorts of performance. I'd have to think a bit more before commenting further on that.

REFERENCES.

1 : A. Hunt, R. Kirk : "MidiGrid – a computer-based musical instrument", *J. Inst. Musical Instrument Technology* **1**, 3 (June, 1994).

2 : G.A. Creak : *Reaching Beyond Words In Rehabilitation Computer Systems*, unpublished Working Note AC96 (May, 1996).

3 : A. Hunt, meeting with me and Alistair Edwards, 26 February 1996.