Composing and performing with switches, using specialised or adapted music software

Presented At The International Conference on Assistive Technology, Derby April 2002

DR. TIM ANDERSON

R&D Manager, Drake Music Project *www.DrakeMusicProject.com*

Creative people with disabilities who use switch control have particular problems using music software to compose and arrange music. Two complementary approaches to these difficulties are examined:

Specialist music software designed for switch use, such as the Drake Music Project's 'E–Scape', can provide high-level guided operation which can greatly aid switch users. Presentation of choices, as well as guidance and feedback through processes can enable beginners to computers and/or music to get experience and motivation at an early stage. Some of these features are illustrated via some simple composing tasks.

Another approach is to use standard music software alongside 'overlay' emulation software such as 'Hands Off', which presents grids of cells which can be selected by a switch user to operate keyboard and mouse functions on the computer. However, a variety of problems are still encountered when operating music software, which can disbar very many switch users. At the moment, a few switch users may be able to progress to operating more advanced industry standard music software via overlays, while the vast majority have to remain with a specialist music system built around the expectation of switch use.

To investigate the possibilities of providing higher-level, more guided operation of standard software for composing music, overlay grids were developed to undertake some simple musical tasks. However, a variety of difficulties were still encountered, which has led to a number of ideas for enhanced functionality for overlay systems. These would enable user interfaces which behave more like specialist systems to be constructed, and enable switch users to control any standard music software more effectively. Further work is planned in conjunction with Sensory Software to expand the possibilities for high-level overlay control of music software.

There is, however, still a continuing need for specialist software to enable a switch user to effectively *perform* music live.

INTRODUCTION

As anyone who does it can tell you, making music - whether it be composing or performing - can be incredibly fulfilling, satisfying and life enhancing. This is especially so for people with disabilities, who often have a narrower range of opportunities for

creative expression. In the Drake Music Project, one of the key aims is to enable people's musical creativity; this includes the provision of suitable systems as well as tuition and music making opportunities. The focus of this paper is on providing systems for music making for people whose physical disabilities are more profound - for example someone who can't use a trackball or other mouse control, or use a computer or music keyboard. This usually implies switch operation, and this paper explores the issues and possible options for switch users in operating music software effectively.

USING MUSIC SOFTWARE VIA SWITCHES

People with such disabilities will have little of no experience of 'playing about' with musical instruments, and will have had no opportunity (or ability) to have done even simple rhythmic activities such as clapping along to a tune, which a primary school child will often experience. Singing is another activity which can enable many children to acquire ability and interest in music, and again many disabled people will not have had this experience. However, they may have a deep love of music at a listening level, plus - for some - musical ideas which have no outlet.

As in many areas, over the past 20 years or so computers have hugely enabled people in making music, and there is now a vast array of very capable software, enabling people to create and arrange music in any style, print out high quality scores, and play back their music with excellent fidelity and range of instrumentation.

The main way a switch user can operate such software is to use secondary 'overlay' software, such as 'Ke:nx' (Don Johnston) for Mac, or 'Hands Off' (Sensory Software) for PC. These consist of grids of cells which appear on top of ('overlay') an application they are controlling. Rows and columns can be scanned by switches, and when a cell is selected it can communicate with the underlying application and effectively perform any mouse or keyboard action, although only low-level mouse operations tend to be provided such as moving, holding down a button etc. A cell can contain sequences of commands ('macros') which can control more complex operations, such as opening a dialog, selecting a certain value via a radio button, toggling a second parameter via a push button, then closing the dialog. The user can also jump to other grids with different sets of commands suitable for different situations.

However, powerful and sophisticated music software comes at the price of complexity of operation, and is heavily geared around extensive fine control of the mouse cursor, as well as a large number of repetitive operations, choices and changes of mode. Examples are the large number of palettes of music symbols; the many editing windows, with large variation in display properties and scale; the large number of parameters for each note; the variation in organisational hierarchy and grouping; the time scale, looping and audition points; the many status and value dialogs or displays, and overall the large number of mouse dragging actions demanded to set values and manipulate graphic objects. In addition, it is usually also geared around the expectation of using MIDI musical instruments to input notes or record live playing, rather than entering individual notes by specifying position, pitch and duration as a switch user needs to do.

All this adds up to making operation of music software arguably far more demanding and intensive than most office software, and gives *switch* users even greater problems. Using switches to operate an overlay menu system which then operates mouse movements or key presses to control music software is very labour intensive and long-winded, as will be demonstrated. This complexity and laboriousness, combined with people's lack of experience in music making, can all conspire to discourage people at an early stage.

COMPARING OPERATION VIA OVERLAY VS. DEDICATED SWITCH SOFTWARE

We will now explore the possibilities of using overlays to control music software in a less laborious way. We will first illustrate some of the guidance, feedback and ease of use features provided within 'EScape' - specialist music software initially developed by the Drake Music Project to provide composing facilities for switch users (although its range of usage has considerably broadened since the outset). Such features have

been found to greatly reduce the workload on switch users, and enable them to focus on creative ideas rather than get bogged down in low-level and often long-winded computer operation.
To do this, we first describe how some simple musical tasks can be carried out in E-Scape: (A) entering two groups of 4 notes, with two different lengths; (B) auditioning the result; (C) selecting four of the notes, and (D) transposing (changing their pitches). We then compare how the same simple operations can be carried out using an overlay with standard music software, by developing a series of dedicated grids for the ' Hands Off' overlay, with the aim of giving a user the same kind of high-level control provided by E-Scape. These grids are designed to present the user with context sensitive choices - ie only those which are meaningful and sensible at the time, with easy to understand text, and far less need to understand the detailed operational control of the music software.
This goes beyond the more basic usage of overlays, where actions are triggered which press keys and move the mouse, and where the user not only needs to remember which shortcuts do what, but also in which context.

EXAMPLE MUSIC ACTIVITY USING SPECIALIST 'E-SCAPE' SOFTWARE

Because it is designed from the outset for switch operation, E-Scape can provide the common musical operations involved in entering, editing and arranging music in a guided manner. Users can choose a ' level' to operate at - lower levels have more guidance, with musical activities built from simpler actions which appear to the user as a series of questions.
This enables people - after little or no training - to focus on musical choices rather than issues of how to operate the system, and even a beginner (q.v.) can quickly be left alone to work unaided. A user can also decide to work at a higher ' user level' , where operations are more open-ended, more choices are provided and more advanced language used.

(A) entering some notes with two different lengths; (B) auditioning the result
1. Press switch to open the main menu (NB. shown here *after* notes have been added):

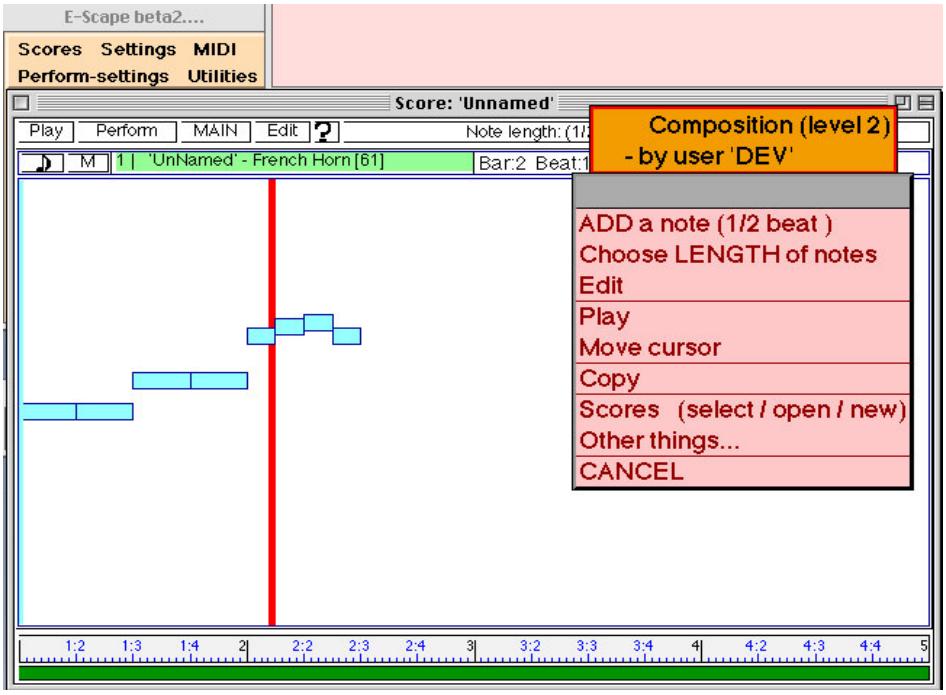

Fig. 1 E-Scape window with main menu

2. Then select [Add a note] (with default length of 1 beat). This brings up a menu with notes:

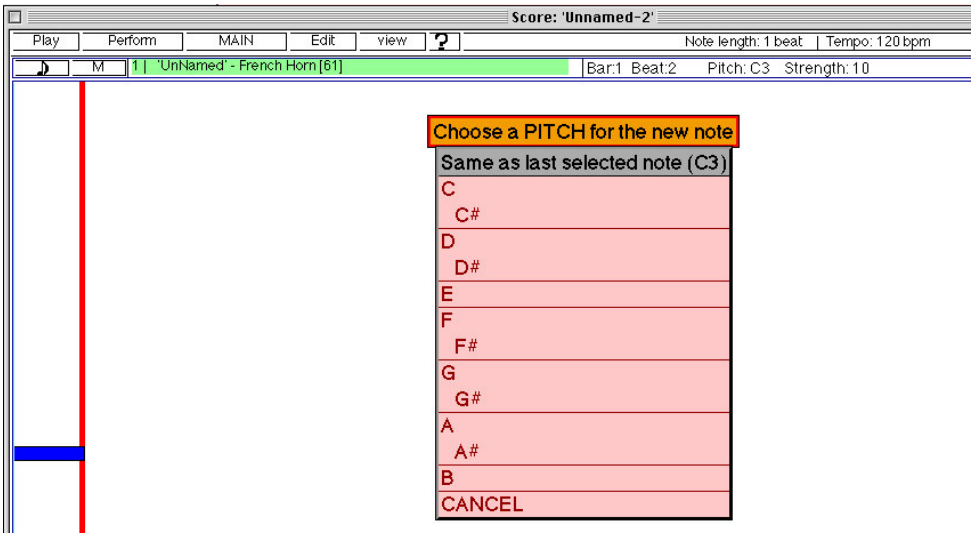

Fig. 2 E-Scape menu to select the pitch of a note to be added

Each note plays itself from the menu. In addition, if the user dwells on a note in the menu, it will automatically audition - ie play along with any surrounding notes - to illustrate what the track would sound like *if* this note were added.

3. Select a note to add, then repeat to add 3 further notes of 1 beat.

4. Select [Choose length of note] from the main menu, to open the duration menu:

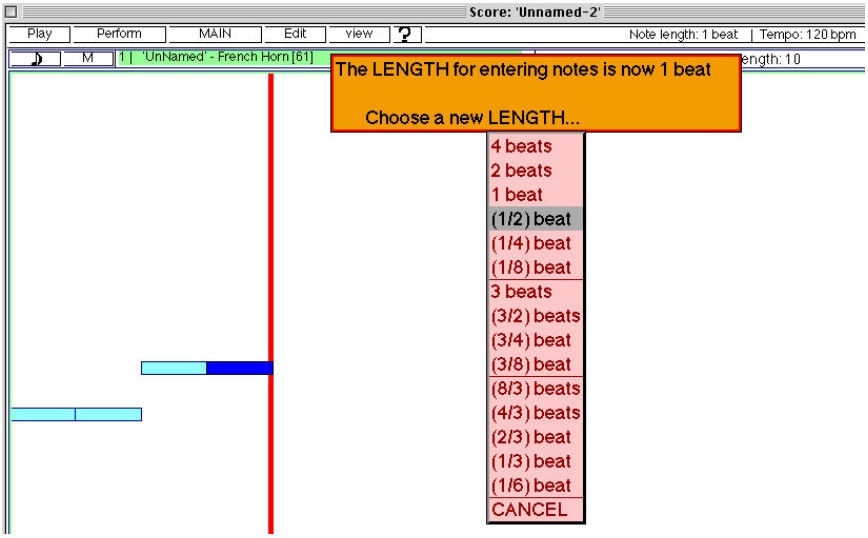

Fig. 3 E-Scape menu to select the length of note to be added

5. Select [1/2 beat]. Each item again plays itself, as a sample 2 bar rhythm, with notes of the selected length to illustrate audibly how long it is.
6. Repeat 1,2,3 above, to add four notes of this length.

(C) selecting four of the notes

1. First, select [Edit] from the main menu, to open the Edit menu:

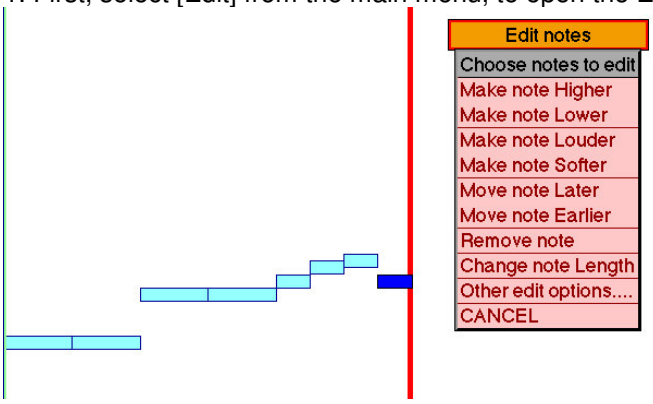

Fig. 4 E-Scape ' Edit' menu

2. Then select [Choose notes to edit], which then guides the user through the process (fig 6, a - e) of selecting a block of notes:

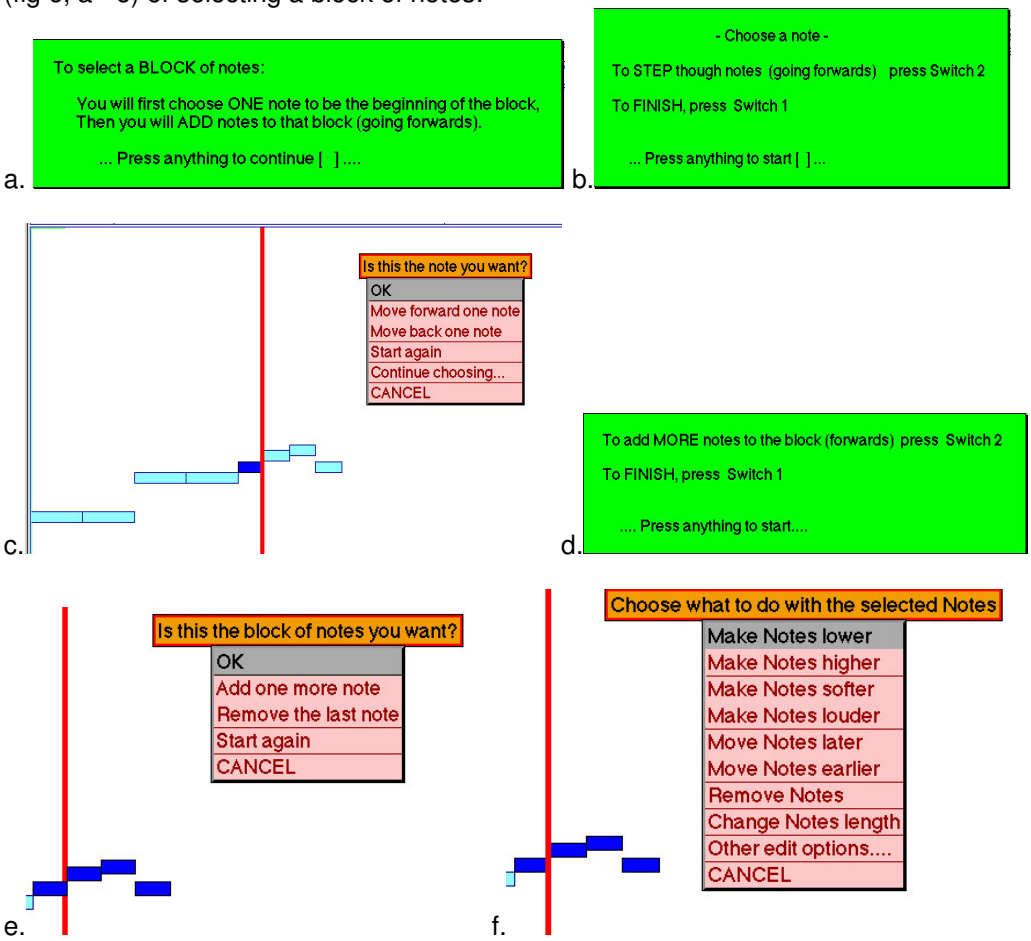

Fig. 5 Six stages of the E-Scape guided procedure to select notes for editing

After b & d, each switch-press highlights the next note to show it is selected, and also plays it.
Finally a variation of the Edit menu (f) appears again which prompts the user for what they want to do with the notes.

(D) transposing notes

1. From this Edit menu (f), select [Make Notes higher], and another guided procedure is launched. First some instructions are shown:

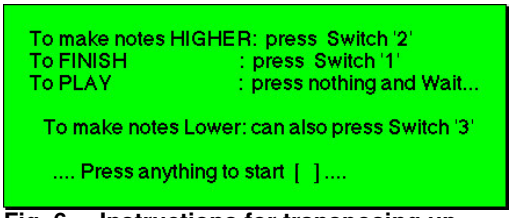

Fig. 6 Instructions for transposing up

2. Each subsequent switch press then transposes the notes up one, but if the user waits (does nothing) for a moment, the changed notes automatically audition - first playing once alone, then playing in context with surrounding notes in the track (and also, then with any other tracks). This happens in E-Scape in any editing operation, and has been found to be very helpful to disabled users - greatly reducing the amount of switching, and distraction from the musical task.

Fig. 7 Leon Hippolyte (two switches) and Baz Wright (single switch) composing using 'EScape' software

EXAMPLE MUSIC ACTIVITY USING OVERLAY WITH STANDARD SOFTWARE

We now show how the same musical operations can be carried out using the standard 'Cakewalk Pro Audio v8' music software, controlled by the 'Hands Off' overlay with dedicated grids. Cakewalk has a relatively good spread of keyboard shortcuts and controls, which gives Hands Off a reasonable task, whereas initial attempts to operate 'Cubasis' and 'Rebirth' music software had to be abandoned due to the larger number of options only available via mouse control.

First, some nomenclature is needed to clarify description. Each Hands Off 'Grid', contain 'cells' which are named here within [square brackets]. When a [cell] is 'selected' (using scanning menus etc) it effectively "type" a key such as <Return> which may have the effect (as a keyboard shortcut) of "pressing" a graphic 'button' (named with inverted commas) within a dialog on screen. A [cell] can also initiate a "jump" to another Grid.

(A) entering some notes with two different lengths

When Cakewalk is launched, the top-level Hands Off grid for Cakewalk appears - here at bottom left. (NB. Cakewalk shown here with the edit window open, and *after* notes have been added):

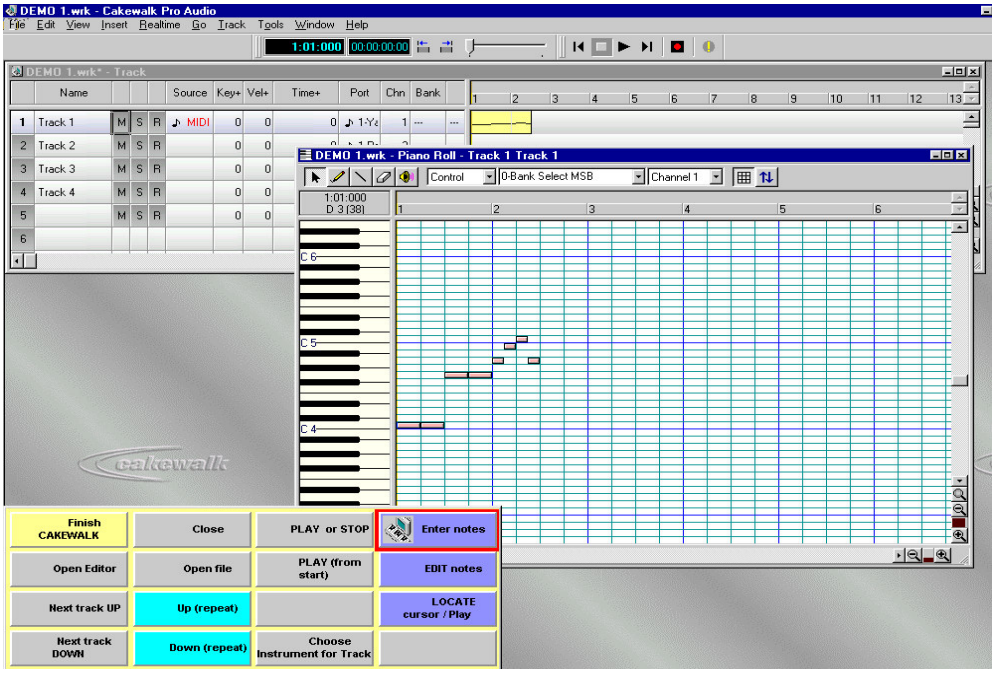

Fig. 8 Cakewalk edit window, with Hands Off top-level grid

1. Press switch to start scanning the grid…
2. Select [Open editor] to open an edit (' Piano Roll') windowas shown in fig. 8.
3. Select [Enter notes] which opens a ' Step Record' window, and jumps to another Grid:

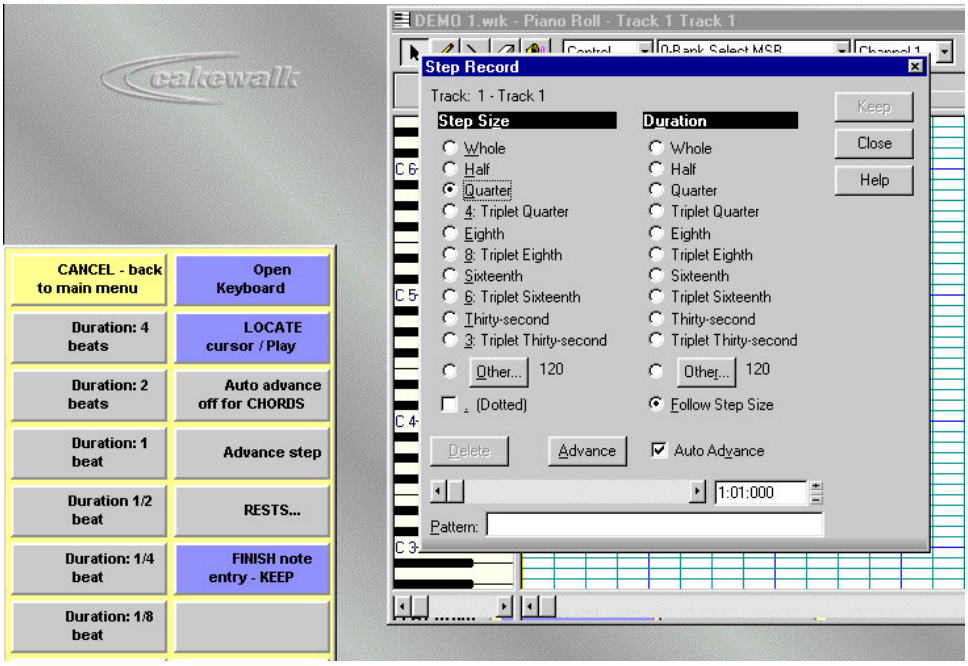

Fig. 9 Cakewalk ' Step Record' window, with associated Hands Off grid

Note how the Grid shape has been designed to avoid obscuring the window, which Cakewalk will not allow to be moved. A number of complex problems were encountered in this window - see Appendix 1.1 for details.

4. Select [Open keyboard], leaving the duration set at default crotchet (= ' 1 beat' at 4/4). NB. Cakewalk displays the US ' Quarter note' . This opens a graphical ' Keyboard' window, and jumps to another Grid of notes [C], [D] etc:

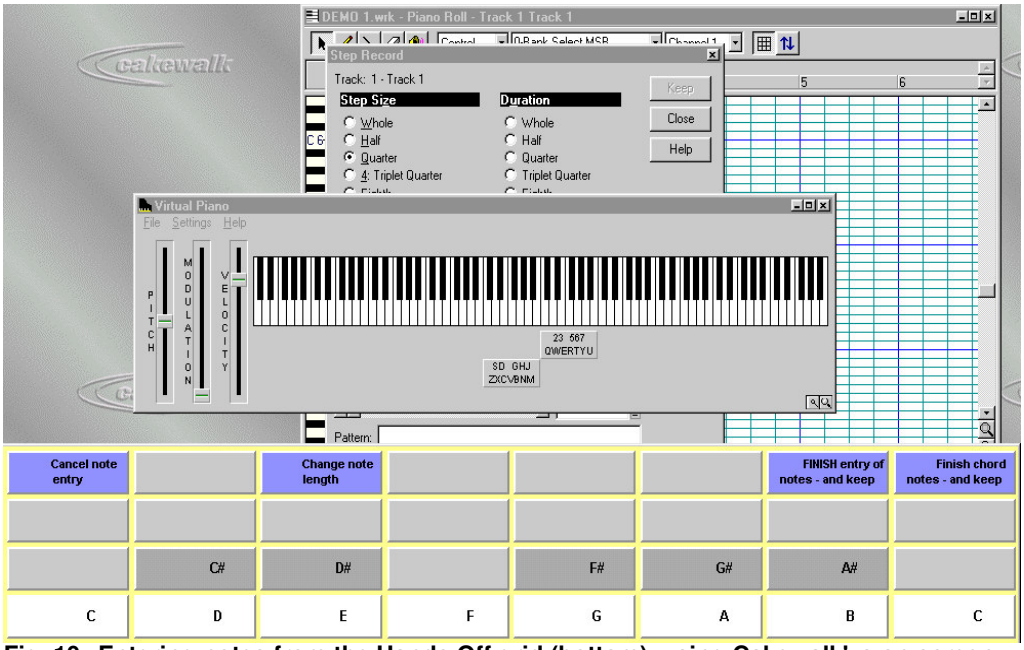

Fig. 10 Entering notes from the Hands Off grid (bottom), using Cakewalk' s on screen piano keyboard

5. Select [C], [C], [G], [G] in turn to enter 4 notes.
6. Then select [Change note length]. This closes the ' Keyboard' window, and returns to the ' Step Record' window also jumping back to the previous Grid (as 3 above).
Another subtle and difficult problem arises in doing this, due to the fact that user actions in the Keyboard window can change the *effect* of subsequent key presses in the Step Record window - and the overlay has no way of knowing this. See Appendix 1.2 for details.
7. In the Step Record grid (as 3 above), select [Duration: 1/2 beat],
8. Then select [Open Keyboard] to reopen the Keyboard window (as 4 above).
9. In the keyboard grid, select four note cells in turn (as 5 above) to add four more notes with this new duration.
10. Then select [FINISH entry of notes - and keep]. This closes both the Keyboard and Step Entry windows to reveal the Editor window with the notes in place, and also jumps back the the main Cakewalk Grid (as shown in fig. 8).

(B) Auditioning the resulting notes
1. From the main grid select [Play from start] or [Play].
NB. If there were a large number of notes already entered, the user would need to select [Locate cursor / Play] which jumps to another Grid where they can move to, then play from, a particular cursor position rather than the start.

(C) Selecting four of the notes
This is not straightforward, as the only way - without using low-level mouse operation - is by specifying a start and end time within which notes will be selected. Due to Cakewalk' s idiosyncrasies, even this has been a challenge to make work (see Appendix 1.3) and it is also still not possible to select individual notes within a chord, for example.

1. From the main grid, select [EDIT notes], then [Select notes].
This opens a ' Select by time' dialog, with the ' From' field active, and jumps to another Grid where times can be entered, or changed via the [Decrease..] or [Increase..] cells:

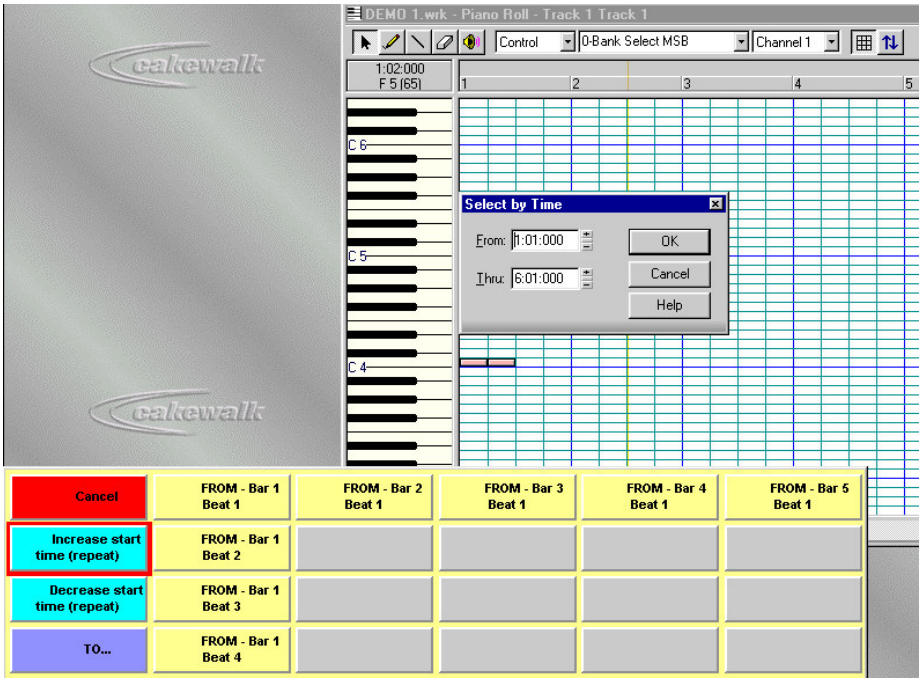

Fig. 11 Using Cakewalk's ' Select by Time' dialog to select notes, with a Hands Off grid to specify a ' from' time

2. Specify a ' from' time by selecting [Increase start time (repeat)], and pressing switch repeatedly.
3. Then select the [TO..] cell. This changes the focus in the dialog to the ' To' ("Thru") field, and also jumps to a second Grid where the user can select an ' end time' for selected notes:

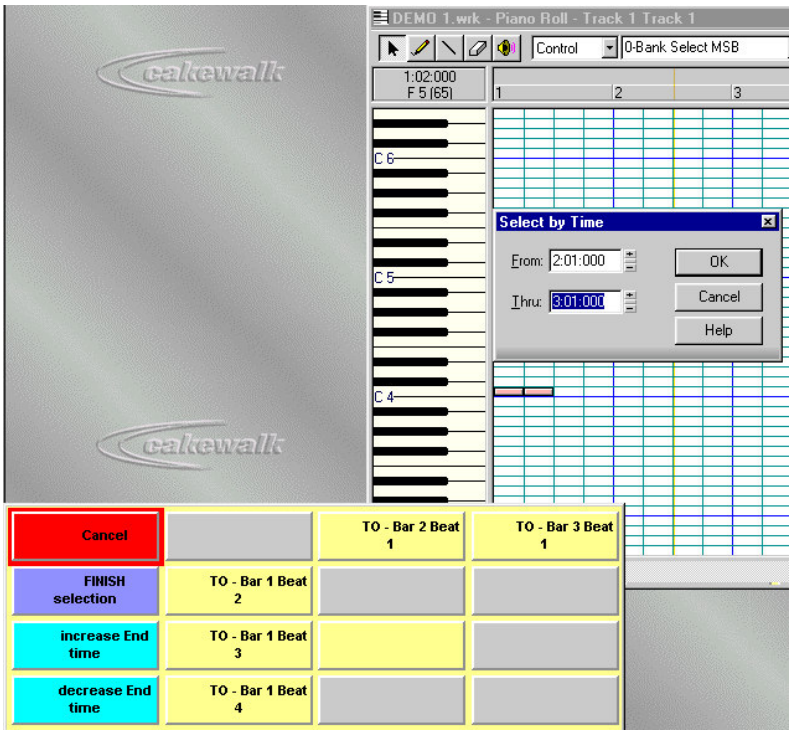

Fig. 12 The second field of the Cakewalk dialog now made active - with a second Grid to select a ' To' time

4. Select a ' Thru time (ie the ' end' time for events to be selected) which the user has to make sure is higher than the ' From' time.
5. Select [FINISH Selection]. This closes the dialog, leaving the desired notes (between the two times) selected; it then jumps back to the ' EDIT' Grid:

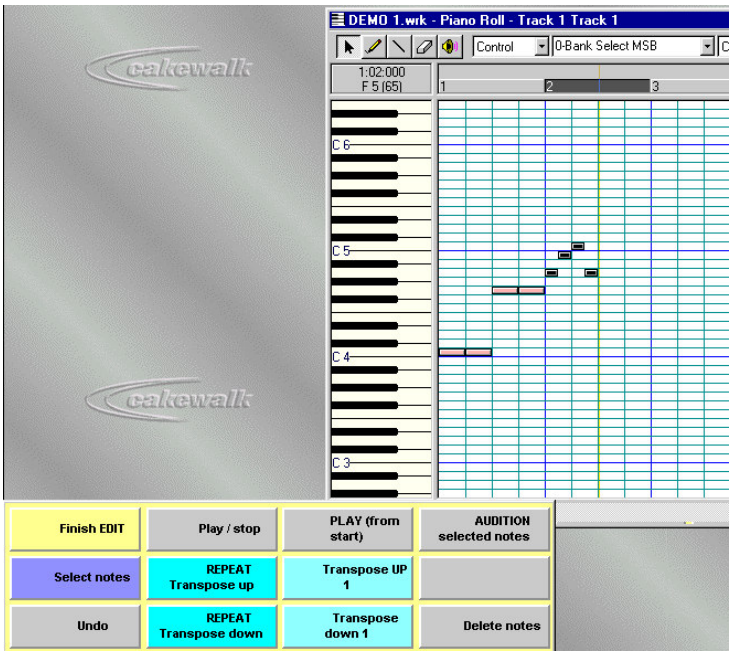

Fig. 13 Cakewalk edit window and associated Hands Off ' edit' grid, with 4 notes now selected

(D) transposing (changing note pitches) and audition

1. Select [Transpose up], or [Repeat transpose] etc.
2. Select [AUDITION selected notes] to play the selected notes, or [Play..] etc to play all the notes. However, this does not really ' audition' in the sense of playing the selected notes along with others nearby.

PROBLEMS USING OVERLAYS FOR HIGH-LEVEL CONTROL OF APPLICATIONS

It is not necessary to follow the necessarily detailed examples given to get an impression of the operational difficulties involved in controlling even simple musical procedures via overlays. Various specific problems were encountered - detailed in Appendix 1 - which are difficult to present in a brief demonstration, but two key general points can be made:
1. One factor is that the application needs to be ' friendly' to overlay control. This is not just a matter of having plenty of keyboard shortcuts for example, although interestingly, even the recent version (v8) of the PC music application used here (which grew out of DOS) *still* has several vital functions for which there is no keyboard control. Another aspect is of applications having various ' annoying habits' as regards co-operating with an overlay, for example, the ' aberrant behaviour' of dialogs when values are entered via the overlay (c.f. typed in manually). This is usually caused by the programmer of the application using ' unofficial' shortcuts to achieve subtle effects (such as not allowing certain inputs to be typed into a dialog), and not using the standard controls to process Windows messages in the conventional way. The effect is to make the life of any overlay system very difficult, and using it can become a game of finding ' work arounds' , or lengthy ' fiddles' to achieve control, again illustrated in Appendix 1.
2. Another more fundamental problem is that overlay systems are inherently ' blind' to the context and state (and indeed presence) of the application they are controlling, and

the music application can not communicate information back to them - eg of where it is in a score, and what state or edit mode it is in. Only if a standard protocol could be agreed for inter-application communication could this issue begin to be addressed. Thus an overlay can't tell where it is in the piece of music, or whether the edit screen is scrolled or zoomed, horizontally or vertically. This means that in many cases, the user will need to understand and operate the overlay to access the application at a *lower* level - eg ' pressing' <tab> to highlight different value fields in a dialog, or comprehending the current location (in bars, beats and ticks), and entering numbers to locate the cursor so as to play particular notes.

PERFORMING MUSIC WITH SWITCHES

Another issue for switch users is that conventional music software - even that designed for disabled use - does not support live performing via switches in a suitably sophisticated manner to be able to be used to perform with other musicians, although the MIDIGrid application has been successfully used by Drake tutors for simple live triggering of music by switches. For this kind of activity, dedicated software such as E–Scape is really essential [Anderson, 99]. Using E-Scape, a musician can prepare aspects of the musical performance in advance, and control many other aspects live, using a number of palettes of material - all via one or more switches.

SUMMARY AND CONCLUSIONS

The examples given hopefully illustrate some of the advantages of a custom system, with feedback, guidance, clear language, and avoidance of mistakes. The example activity given was deliberately chosen to be very simple to achieve using the overlay system, but even so the number of switch presses is at least 50% higher, although this is not by any means the only measure of difficulty in using the system. If the same degree of feedback and auditioning that is present in E-Scape (which enables beginners to choose notes by ear) were demanded of the Cakewalk / overlay system, then the complexity of operation would vastly increase from that illustrated, and at least 3 times as many switch presses would be needed, although with more work on developing grids this may be able to be improved on.

However, the key disadvantage of the specialist system is just that it *is* specialist; most disabled users would prefer - if they could - to use industry standard software, for a number of reasons, both practical and psychological. Another factor is the development power that a commercial company can bring to bear, and the far larger scale of the customer base. This means it is very difficult for a small specialist organisation such as the Drake Music Project to be able to develop systems with the sophistication, power and complexity (at a musical level) of the industry giants.

For those few very advanced switch users who have the tenacity and ability, it is possible to access all areas of software control, *if* low-level mouse use is invoked. For such people, the ability to use the power of standard software is an important pathway to creativity, and worth the far greater effort and laboriousness involved. The importance to people of feeling they are ' fitting in' with the world of ' pro' music should also not be underestimated.
However, the number and precision of the mouse operations required by most music applications, as well as their complexity of operation, makes this pathway unrealistic for by far the vast majority of switch users. This makes it desirable to provide a complementary pathway to using specialist software, to enable some switch users to try out standard software.

Thus, the challenge is to build a user interface *using an overlay system to provide the same level of high-level control, safety and feedback as in a custom system.* This can enable a switch-user to operate a music application without needing to understand its operation at a low-level - ie to choose from clearly presented musical operations, as in E-Scape.
The example overlay grids shown here are an initial attempt to provide such higher-level control of music software for switch users - ie to obviate them needing to

understand the interface of the underlying music application. As has been illustrated, no overlay system - however capable - is presently able to provide the same degree of control flow, economy of operation, guidance and feedback as a system dedicated to switch control. This, combined with the inherent difficulties of intercommunication of overlay and application has made it a somewhat complex and laborious task (not dissimilar to software programming itself) to construct and test these grids.

The results from the work done so far show how much can be done, but the difficulty of implementation does depend on the target application - which will typically *not* have been designed with switch, or overlay, control in mind. The limitations of overlay systems regarding control flow, context, feedback and communication of state also provide constraints on the degree of sophistication and flexibility achievable at present.

However, some recommendations - see Appendix 2 - can already be made for extended functionality for overlay systems to enable the construction of higher-level user interfaces outlined. It can be seen that the suggested level of functionality is really demanding that an overlay system take on some of the features of fully-fledged programming languages, and such a development is certainly an interesting and challenging project worth pursuing.
Such an overlay system would be a very powerful and important tool in providing access to desirable and powerful industry standard software for people whose degree of disability (lower communication and control bandwidth) has hitherto effectively prevented them realistically being able to operate such systems. It could also be applied to non-music software.

The Drake Music Project is therefore hoping to extend the sophistication of control via overlays to more approach that of dedicated systems, and is planning to develop control systems for a range of industry standard music software. This will also involve Drake working alongside Sensory Software, to recommend and test possible additional features which could be incorporated into their new overlay system ' The Grid' . These may make it easier to build the more complex overlay user interfaces discussed above.

These overlay resources, as well as dedicated systems such as E-Scape, will be made available on the new Drake Online web-site which aims to facilitate people to make music whatever their physical disability, and wherever they are. With the planned developments described, to these aims can be added ' with whatever software they like' - an ambitious but worthwhile goal.

APPENDIX 1 - EXAMPLES OF SPECIFIC PROBLEMS USING CAKEWALK WITH OVERLAY SOFTWARE

1. Problems when entering notes - see task A3
Note entry has to be done via ' step entry' , which involves specifying the note length in advance, then entering the pitch from a keyboard (on screen or external music instrument). A graphic keyboard can be opened, on which notes can be entered by clicking on them, but also (luckily) using 24 of the keys on the computer keyboard.

a. However to set which octaves the 24 keys are mapped to the possible 88 or more notes available, two *graphic* objects have to be dragged under the keyboard using the mouse.

b. All such entry has to occur within a small uninformative ' Step record' dialog box which mostly covers the edit screen. But the main problem is that the notes entered do not appear on the display - and can not be played - until the dialog box is closed via a ' keep' button. So while entering notes one can not see what is already there, and cannot hear it, or see where one is - all vital unless one is an advanced composer who can remember large amounts of musical information.

c. To then enter more notes one has to open the dialog again, but find that the settings for note length etc made before have been forgotten. To set the note length, a

shortcut can be used, to which an overlay button can be assigned. However, to change location, the focus must be tabbed to another area of the dialog, and the cursor keys used a certain number of times. Once at a certain value, the overlay has no way of knowing where it is, so subsequent changes of value would have to involve tabbing to other areas then back again (luckily the dialog goes back to the same default value each time this field is entered). All this means that for the user to enter a note then change the values for the next note is a complex, long-winded and problematic process.
In effect this means that the dialog must be opened then closed again for every note or chord to be entered, unless the user can assume to have good understanding of the order of fields within the dialog and where they are, and operate within it at a lower-level.

2. Problems in exiting the keyboard window - see task A5
The user can go back to the 'Keyboard' window and enter some more notes. They can then select [FINISH] on the note grid, which returns to the Step Record window and types <Return> which presses the window's 'Keep' button. But a problem can arise if the user decides to exit the keyboard window, having *not* recorded any further notes: if they select [CANCEL -back to main menu], this closes the 'Step Record' window (by typing <Esc>) which will not 'keep' the recording (to do that we needed to press the <Return> key).
Thus we really need an additional [Finish entering notes] cell which will type <Return> (not <Esc>) to press the 'Keep' button before closing. However, if the user were to select this cell when they had *not* recorded any notes, then the 'Keep' button will not be active, and so the Step Record window will *not close*, although we do jump back to the main Grid. This then leads to further problems: because the 'Step Record' window is now still open and active, the shortcuts typed by cells in the main Grid (which assume the window is not there) have erroneous effects and can leave the user 'stuck' - eg selecting [Enter notes] again will have the unintended effect of opening a 'Duration' dialog.
What is needed is for the [FINISH] cell to either press <Return> *or* <Esc> depending on whether the user has recorded any notes on the keyboard. This is a good example of the inherent difficulty in using an overlay, as it cannot know the status of the controlled application. One answer might be for any [note] cell in the Keyboard grid to jump to an identical looking 'Keyboard' Grid of notes with two differences: its [note] cells do *not* then jump to yet another grid; its [Change Note length] cell jumps to a variant of the 'Step Record' grid, which 'knows' that at least one note has now been entered, and whose [FINISH] cell *can* therefore safely press <Return> to finish correctly. This needless to say is rather unwieldy, as well as complex and laborious to set up, and is one area where it is hoped advanced developments within Sensory Software's new product 'The Grid' may help.

3 Problems entering data in the 'Select by time' dialog
Selecting of notes in Cakewalk is only provided by drawing round notes or clicking on them with the mouse. As illustrated above (figs 11-12), there is a dialog which lets you select all notes within a time range, with fields for 'from' and 'to'. A keyboard user can also type in time values directly, but non-standard Windows controls (described above) prevent the overlay being able to enter numbers into the fields. Thus the only way for the overlay to enter values into the dialog is to use the keyboard shortcuts (luckily available) to increment or decrement the time shown by 1 beat. For a cell in the grid to be able to enter a specific time, it has to press the increment key a certain number of times to get to the desired time, but to make sure it always starts from the beginning, the existing time show when the dialog first opens has to be removed by pressing the delete key.
Each of these 'from' cells also then jumps to a second grid, which has similar cells to set the 'to' time. Each of these cells has a similar set of key presses: first <tab> to move to the second 'to' field, then <delete> to remove the existing time, then an appropriate number of presses of the increment key, finally followed by <return> to close the dialog, then a jump back to the main grid. As with most dialogs the controls to eg play are not active while it is open.

By this somewhat convoluted series of key presses (eg below), the application could be persuaded to co-operate, and grids as illustrated were able to set the times to enable notes to be selected. However, it is only possible to select notes which lie within the *same* time span (eg individual notes within a chord) using low-level mouse control.

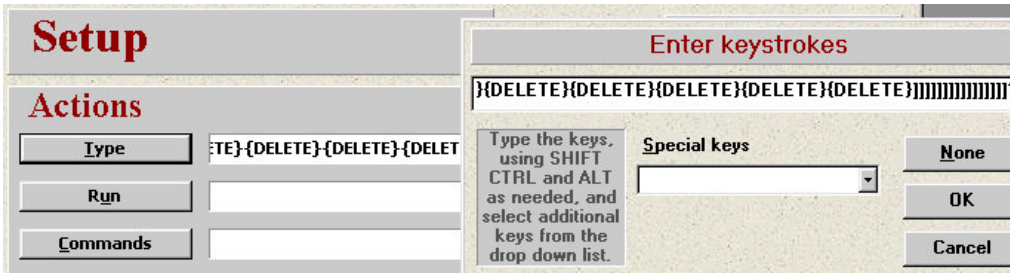

Fig. 14 A Hands Off cell definition 'workaround' to set a specific time

There are still several other problems which make the whole 'selecting notes' operation difficult:
a. the user needs to see and *interpret* the graphic display to divine which times to enter, in order to select the desired notes.
b. the time dialog obscures the edit window and refuses to be moved from the overlay's commands, so notes in the window can't be seen while being selected.
c. there is no visual feedback on which notes are being selected until the time span has been specified and the dialog closed.

4. Problems when entering chords
To enter *chords* (notes with the same start time), the button for 'auto step' must be switched off, as Cakewalk assumes that a user would usually enter chords by pressing several keys on the graphic keyboard at the same time. But each time the dialog is entered it has reset itself *on.* To audition the notes of the chord which have been entered already along with the new note is very long winded, as we need to exit the dialog, then go to another overlay grid to move the cursor back to the start of the chord, then play, then move the cursor back yet again, then go into the dialog again, then switch off 'auto step again' …
Again, with further work, this problem might be solved by a long-winded set of grids which repeatedly close the Step Record window, move the cursor and go back into it again, but this is not straightforward, as there is no way the overlay can *remember* what the last step duration was set to by the user, and there is no way of passing arguments within the overlay. Thus each [duration] cell would have to exit the window and jump to a *different* other grid just to set the step cursor back the right amount; then the user would have to select a single [next chord note] cell, making for a very complex overlay structure, with dozens of variant grids needed.
A better solution would be to have a cell labelled [Start chord] to switch on a key lock, before selecting several cells for chord notes, then finally selecting another cell to release the held keys.

5. Other general overlay problems
a. There is a limit to the length of text which can be displayed in a grid (as a title), so the presentation of instructions or prompting is limited.
b. As each grid has fixed cells with fixed values, a grid can have cells which allow a user to select inappropriate values, eg to specify a 'from' time greater than the 'to' time.
c. Cells have to be in a grid-like arrangement, which limits the ability to present a visually appropriate design - eg a piano keyboard.

APPENDIX 2 - INITIAL RECOMMENDATIONS FOR FUNCTIONALITY OF OVERLAY SYSTEMS

These initial recommendations would help solve some of the problems encountered using overlays to build higher-level user interfaces. Some may prove problematic or impractical, or may need refining, and more will undoubtedly arise during the project.

1. Each overlay grid should be able to contain remembered state (similar to OOPS instance variables), and be able to pass and be passed parameters or arguments from other grids when jumped to or from, or from cells when activated. This will enable a grid to eg know and remember how many times a switch press has been repeated, or which cell was last selected. Another example would be to launch a grid forming a menu of numbers which can be specified dynamically at launch. This structure and parameter passing is likely to be the key to implementing some of the ideas below.
2. Grids should be able to take actions immediately on launch in response to parameters passed from the calling grid. Such structure would for example make eg, the ability to jump to ' previous grid but two' easy to implement.
3. Conditional jumping to different grids, depending on recent actions in the grid. This will depend in the provision of remembered states in grids as above.
4. Cell actions should be able to be nested - so that a complex cell action can be set up by calling other cells already defined. ' The Grid' will address this using ' Scripts' .
5. Each grid should be able to have (lengthy) text at the top or bottom, which can automatically speak on launch of the grid - to act as an instruction, question or prompt.
6. Text should be dynamically assignable when a grid is launched, passed as parameters from the calling cell.
7. For places where there are only a few options, a grid should be able to operate with a single row or column (to act as a conventional menu), and only require a single switch press to select a cell.
8. Each grid should remember which grid it was called (jumped to) from, as well as grids it then jumps to. At present, loops can be formed with two grids jumping back to each other.
9. Cells need to be arrange-able in groups within a grid, with the possibility of empty space between, and not restricted to a grid-like arrangement - eg which could enable a proper looking piano keyboard to be constructed.
10. Cells should be able to send MIDI messages, to trigger music synthesisers - eg useful when scanning a grid of notes. These again will need to have parameters or state passed from the grid or cell, so that, for example, a grid can send on the correct MIDI channel depending on which track in the application has been selected.
11. Grids should respond to incoming MIDI messages, so they can be controlled by MIDI instruments. Many users like to only have a single controller in front of them, and would like to press notes on a MIDI keyboard to scroll and select, for example.

REFERENCES

Anderson, T. M. and Smith, C. (1996). ‘Composability’: widening participation in music making for people with disabilities via music software and controller solutions’. Proc. ACM Conference on Assistive Technologies, Vancouver.

Anderson, T.M. (1997). Making music with Computers. Ability 20: 9-15.

Anderson, T.M. (1999). Using Music Performance Software with Flexible Control Interfaces for Live Performance by Severely Disabled Musicians. Proc. EuroMicro 25(Vol.2): 20-27.