

# The Design of a Sonically-Enhanced Interface Toolkit

*Stephen A. Brewster*

Department of Computing Science  
The University of Glasgow  
Glasgow, G12 8QQ, UK  
Tel: +44 (0)141 330 4966  
Fax: +44 (0)141 330 4913  
stephen@dcs.gla.ac.uk  
<http://www.dcs.gla.ac.uk/~stephen/>

## ABSTRACT

This paper describes the design of a user-interface toolkit composed of sonically-enhanced widgets. Research has shown that the addition of sound can improve the usability of human-computer interfaces. However, sonically-enhanced widgets are difficult to create. The motivation for this work is the same as that which motivated the creation of graphical interface toolkits: To simplify their construction; to allow designers who are not sound experts to create such interfaces; to make sure the sonically-enhanced widgets are effective and improve usability; and to make sure the widgets use sound in a clear and consistent way across the interface. This paper describes the design of the audio feedback for several of the widgets in the toolkit, specifically the design of buttons, menu, scrollbars and windows.

**Keywords:** Earcons, auditory interfaces, toolkits, non-speech audio, sonically-enhanced widgets.

## INTRODUCTION

Why should sound be added to human-computer interfaces? There is a growing body of research which indicates that the addition of non-speech sounds to human-computer interfaces can improve performance and increase usability [5, 10, 15, 21]. Non-speech sound is an important means of communication in the everyday world and the benefits it offers should be taken advantage of at the interface. Such *multimodal* interfaces allow a greater and more natural communication between the computer and the user. They also allow the user to employ the appropriate sensory modalities to solve a problem, rather than just using one modality (usually vision) to solve all problems.

In spite of the increased interest in multimedia, little solid research has been done on the effective uses of sound in computers even though many computer manufacturers now include sound producing hardware in their machines. Arons & Mynatt [1] suggest one reason for this: "...the lack of design guidelines that are common for the creation of graphical interfaces has plagued interfaces designers who want to effectively build on previous research in auditory interfaces".

Using sound can be beneficial but, because this area is still in its infancy, sounds are often added in *ad hoc* ways by individual designers and this can lead to them being ineffective [2, 22]. The aim of the research described here is to help designers to create effective sonically-enhanced interfaces. This paper describes the development of a sonically-enhanced, multimodal toolkit that provides for the use of sound in an effective and consistent way across the human-

computer interface. Each interface widget will be analysed to find usability problems and sound added to overcome these. Sound can be used to replace graphical feedback with more an effective auditory form or enhance the existing graphical feedback. Once each widget has been sonified they can be combined into a complete widget set. This widget set will allow a designer to create an interface that makes effective use of sound.

Sound has many advantages. It is omni-directional so the user does not have to concentrate on a particular part of the display to perceive it. In fact he/she does not even have to be looking at the display at all. It takes up no valuable space on the screen. Sound is also attention grabbing and can be effectively used to indicate problems to users. It can work alongside synthetic speech in purely auditory interfaces or be integrated with graphical feedback. Often in graphical interfaces more and more information is displayed on-screen. This can result in overload and important information may be missed. One way to overcome this problem is to use sound. Important information can be displayed on the screen and other information in sound, reducing overload of the visual sense. Brewster *et al.* [7] showed that by adding sound to a graphical interface both the time taken to complete certain tasks and the time taken to recover from errors could be reduced.

#### **Aims of the toolkit**

The four main aims of this toolkit are similar to those that motivated the development of graphical interface toolkits. These are:

- *To simplify the implementation of applications that include sound in their interfaces.* Currently it is very difficult to create sonically-enhanced applications. It is difficult to write the code necessary to include the sounds because it is usually very device-dependent and takes the implementer much time to write. This is a similar problem to that faced by interface designers before graphical toolkits were available; much time had to be spent writing the device dependent parts of the graphical interfaces. Myers [20] suggests that the use of graphical toolkits significantly reduces the development time of graphical interfaces. This toolkit will do the same for sonically-enhanced interfaces. The toolkit will include the code for controlling sounds therefore all an interface designer must do is include the new toolkit in his/her application.
- *To allow designers who are not sound experts to create sonically-enhanced interfaces.* Interface designers are not often skilled in sound design. A toolkit that has the sounds included would remove the need for detailed knowledge of sound design. This again follows the same approach as graphical toolkits in that an interface designer without a detailed knowledge of graphic design can create an interface using a standard graphical interface toolkit.
- *To make sure that the sounds added are effective and enhance the user's interaction with the computer.* The sounds added will not be gimmicks. Detailed investigations of usability problems will show where sounds can help usability. Sounds will be added to overcome these problems, so improving usability.
- *To make sure the sounds are used in an clear and consistent way across the interface.* This will avoid the problems of each application having its own sounds that mean different things to other applications. In graphical interface

toolkits, the widgets look consistent across different applications, a scrollbar looks the same in any application where it is used. In the sonically enhanced toolkit, widgets will sound consistent across different applications.

This paper brings together previous work on individual sonically-enhanced widgets to form a complete interface toolkit. In each of the widgets sound is used to support graphics. The motivation is that users' eyes cannot do everything. The visual system has a small area of focus. If users are looking at one part of the display then they cannot be looking at another part at the same time. In highly complex and detailed graphical interfaces the user must concentrate on one part of the display to perceive the graphical feedback. It is suggested here that other information should be presented in sound. This will allow users to continue looking at the information required but to hear information that would otherwise not be seen (or would not be seen unless they moved their visual attention away from the area of interest, so interrupting the task they are trying to perform). Sound and graphics will be used together to exploit the advantages of each.

### **SOUNDS USED**

The non-speech sounds used for this investigation are based around structured audio messages called *Earcons* [4, 5, 25]. Earcons are abstract, synthetic tones that can be used in structured combinations to create sound messages to represent parts of an interface. Detailed investigations of earcons by Brewster, Wright & Edwards [8] showed that they are an effective means of communicating information in sound.

Earcons are constructed from motives. These are short rhythmic sequences that can be combined in different ways. The simplest method of combination is concatenation to produce *compound earcons*. By using more complex manipulations of the parameters of sound *hierarchical earcons* can be created [4] which allow the representation of hierarchical structures.

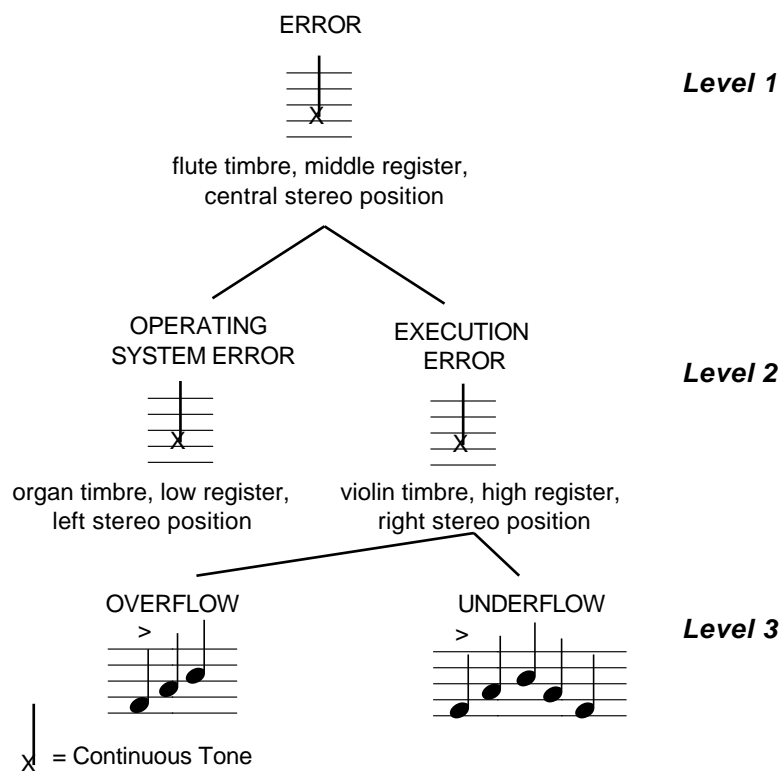
Figure 1 shows a simple hierarchy of earcons. Each earcon is a node on a tree and inherits all the properties of the earcons above it. The different levels are created by manipulating the parameters of earcons (for example, rhythm, pitch, timbre, register, tempo, stereo position, effects and dynamics). In the diagram the top level of the tree is a neutral earcon representing a fictitious family of errors. It has a flute timbre (a 'colourless' instrument), a middle register and a central stereo position. The structure of the earcon from level 1 is inherited by level two and then changed. At level 2 the sound is still continuous but non-neutral timbres are used (in the figure organ and violin). Register is changed so that it matches a conventional musical layout (low register on the left, high on the right) and stereo position reflects the layout of the hierarchy, for example the node on the left has a left stereo position. At level 3 a rhythm is added to the earcon from level two to create a sound for a particular error. This rhythm is based on the timbre, register and stereo position from the level above. Other levels can be created by using other parameters such as tempo or effects.

Using earcons this hierarchy is easily extensible. For example, to add another major category of errors all that is needed is a new timbre. This could then be given a middle register and a middle stereo position and be placed between the operating system and execution errors in the diagram. To create a new type of execution error only a new rhythm is

needed and it can be added to the existing hierarchy. Therefore earcons provide a very flexible system for creating hierarchies.

### OVERALL STRUCTURE OF THE SONICALLY-ENHANCED TOOLKIT

Each of the widgets in a standard widget set will be sonified. The overall structure of the sounds will be as follows: Each application will have its own timbre and spatial location (via stereo) as a base for all of its sounds. All widgets within an application will use these and modify them by changing the rhythm, pitch, etc. Figure 2 shows such a hierarchy. At level 1, the three applications all have different timbres and spatial locations. These are inherited by level 2 and modified with pitch, rhythm, etc. These modifications are constant across applications so that widgets in different applications sound consistent (similar to graphical widgets that look consistent across applications). It is hoped that after using the system, users would come to associate a certain timbre with a particular application.



**Figure 1:** A hierarchy of earcons representing errors.

As an example, consider a button widget in the three applications in Figure 2. If a button was used in the Write application it would have the Write instrument, for example an organ, and stereo position, for example on the left of the stereo space. The button would have its own rhythm and note structure, for example a two note chord. In the Write application this chord would be played by an organ in the left of the stereo space. If the Spreadsheet application had a piano timbre and a right stereo position, then the same two note chord would be played but modified by that instrument and stereo position. In this way the earcons for each widget would be consistent across the whole interface (the button would always use the same two note chord) but would also fit with the sounds of the application of which it was part.

The sounds will be controlled by MIDI [18]. Almost all modern computer systems either have built-in MIDI-controlled synthesisers (for example on sound cards or via DSP chips) or can easily be connected to them. Using MIDI will also provide an easy way for users to customise the sounds. Standard synthesiser control software can be used to change the timbre or intensity of the sounds in any widget. In practice, it may not be possible to give all applications a different timbre and spatial location (synthesisers often only have 127 different timbres and not all of these are distinct). Instead, applications could be grouped (for example, communications, text processing or graphics) and each group given a different timbre and location. Different applications within a group could then be separated by register.

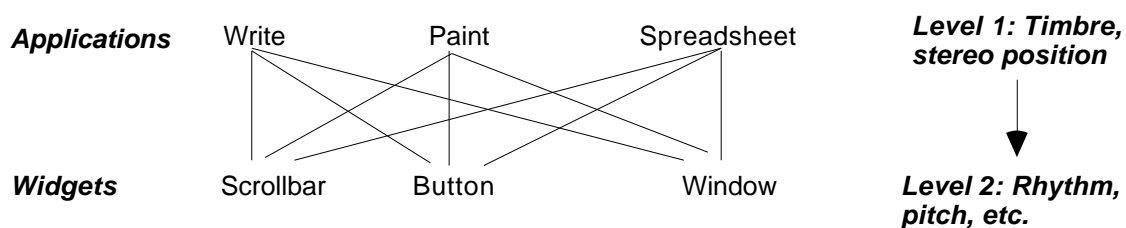
### SONICALLY-ENHANCED WIDGETS

The earcons to be used have now been described but what should they be used for? Earcons will be used to overcome usability problems in standard graphical widgets. Each of the widgets in a standard toolkit will be analysed to discover any usability problems. Brewster and colleagues [5, 7] proposed a structured method that can be used to analyse interactions in terms of event, status and mode information to find where usability problems may occur due to incorrect feedback. From this analysis earcons can be created for the auditory feedback. These new widgets can then be experimentally to ensure the sonic enhancements improve their usability. All of the earcons described in the widgets below were designed using the guidelines for the creation of earcons by Brewster *et al.* [9].

The following sections describe some of the sonically-enhanced widgets in the toolkit. For each of the widgets, the usability problem to be corrected will be described and then the sonic enhancements discussed. This paper will just discuss the design of the widgets, for more details on the experiments to test their usability the reader is referred to the references given in the appropriate section. The widgets will be described below as if they were part of an application that used an electric organ timbre and central stereo position as the base for its sounds.

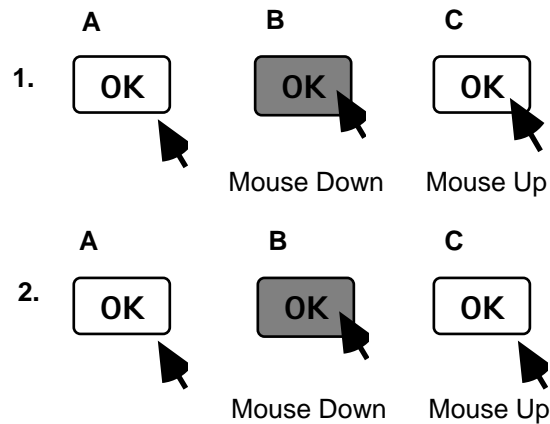
#### Sonically-enhanced buttons

One of the most fundamental widgets in all graphical human-computer interfaces is the graphical button (to avoid confusion, *graphical button* will here be used to refer to the button on the computer display and *mouse button* will be used to refer to the button on the mouse). Although these are very common they are not without problems [12, 13]. One of the main difficulties is that the user may think the graphical button has been pressed when it has not. This can happen because the user moves off the graphical button before the mouse button is released. This is caused by a problem with the feedback from the graphical button (see Figure 3). Both correct and incorrect presses start in the same way (1A and 2A). In the correct case, the user presses the graphical button and it becomes highlighted (1B), the mouse button is then



**Figure 2:** A hierarchy of sonically-enhanced widgets across applications.

released with the mouse still over the graphical button, it becomes un-highlighted (1C) and the operation requested takes place. The button slip-off starts in the same way. The user presses the mouse button over the graphical button (2B), then moves (or slips) off the graphical button and releases the mouse button (2C), the graphical button becomes un-highlighted (as before) but no action takes place. The feedback from these two different situations is *identical*. This problem occurs infrequently but, as the error may not be noticed for a considerable time, the effects can be serious. With a one-step undo facility users must notice before the next action takes place otherwise they may not easily be able to correct the mistake.



**Figure 3:** Feedback from pressing and releasing a graphical button.

(1) shows a correct button selection, (2) shows a slip-off.

The identical feedback would not be a problem if the user was looking at the graphical button to see the slip-off, but this is not the case [13]. Dix & Brewster suggest there are three conditions necessary for such slip-off errors to occur:

- i) The user reaches closure after the mouse button is depressed and the graphical button has highlighted.
- ii) The visual focus of the next action is at some distance from the graphical button.
- iii) The cursor is required at the new focus.

Closure occurs when a user perceives a task as being completed, which in this case is when the graphical button is highlighted (the mouse button is down). In reality, the task does not end until the mouse button is released. Because closure (i) is reached before this, the user starts the next task (mouse movement, iii) in parallel with the mouse up and a slip-off occurs. The user's attention is no longer at the graphical button (ii) so the feedback indicating a slip-off is not noticed. The problem occurs with expert users who perform many operations automatically and do not explicitly monitor the feedback from each interaction. This type of error is an *action slip* [23], as Lee [17] describes (p 73): "...as a skill develops, performance shifts from 'closed loop' control to 'open-loop' control, or from monitored mode to an, automatic, unmonitored mode of processing." As users become familiar with a task they no longer monitor the feedback so closely. Many of the errors described in the widgets below are action slips; users do not monitor the feedback from the widget but continue to look at the information in which they are interested. If they must look at the widget then it forces the interface to intrude upon the task they are trying to perform.

These problems occur in graphical buttons that allow a 'back-out' option: Where the user can move off the graphical button to stop the action. If the action is invoked when the mouse button is pressed down on the graphical button (instead of when it is released) then these problems do not occur as the user cannot slip off. These buttons are less common because they are more dangerous as users cannot change their minds.

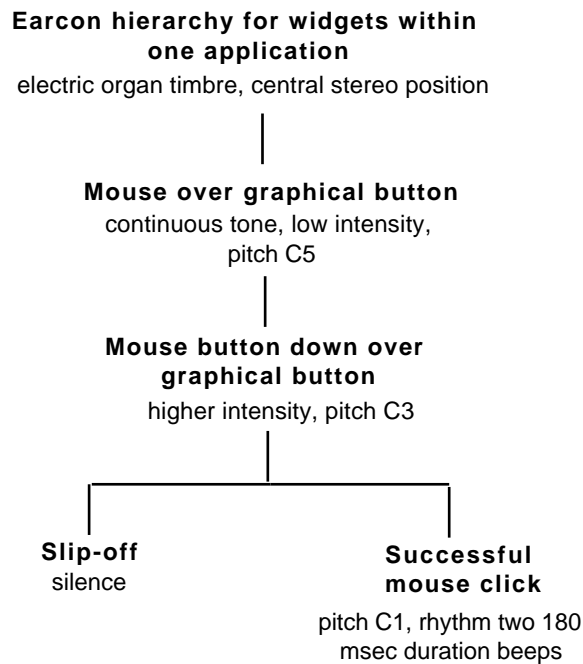
In this situation sound should be used because the user's eyes are occupied. Moving the mouse to the location of the next action requires visual attention so that the mouse can be positioned correctly. Therefore, the user cannot look at the button to see feedback indicating a slip-off. It would be very difficult to correct this problem using visual feedback. Buttons could be changed so that they indicated a difference between a successful and unsuccessful click. For example, the button could flash in a different way. This would not work because users will not be looking at the button but at the location of the next action. The area of visual focus is too small to allow them to see the feedback. Perhaps feedback could be given at the mouse location but again we cannot be sure that the user will be looking there either. Sound would allow us to present the information to the user without knowing where he/she was looking.

#### *The design of the sonically-enhanced buttons*

Three sounds were needed to overcome the usability problems discussed above: One to indicate to the user when the mouse was over a graphical button; one to be the auditory equivalent of the graphical highlight when the mouse button was pressed down on the graphical button; the other to indicate when a button was pressed correctly or when a slip-off occurred. Figure 4 shows the hierarchy of earcons used. The top of the hierarchy shows the instrument and stereo position to be used for the application. Each level inherits properties from the levels above. For example, at level 3 (mouse down over graphical button) the timbre and stereo position are inherited from level 1 and the continuous tone from level 2. At this level, the new intensity and pitch overwrite those from level 2. For more details on the experiment see Brewster *et al.* [6].

A base sound was created for when the mouse was moved over a screen button. This was a continuous tone at C<sub>4</sub> (130Hz). The volume of this was kept to just above the threshold level. This sound was played as long as the mouse was over a graphical button; it stopped when the mouse was moved off. When the mouse button was pressed down over a graphical button a continuous sound at pitch C<sub>3</sub> (Middle C, 261Hz) was played. This continued for as long as the mouse button was down and the mouse was over the graphical button. If the mouse was moved off the graphical button the sound stopped. If the mouse was released over the graphical button then a success sound was played. This consisted of two notes, played consecutively, at C<sub>1</sub> (1046Hz) each with a duration of 40 msec. This success sound had to be kept short so that users would not get confused as to which button the feedback was coming from: The audio feedback had to be able to keep pace with interactions taking place. To make sure that the number of sounds was kept to a minimum and speed maximised, if a user quickly clicked the mouse over a graphical button the mouse down sound was not played; only the success sound. The mouse button down and success sounds differentiated a successful and unsuccessful mouse click.

The two sounds used a combination of pitch, duration and intensity to get the listener's attention. This meant that a lower level of intensity could be used, making the sounds less annoying for the primary user and others working nearby. Annoyance is most often caused by excessive intensity [3]. It is important to note that intensity is not the only way to get the user's attention. The human perceptual system is good at detecting dynamic stimuli. As Edworthy *et al.* [14] showed, attention-grabbing sounds can be created by varying other sound parameters, such as those used here.



**Figure 4:** The hierarchy of earcons used in the sonically-enhanced buttons.

### Sonically-enhanced menus

Menus are very similar to buttons. Menu items are selected by moving up or down a menu with the mouse button pressed down and then releasing the button over the required item. Users can back-out if they want to by moving off the menu but this facility can again lead to slip-off errors. In some systems (such as the Macintosh) the menu can be made to flash if a correct selection is made. This is different to graphical buttons in that there is a difference in the feedback between a correct and incorrect menu selection. However, as mentioned above, it may still not help as the user will be looking at the location of the next action rather than the menu. The narrow focus of visual attention then prevents him/her from seeing the feedback.

There is another problem that menus have which buttons do not. In a menu the user can slip-off the required item on to one above or below. This will still be presented as a correct selection (the user chose an item from the menu correctly) but will not be the item required. This can happen, in a similar way to that described for buttons, because the user moves the mouse from the menu to the location of the next interaction. The movement overlaps with the release of the mouse button so that the user releases in the wrong menu item. The user will be about to release on the item required, will start to move the mouse to the location of the next action and so slip into one of the other menu items for a very short time and then release. This behaviour is described in more detail by Crease [11].

### *The design of sonically-enhanced menus*

The sounds were based around those in the sonically-enhanced buttons above [11]. A continuous low note at pitch C<sub>4</sub> (130Hz) was played when the user pressed the mouse down in the menu bar. The volume was kept to just above the threshold level and it was played as long as the mouse was in a menu; it stopped when the mouse was moved outside the menu. This indicated to the user that he/she had slipped-off the whole menu.

When moving within the menu, if the user stayed on an item for more than 400 msec. a higher pitched constant tone was played. This was similar to the graphical highlight that indicates which item is selected. This highlight tone alternated in pitch between E<sub>3</sub> (330Hz) and B<sub>2</sub> (988Hz) for each menu item. For example, the first item in the menu would be played at E<sub>3</sub>, the second at B<sub>2</sub> and the third at E<sub>3</sub>. If the mouse was released over an item then a success sound (similar to the correct button press sound above) was played. This consisted of two notes each with a duration of 40 msec played consecutively at the pitch of the highlight tone for the menu item. These earcons helped with the item slip-off problem. In the case of a correct selection, the user would move to the item required, wait for a moment and then release. The highlight sound for the menu item would be played followed by the success sound. In the case of an item slip-off, for example the user wanted to select the second item on the menu but slipped off onto the third, the user would move to the item required, wait for a moment and hear the highlight sound at pitch B<sub>2</sub> but then just as the mouse was to be released he/she would slip-off to the item below. He/she would then hear the success sound played at E<sub>3</sub>. This would indicate that a mis-selection had taken place.

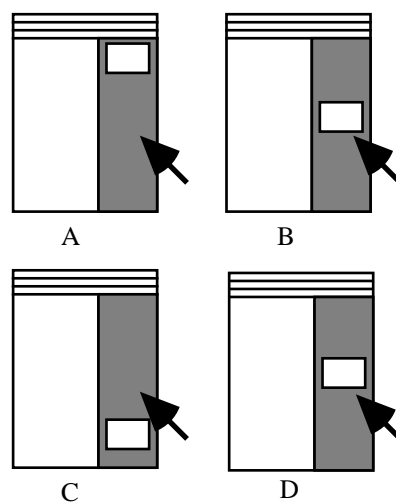
### **Sonically-enhanced scrollbars**

There are several usability problems with scrollbars [7]:

- *Position awareness in documents:* When scrolling through a document it can be hard to maintain a sense of position. The text can scroll by too fast to see (and the thumb wheel only gives an approximate position). The user really wants to look at the information in the window, not the scrollbar. However, in order to get location information he/she must look at the scrollbar, forcing visual attention away from the document which is what is really of interest and causing the interface to intrude into the task being performed. As before, the user cannot be looking in two places at once. Some systems, such as Microsoft Word, put a page count at the bottom left hand corner of the screen but this is too far from the centre of visual focus - again the user must move his/her visual focus to the page counter. Sound can be used to present this location information so that the user can perceive it wherever he/she is looking.
- *'Kangarooing' with the thumb wheel:* Repeatedly clicking in the scroll area above or below the thumb wheel scrolls by a window-sized step. Clicking below the thumb scrolls down in the document and clicking above scrolls up. Figure 5 shows an example of kangarooing. In A the user begins clicking to scroll down towards the mouse pointer. In B the thumb wheel is just above the pointer. In C the user has clicked and the thumb has scrolled below the pointer. In D the user clicked again and this time the thumb scrolled back above the pointer and kangarooing occurred. Unless he/she is looking at the thumb it can be hard to recognise that this has happened. If the user continues clicking the thumb wheel will bounce above and below the mouse pointer location. If clicking is rapid, this can be difficult to see because the information in the window is changing rapidly. Information about

kangarooing errors could be presented in sound so that the user could see if even if he/she is looking at the document.

- *Moving the mouse pointer outside the thumb wheel:* Dragging the thumb wheel allows the user to scroll by an arbitrary amount. When dragging in systems such as the Macintosh, the document does not scroll until the mouse button is released. When dragging the thumb up or down in the document one may move too far to either the top, bottom, left or right of the scroll bar and the thumb is dropped. The document then does not scroll when the mouse is released and the user may become confused. Often, only the outline of the thumbwheel is dragged, the real one stays where it is, and this does not grab the user's attention because it is harder to see. Dragging within the area of the scrollbar requires visual attention to be taken away from the document, if this information was presented in sound, users would be able to hear when they had moved outside of the scrollbar area.



**Figure 5:** Scrollbar 'kangarooing'.

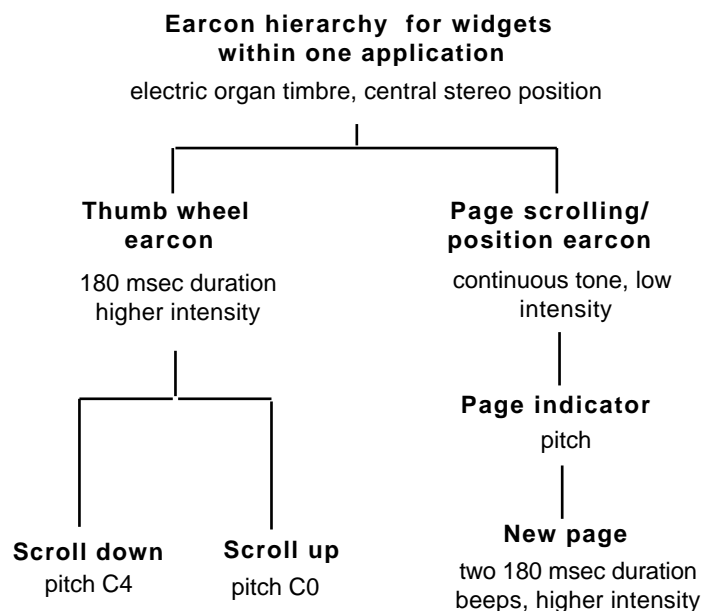
#### *The design of the sonically-enhanced scrollbars*

Two types of sounds were needed to solve the problems described above: One to give scrolling information to indicate when the thumbwheel reached the target location (to avoid kangarooing) and one to give location information. Figure 6 shows the hierarchy of earcons used. See [7] for more details.

The first sound was a fixed tone of duration 180 msec. and was used to indicate a window scroll event with the thumbwheel. The sound was kept short so that it could keep up with the interactions taking place. If the user scrolled towards the bottom of a document, the short tone was played at a low-pitch, C<sub>4</sub> (130Hz). When scrolling up a high-pitched note C<sub>0</sub> (2093Hz) was played. High pitch was used as up and low pitch as down because of the natural tendency to perceive higher pitch as higher spatial location [19]. If a user was clicking to scroll down towards the mouse pointer location he/she would hear the repeated low-pitched sound. If kangarooing occurred then the user would hear a demanding high-pitched tone when not expected and this would indicate the error.

A low intensity, continuous tone was used to give location information. This earcon changed in pitch when a page boundary was crossed; lower pitch when scrolling downwards and higher when scrolling upwards. To indicate a page boundary event the background tone was increased in volume for two tones of 180 msec. each to demand the listener's attention. It then decreased again to just above threshold level so that it could be habituated. The different number of notes differentiated this earcon from the previous window scroll sound. Again the sound was short so that it did not hold up the interaction. The notes played when scrolling towards the bottom of the document decreased in pitch from B<sub>1</sub> (1975Hz) to C<sub>4</sub> (130Hz) when a page boundary was crossed. The reverse occurred when scrolling up from the bottom of the document. When the scrollbar was clicked the thumb sound was played first followed by the page boundary sound after a 180 msec. delay (if a page boundary had been crossed).

The continuous location sound was only played when the mouse pointer was in the scrollbar area. If the user moved the mouse out of the scrollbar then the sounds stopped. The continuous sound would be annoying if played all the time, so it was only played when the user was engaged in scrolling. This earcon was also used to stop the problem of moving the mouse pointer outside the thumb. When this happens the sounds would stop so that users could then tell (by the lack of sound) that they had slipped out of the scrollbar. When dragging the thumb the earcon would change in pitch as described above for the location indicator, changing when the user dragged over a page boundary. This would give location information when the user's eyes were busy looking at the document. The same earcon would be used for scrolling with the arrow buttons at the top or bottom of the scrollbar (which allow line by line scrolling). The location sound would play and change when the user crossed a page boundary. If the user moved the mouse out of the arrow then the sound would stop, indicating that the mouse had slipped off the button.



**Figure 6:** The hierarchy of earcons used in the sonically-enhanced scrollbars.

### Sonically-enhanced windows

One common error in multiple window environments is the 'unselected-window' error [5]. This occurs when the user tries to interact in one window but another is actually the selected, active one. This may cause keystrokes to be lost or

errors to occur because the selected window is interpreting commands not meant for it [16]. There are two situations where this may occur [16]:

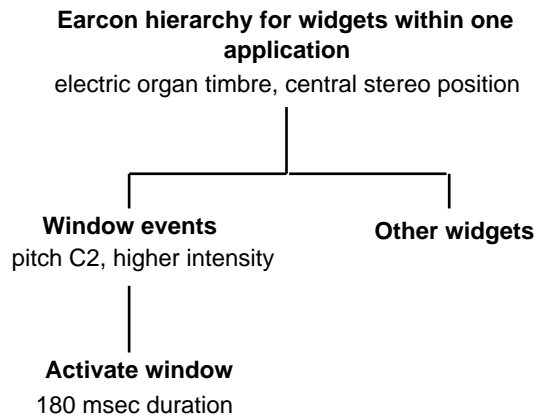
- On resumption after a pause the user's focus has changed so that he/she is no longer ready to engage in the interaction the system is expecting. For example, coming back to the system after having been away for some time, the user must re-orient to the system. Often the first task to be done will cause the user to re-orient to that window because it is a priority, whereas another window is really the active one.
- Discontinuity of input, for example when the mouse is knocked into another window by mistake (in systems where the mouse need only be placed in a window for that window to become the active one). The user still thinks the old window is active but it is not. A further example is when a user activates another window to look at some data, but then fails to re-activate the main window again when switching attention back to it.

The way to solve this problem is to reinforce the feedback from the active window so that the user does not mistake it. As Reichman says ([24], p 296): "Most of these systems mark the active window by a little blinking pointer or a highlighted title bar. These visual indicators are too limited". The user is likely to be looking at the contents of a window rather than the title bar. This means that it is outside the area of visual focus so that the user will not see the grey indicating an inactive window. Using sound, information about the active window can be communicated to the user wherever he/she is looking.

#### *The design of the sonically-enhanced windows*

Figure 7 shows the hierarchy of sounds used in the sonically-enhanced windows. As before, the example used is an application that has an electric organ timbre and a central stereo position. The auditory feedback to reinforce the active window on resumption after a pause came partly from the other widgets described above. If users pressed a button, used a menu or scrollbar, they heard the timbre and stereo position of the currently active application. This may have been the sound of the application they were expecting so there was no error. However, if they heard the sound of an application they were not expecting they would notice that an unselected window error had been made. Using sound does not rely on users noticing the colour of the title bar of the window, the sounds could be heard wherever they were looking in the window.

Audio feedback was also be used to overcome the problem of discontinuity of input. The event of switching from one window to another was marked by a short, high-pitched note in the new window's timbre at increased intensity. For example, when switching from another application to our example one, an electric organ note at pitch C<sub>2</sub> (523Hz) was played for 180 msec. in the central stereo position. This would indicate to the user if, for example, the mouse had been knocked into another window by mistake. Other window events, such as dragging or re-sizing the window could also be indicated in the same way.



**Figure 7:** The hierarchy of earcons used in the window experiment.

### Results and future work

These widgets have been implemented and experimentally tested to discover if the sonic enhancements improved usability. The results were very promising [5]. For example, the sonically-enhanced buttons were given a significantly higher overall preference rating by users; users preferred them rather than the standard graphical buttons. The time and number of mouse-clicks needed to recover from slip-off errors were both significantly reduced. This was not at the expense of making the buttons more annoying to use. There was also no difference in terms of annoyance between the standard buttons and the sonically-enhanced ones. These results indicate that, if sonically-enhanced buttons were included in an interface toolkit, slip-off problems can be dealt with very effectively [6].

As another example, sonically-enhanced scrollbars were tested and found to significantly reduce time taken to complete certain tasks. As with sonically-enhanced buttons, error recovery times were significantly reduced. Sonically-enhanced scrollbars were again given a significantly higher overall preference rating by users. They also significantly reduced the mental workload felt by users when performing scrolling tasks [7]. There was no increased annoyance due to the sounds. This again indicates that the integration of auditory feedback into graphical widgets is likely to provide more usable interfaces.

Each of the widgets in the toolkit has now been sonified and tested. The next stage is to combine them into a complete toolkit. They have been designed so that this should be simple. As shown in the sonically-enhanced windows above, one group of auditory widgets can provide some of the feedback needed by another. Once the complete toolkit has been built applications with sonically enhanced interfaces will be much easier to create.

### CONCLUSIONS

The aim of this research is to produce a toolkit that interface designers can use to integrate effective sounds into their interfaces. Currently this is very difficult to do because of the programming necessary and because most interface designers are not sound experts. Having a toolkit would allow designers to include sound by using a standard set of sonified interface components. The interface components have been tested to make sure the sounds are added in ways that improve usability. They will also be consistent across the interface to create a coherent sounding environment. Designers will no longer have to add sounds to their interfaces in *ad hoc* ways, they will be able to use a toolkit to allow

them to add effective sounds to improve interactions. This will allow them to build upon previous research and speed up the development of systems with sonically-enhanced interfaces.

The motivation for adding the sounds is that users cannot look at two places at once. Users really want to look at the information involved in their current task rather than at interface components. Using sound allows users to concentrate their visual attention on the information they are interested in and get other feedback through sound, it does not force them to take their eyes off their task and does not allow the interface to intrude into the task being performed.

## REFERENCES

1. Arons, B. and Mynatt, E. The future of speech and audio in the interface. *SIGCHI Bulletin* 26, 4 (1994), 44-48.
2. Barfield, W., Rosenberg, C. and Levasseur, G. The use of icons, earcons and commands in the design of an online hierarchical menu. *IEEE Transactions on Professional Communication* 34, 2 (1991), 101-108.
3. Berglund, B., Preis, A. and Rankin, K. Relationship between loudness and annoyance for ten community sounds. *Environment International* 16 (1990), 523-531.
4. Blattner, M., Sumikawa, D. and Greenberg, R. Earcons and icons: Their structure and common design principles. *Human Computer Interaction* 4, 1 (1989), 11-44.
5. Brewster, S.A. *Providing a structured method for integrating non-speech audio into human-computer interfaces*. PhD Thesis, University of York, UK, 1994.
6. Brewster, S.A., Wright, P.C., Dix, A.J. and Edwards, A.D.N. The sonic enhancement of graphical buttons. In *Proceedings of Interact'95* (Lillehammer, Norway) Chapman & Hall, 1995, pp. 43-48.
7. Brewster, S.A., Wright, P.C. and Edwards, A.D.N. The design and evaluation of an auditory-enhanced scrollbar. In *Proceedings of CHI'94* (Boston, Ma.) ACM Press, Addison-Wesley, 1994, pp. 173-179.
8. Brewster, S.A., Wright, P.C. and Edwards, A.D.N. An evaluation of earcons for use in auditory human-computer interfaces. In *Proceedings of INTERCHI'93* (Amsterdam) ACM Press, Addison-Wesley, 1993, pp. 222-227.
9. Brewster, S.A., Wright, P.C. and Edwards, A.D.N. Experimentally derived guidelines for the creation of earcons. In *Adjunct Proceedings of HCI'95* (Huddersfield, UK), 1995, pp. 155-159.
10. Brown, M.L., Newsome, S.L. and Glinert, E.P. An experiment into the use of auditory cues to reduce visual workload. In *Proceedings of CHI'89* (Austin, Texas) ACM Press, Addison-Wesley, 1989, pp. 339-346.
11. Crease, M. *Making Menus Musical*. Undergraduate Thesis, University of Glasgow, 1996.
12. Dix, A., Finlay, J., Abowd, G. and Beale, R. Chapter 9.4 Status/Event Analysis. In *Human-Computer Interaction*(Eds.), Prentice-Hall, London, 1993, 325-334.

13. Dix, A.J. and Brewster, S.A. Causing trouble with buttons. In *Ancillary Proceedings of HCI'94* (Sterling, UK) Cambridge University Press, 1994.
14. Edworthy, J., Loxley, S., Geelhoed, E. and Dennis, I. The perceived urgency of auditory warnings. *Proceedings of the Institute of Acoustics* 11, 5 (1989), 73-80.
15. Gaver, W., Smith, R. and O'Shea, T. Effective sounds in complex systems: The ARKola simulation. In *Proceedings of CHI'91* (New Orleans) ACM Press, Addison-Wesley, 1991, pp. 85-90.
16. Harrison, M. and Barnard, P. On defining requirements for interaction. In *Proceedings of the IEEE International Workshop on requirements engineering*(Eds.), IEEE, New York, 1993, 50-54.
17. Lee, W.O. The effects of skill development and feedback on action slips. In *Proceedings of HCI'92* (York, UK) Cambridge University Press, 1992, pp. 73-86.
18. Loy, G. Musicians make a standard: The MIDI phenomenon. *Computer Music Journal* 9, 4 (1985), 8-26.
19. Mansur, D.L., Blattner, M. and Joy, K. Sound-Graphs: A numerical data analysis method for the blind. *Journal of Medical Systems* 9 (1985), 163-174.
20. Myers, B.A. State of the art in user interface software tools. In *Advances in Human-Computer Interaction*, Hartson, H.R. and Hix, D. (Eds.), Ablex Publishing, 1991
21. Perrott, D., Sadralobadi, T., Saberi, K. and Strybel, T. Aurally aided visual search in the central visual field: Effects of visual load and visual enhancement of the target. *Human Factors* 33, 4 (1991), 389-400.
22. Portigal, S. *Auralization of document structure*. MSc. Thesis, The University of Guelph, Canada, 1994.
23. Reason, J. *Human Error*. Cambridge University Press, Cambridge, UK, 1990.
24. Reichman, R. Chapter 14: Communications paradigms for a window system. In *User-Centered System Design*, Norman, D.A. and Draper, S.W. (Eds.), Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986, 285-314.
25. Sumikawa, D., Blattner, M., Joy, K. and Greenberg, R. *Guidelines for the syntactic design of audio cues in computer interfaces*. Lawrence Livermore National Laboratory, 1986.