FaST Sliders: Integrating Marking Menus and the Adjustment of Continuous Values

Michael McGuffin[2], Nicolas Burtnyk[2], and Gordon Kurtenbach[1,2]

[1] Alias|wavefront
210 King Street East
Toronto, Ontario
Canada M5A 1J7

[2] Dept. of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 3G4

mjmcguff@cs.toronto.edu, n.burtnyk@toronto.edu, gordo@aw.sgi.com

Abstract

We propose a technique, called FaST Sliders, for selecting and adjusting continuous values using a fast, transient interaction much like pop-up menus. FaST Sliders combine marking menus and graphical sliders in a design that allows operation with quick ballistic movements for selection and coarse adjustment. Furthermore, additional controls can be displayed within the same interaction, for fine adjustments or other functions. We describe the design of FaST Sliders and a user study comparing FaST Sliders to other transient techniques. The results of our user study indicate that FaST Sliders hold potential. We observed that users found FaST Slider easy to learn and made use of and preferred its affordances for ballistic movement and additional controls. A sample program demonstrating our technique can be downloaded at http://www.dgp.toronto.edu/~mjmcguff/research/FaSTSlider/

Keywords: marking menus, control menus, flowmenus, gestures, sliders, fast slider, interaction design

Introduction

The adjustment of continuous values is a common transaction in many computer applications. Adjustment generally involves the setting of a value within a range of values with a certain degree of precision. For example, many GUI desktops use a graphical slider to control the computer's audio output level.

Many applications allow users to adjust numerous continuous values. Audio mixing applications, like physical audio mixing consoles, present users with a myriad of adjustable continuous values. Other applications with similar rich functionality like 3D modeling and animation applications may also make heavy use of continuous values (Figure 1).

Figure 1: Examples of 3D scenes containing objects where each object has many associated parameters. The top screen shot, of a UI for controlling facial expressions, shows how the sliders consume screen space if they are all displayed simultaneously. In the bottom sequence, a user invokes a marking menu over a duck object to select and adjust one of its parameters. The slider that appears can be dismissed after the adjustment is complete.

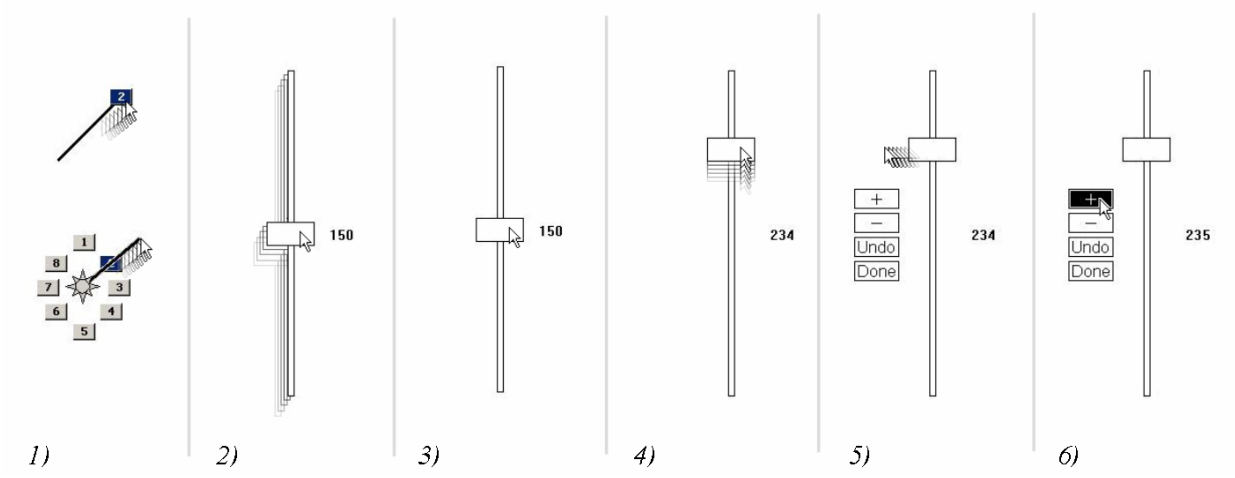

Figure 2: FaST Slider interaction 1) The user does a drag-release using either a menu or a quick "flick" gesture. This displays a slider. 2) With the mouse button released the entire slider follows the cursor. 3) When the mouse button is pressed the slider is "glued" to the screen. 4) Dragging adjusts the wiper (releasing at this point would dismiss the slider). 5) Dragging perpendicular to the slider posts the slider and some additional controls 6) these controls can then be used. Clicking on "Done" completes the interaction.

A popular approach to representing adjustable continuous values is the common graphical slider. Typically a graphical slider is presented to a user in a window, perhaps grouped with other related sliders. This arrangement can work well since it allows a user to see the relative settings of values and adjust them directly by dragging a slider's "wiper" with the cursor.

However, there are many situations in which it is not important to see the relative settings of sliders side-by-side. In these cases, a significant drawback of displaying multiple sliders at once is the consumption of screen space. As the top image in figure 1 shows, this can become an acute problem as the number of adjustable values grows. Another potential drawback of displaying multiple sliders grouped in a window can be the dissociation between sliders and the objects they control. While there has been much work on designing space saving small widgets to adjust values [3][7], another avenue of exploration is to make slider interaction transient to save space, as suggested by the bottom sequence in figure 1.

In addition to trying to save screen space, we are also interested in basing our transient slider design on some of the successful properties of marking menus [8]. Over the past six years we have gained extensive experience in deploying and using marking menus in commercial software made by Alias|wavefront. We have found that marking menus' property of "scale independence"—interpreting marks based on their shape and not their size—allows users to select very quickly and casually, and has become extremely popular with expert users. For example, some experts are so proficient with marking menus that they can perform an entire product demonstration without displaying a single menu. Thus our goal is to design a transient slider interaction technique that allows fast and casual operation consistent with marking menus.

The interaction technique we propose is a combination of marking menus and graphical sliders. Other interaction techniques have proposed similar combinations of radial menu techniques and dragging to control values [9][10][5]. These techniques use a single drag to perform both the selection of a value and its adjustment. Our technique differs in that we use two distinct drags to perform value selection and value adjustment. This decoupling produces some important differences in the resulting interaction.

In this paper we describe the design of our technique, the design principles it is based on, how it compares with other similar techniques, and users' reactions to the technique relative to other techniques. We conclude with a discussion of the overall merits of the technique.

FaST Sliders

We call our technique FaST Sliders, which stands for "Flick and Slide or Tweak". Our technique has three distinct steps: the first step ("Flick") is the selection of the value to be adjusted and the second step ("Slide") is the actual adjustment of the value. The third step ("Tweak") is optional and allows for additional kinds of adjustments of the value. The technique is a

combination of a marking menu and a one-dimensional graphical slider (Figure 2). It works as follows:

Flick Step – Selection of a value The user pops up a marking menu by holding down a mouse button. This menu contains menu-items which represent adjustable values. Like a regular marking menu, an item can be selected by moving into the radial area for the menu item to highlight it and releasing the mouse button. The menu is then popped down. Since the menu is a marking menu, selection can also be accomplished quickly without displaying the menu by performing a "flicking" drag movement.

Slide Step – Adjustment of the value Once selection is complete the system goes into "follow-mode". In this mode, a slider appears with its wiper directly under the cursor. If the cursor is moved, the entire slider follows the cursor, keeping the wiper located directly under the cursor. To adjust the slider value, the user presses down the mouse button and this "glues" the slider to the screen, allowing the user to drag the wiper to adjust the value. This drag is called "adjust-mode". When the mouse button is released the slider disappears and the interaction is complete.

Tweak Step – Additional controls If the mouse button is not released in adjust-mode, and the user drags the cursor off of the wiper by moving it perpendicular to the sliders' trough, this results in additional controls appearing (Figure 2.5). If the user releases the mouse button in this state (outside of the wiper), the wiper and additional controls stay posted. Now the user is free to move over and activate any of these controls as many times as needed. Once the user is satisfied with the adjustment they can dismiss the slider and the controls by clicking on the "Done" button. This ends the interaction.

Discussion of Design

In the design of FaST Sliders we have attempted to support several design properties. The first principle is to allow and exploit ballistic cursor motion. Ballistic motion is based on the concept of a *motor program* in motor control studies. A motor program is "a set of muscle movements structured before a movement begins, which allows the entire sequence to be carried out uninfluenced by peripheral feedback" [6]. Ultimately we want our technique to allow for ballistic movement and therefore require a minimum amount of user attention to system feedback.

Ballistic movement is supported in several ways. First, since we use a mouse up event to signal the selection step of the interaction, this is compatible with marking menus' "scale independence" property. Because of this, users can make marks of arbitrary size to select menu-items thus making selection fast and casual, generally a "flicking" ballistic movement. Second, because the entire slider follows the cursor after this ballistic flick, a user does not need to move the cursor over the wiper before starting to drag/adjust the value. This is especially important when ballistic mouse movements cause the cursor to move far away from the point where the mouse button was released. Essentially this design allows a user to "flick and slide"--using two very fast and casual mouse drags to display, select, adjust, and undisplay adjustable values.

Another important design property is what we call "distinct engagement"—that is, a distinct user gesture (mouse down) is used to engage the actual adjustment of the slider. We believe this may be an improvement over other techniques that use the less explicit event of dragging past some activation threshold (for example, past the outer edge of the radial menu). This property produces a subtle but important effect. Imagine a user is making an audio recording and wants to adjust a recording level slider. It is critical in this case that the slider be moved gently either up or down so that no "spikes" or "dips" occur in the recorded material. If an activition threshold is used, the user must be very attentive as to when they cross the threshold and once they cross the threshold their movement must be very controlled. Note that the difficultly of this situation increases as the user increases the speed of their initial selection movement. Thus this approach does not bode well with our goal of fast and casual movement.

However, the use of an explicit user trigger event eliminates this interference between selection and adjustment. Once a user releases the mouse button following a selection, they enter an interim mode (follow-mode) where they have a chance to stabilize their movements before engaging the adjustment of slider. This allows a quick ballastic selection movement followed by a controlled adjustment movement. One additional benefit of following-mode is that it also allows the user to reposition the cursor or input device before beginning to adjust the slider. This can be used to move into a more comfortable position or to position the slider away from or close to a particular part of the display.

Another important design principle is to allow in the design the incorporation of additional methods for modifying a value. For example, many applications have graphical sliders which, in addition to having a wiper for adjustment by dragging, may also have increment/decrement controls, numeric entry, default values, etc. Ultimately, a slider control could be a dialog box with a variety of common GUI controls.

Given this requirement, we allow additional controls to be accessed through the tweak step. It is important to note that once the user has dragged out of the wiper, the additional controls are posted and the mouse button can be released. This essentially leaves the user free to move to and click on any of these additional controls. Thus, in addition to the controls shown in our example of FaST Slider in Figure 2.5, any sort of dialog element could be available. Essentially, each invocation of a FaST Slider is capable of evolving into interaction with a full blown dialog box. Conceptually, this works out nicely since a UI designer has the option of encapsulating all of the controls associated with a value into a single "interaction location" in a user interface.

Comparison with Other Techniques

Two other interaction techniques that are similar to FaST Sliders have been proposed in previous work. Perhaps the most similar are Control Menus [9]. Control Menus, like FaST Sliders, use a marking menu to select the value to adjust. However, rather than using a mouse up event to signal the end of the selection step, Control Menus enter adjustment mode the moment the cursor is dragged beyond a fixed threshold distance from the center of the menu. Figure 3 shows an example of a Control Menu. As described earlier, we believe this approach can make ballistic motion difficult and carefully controlled engagement of adjustment difficult. However, this cost comes at the benefit of being able to perform the entire interaction in a single drag. Our user testing section discusses users' reactions to this cost/benefit trade-off.

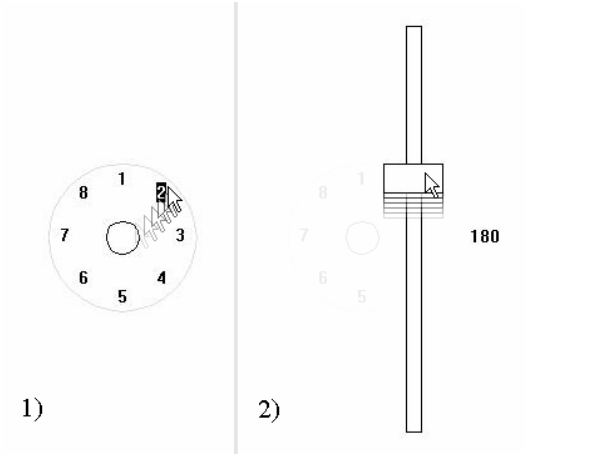

Figure 3: Using a Control Menus. 1) the user drags to select a value to adjust. When they drag past the edge of the menu, the selected slider is displayed. 2) Continuing to drag immediately adjusts the slider.

Another difference between FaST Sliders and Control Menus is that, with the latter, dragging is the only means available to adjust a value. FaST Sliders provide a method to escape dragging adjustment and access additional controls via its tweak-step. However, this benefit comes at the cost of limiting our current design of FaST Slider to only adjusting one-dimensional values. For example, Control Menus easily support two-dimensional panning, while dragging in the second dimension of a FaST Slider is used to post additional controls.

FlowMenus [4] are another radial menu based technique that is comparable to FaST Slider. FlowMenus, like FaST Sliders, are capable of supporting value selection, adjustment, and other controls. However, the interaction style to support this functionality has some significant differences. First, while FlowMenus use a radial menu layout like marking menus, item selection is performed by dragging into an item then back to the center of the menu. This "return to center" design allows a user to navigate through a hierarchy of menus without moving all over the screen. Second, continuous adjustment is supported by a special menu item, which affords adjustment by circular motion (Figure 4). Finally, like Control Menus, FlowMenus afford selection, adjustment, and other controls in a single drag.

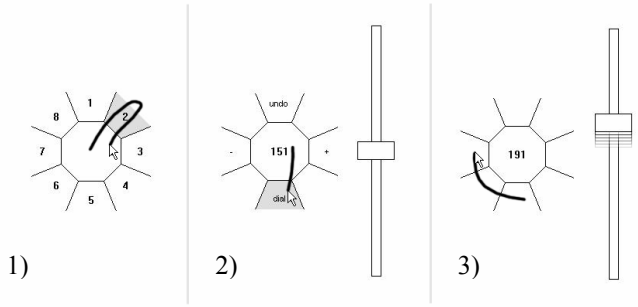

Figure 4: Example of using a FlowMenu to adjust a value. 1) a value is selected by dragging in and out of an item. This causes the display of a submenu with adjustment controls shown in 2) Moving into the dial menu item causes the display of a "rotary dial" shown in 3) where rotation adjusts the slider.

User Testing

To get a better understanding of the advantages and disadvantages of our FaST Slider design, we performed informal user tests. Specifically, we were interested in what effect the major design differences between FaST Sliders, Control Menus, and FlowMenus would have on users' impressions and performance with these techniques. Our intention was to use this information to further refine the FaST Slider design.

We implemented the three techniques such that each one could be used to adjust eight continuous parameters in a test program. The parameters ranged in value from 0 to 300. Since the study focused on adjustment of parameters rather than their selection, we used non-hierarchical menus for the Control Menu and the FaST Sliders (Figures 2 and 3). Similarly, the FlowMenu only had one menu level to select a parameter, followed by an additional level containing adjustment controls (Figure 4).

For all three techniques we attempted to provide the same level of functionality within the limits of the technique. For example, while fine-tuning functions could be provided in FaST Slider and FlowMenus, Control Menus have no obvious way of supporting these additional functions. Furthermore, FlowMenus as described in [4] have many additional ways to adjust a value. To be fair, we designed our own FlowMenu layout, which we thought would be effective but with functionality equivalent to the FaST Slider.

We tested 12 users, all of whom were familiar with Marking Menus. Most of them had 2-8 years of experience using complex 3D modeling software such as Alias|wavefront's Studio or Maya, and some were heavy users of Marking Menus, hotkeys, and other techniques for fast interaction.

To avoid confounding variables, no feedback was shown to the user other than that which the adjustment techniques provided; i.e., except for when a technique was engaged, the screen was blank. The tester controlled which technique was currently available to the user. A mouse was used for input, and the screen was 19" with a resolution of 1024x768.

Initially, users were told that each of the techniques would first present them with a menu for selecting one of eight parameters, and then allow them to adjust the selected parameter in some fashion. We then asked each user to try out and explore the techniques on their own, while talking aloud, to see if they could learn how to use them. The order of presentation of the techniques was permuted for each user. After about five minutes of exploration, if the user had still not completely understood the techniques, the tester explained how they work. (Note, however, that once this explanation was given, no further assistance or coaching was given during the tasks to follow.)

After the exploratory phase, the user was asked to perform a set of tasks using each of the techniques. The first task was *inspection*, where the user had to find out the current values of some of the parameters without changing them. The second was *extremal assignment*, where the user had to quickly set the parameters to their minimal or maximal value. The third was *rough assignment*, where the user had to assign a mid-way value of roughly 150 (± 10) to the parameters. Fourth was *exact assignment*, where the user was asked to assign a given value to various parameters. The fifth task was *fine-adjustment*, where the user was asked to increase or decrease a given parameter by 1 or 2 units.

Test Observations

Control Menus and FaST Slider were easy to learn. FlowMenu was more difficult.

We observed in the exploratory phase that all the users were able to learn how to roughly operate both the Control Menus and FaST Slider without coaching, although Control Menus' undo feature was only discovered by one user and went unnoticed by the others. The FlowMenu, on the other hand, generally required coaching. For example, one user commented that it "wasn't intuitive at all". Users seemed to be especially confused by the dial menu item, because, first, selection of all other items in the FlowMenu requires users to move out to the item and then *back* to the center, whereas dialing requires moving out and around, and, second, nothing in the FlowMenu's shape or feedback suggests rotary movement to the users.

This created many problems using the FlowMenu. For example, two users, intending to increment a value, moved over the increment menu item and released instead of moving back to the center to complete the selection. Even after users were given an explanation of how to operate the menu, mistakes were made. Some users often started to dial in the wrong direction and had to correct their motion. One user performed dialing not in the intended way, but by leaving the menu center in an arbitrary direction and traveling around until they accidentally hit the dial menu item. The user did this many times without noticing that anything was wrong.

Inspection and Fine-tuning were very difficult with Control Menus

The Control Menu was found to be almost impossible to use for inspection, since the user modified the parameter as soon as they traveled more than one pixel beyond the gray threshold circle. One user commented "that really bugs me […] to me that makes this [technique] unusable". However, two users discovered that by performing undo, they could successfully inspect values with the Control Menu without having to gingerly "just cross" the threshold circle. The Control Menu was also difficult to use for fine-adjustment, for similar reasons.

In terms of the other techniques, the FlowMenu easily supported inspection and fine-adjustment. The FaST Slider also allowed fast inspection, as the user only had to "flick" to go into "follow-mode" and see the value. The FaST Slider could then be dismissed simply by click-releasing over the wiper. Unfortunately, the mouse often moved one or two pixels between the click and release, causing the value to change. For fine-adjustment and exact assignment, a similar problem occurred if the user moved off of the wiper to post the additional controls for fine adjustment: as the cursor moved off, it slightly nudged the wiper up or down by an unintended amount. However, we believe simple modifications could correct this problem.

We observed a similar problem with the FlowMenu. Our implementation of FlowMenu allows the user to release immediately after dialing a new value. However, users sometimes tried to return to the center of the menu after dialing, presumably because they either assumed it was necessary (since all the other menu items require a return to center) or because they wanted to access the fine-tuning menu items. Unfortunately, because the return trajectory did not follow an exactly straight, radial line, the value was inadvertently changed along the way. It is not clear how to correct this problem, other than to avoid returning to the centre after dialing.

Users often ran out of screen space using the Control Menu.

Because Control Menus' slider is not movable once popped up, users often started their interaction close to the screen edge and ran out of screen space to complete a adjustment. To recover from this required re-invoking the menu at a different location. Note this is not a problem with FaST Slider, since the follow-mode allows a user to relocate the slider to a better screen location before adjusting. Many users commented positively about this feature, stating how they could use it to their advantage, for example, not only to avoid running out of screen space but to locate the slider as to not obscure their work.

In terms of rough and extremal assignment, apart from Control Menu's problem with screen space, all the techniques performed well.

Other observations

When asked which method they preferred, five chose the FaST Sliders, four users chose the FlowMenu, two were tied between these two techniques, and only one user preferred the Control Menu.

The users who preferred the FlowMenu stated that they enjoyed the challenge of learning how to use it. They described it as "cool", "smart" and "nice". Two users explicitly stated that they preferred it because it was "different". On the other hand, those who didn't prefer the FlowMenu described the circular dialing motion as a "little weird" or "unnatural", or that it "seems like a lot of extra work".

Users who preferred FaST Sliders reported they liked the ability to release and tweak, saying that FaST Sliders were the "safest" and "most stable" of the techniques, or that they were "familiar […] like a popup menu".

The sole user who preferred the Control Menu reported that it required the least work to adjust values.

Finally, one user was observed to make many selection errors with the Control Menu. For example, when the user wanted to quickly select a lower menu item and increase the corresponding parameter, he would stroke down in the appropriate direction (but not far enough) and then stroke up, unwittingly selecting and adjusting the parameter for an upper menu item. This lends support for our belief that scale-invariant interpretation of Marking Menu strokes is important.

Discussion & Conclusions

Our user study gives evidence to the effectiveness of some of our original design principles. In terms of ballistic motion and distinct engagement, we believe that part of the success of FaST Slider is due to careful attention to these design aspects. Evidence for this is given by the problems observed when using Control Menus to perform inspect and fine adjustment. An additional benefit of designing for ballistic motion is the "follow-mode" design feature in FaST Slider. Not only did this allow users to inspect values safely, but also allowed for repositioning of the slider to the user's advantage. In our study, users repositioned the slider to avoid running out of screen space to complete a drag. Users also acknowledged the potential usefulness of repositioning the slider to avoid obscuring their work, e.g. a 3D model, or to place it relative to objects in a scene, e.g. using it as a ruler.

Our other design principle of supporting additional controls proved effective. The additional controls present in FaST Slider and the FlowMenu were used in our study, and the lack of these proved to be a weakness for Control Menu. Furthermore, by allowing users to post the additional controls of the FaST Slider, we open the design to potentially support the complete range of standard controls available in a dialog box. In contrast, it is not clear how to integrate standard controls in the FlowMenu's design. Furthermore, it is important to note that because FaST Sliders can be posted, our

design could allow for multiple sliders to be displayed simultaneously, behaving as modeless dialog boxes.

Despite FlowMenus and FaST Sliders scoring very closely in terms of preference in our user study, we believe a UI designer may prefer FaST Sliders since FlowMenus are a much more unconventional style of interaction and therefore require additional user learning and may not appear to be consistent with other standard GUI techniques.

Future Work

Our current design of FaST Slider deals only with one-dimensional dragging. Extending the technique to two-dimensional adjustment, while still retaining the tweak step, is still an unsolved problem.

Based on some user's preference for the circular dragging in FlowMenus, replacing FaST Slider's linear slider with a rotary slider is an interesting future research topic. One useful property of rotary adjustment is that users can continuously control the C:D ratio by varying the radial distance of the cursor to the dial's center. Furthermore, rotary motion naturally lends itself to relative adjustment.

While the focus of this paper has been on the control of numerical values, future research could be done to adapt FaST Slider to adjust non-numerical data using techniques such as the AlphaSlider [1].

Although our design does not adhere to the concept of physical tension proposed in [2], requiring more than one drag for an interaction phrase did not seem to cause a problem. We believe this is due to the "follow-mode" providing sufficient visual feedback to keep users aware of the current state. It may be that this type of feedback (the entire slider following the cursor) provides a kind of "visual tension" analogous to physical tension. The issue of physical versus visual tension in user interface design bears further investigation.

Our work has not compared the relative speed of operation of each technique. Because FaST Sliders require at least two drags for an interaction, a naïve keystroke model analysis suggests that our design should be slower compared to the other techniques. However, we suspect that the improved support for ballistic motion designed into FaST Sliders may more than offset the penalty incurred by an extra button press. A formal study is required to test this hypothesis, and could contribute to a deeper understanding of the role of gesture design in supporting high quality interaction.

Acknowledgements

We thank George Fitzmaurice, Ravin Balakrishnan, Azam Khan, and Scott Guy for design comments and suggestions. We also thank the participants in our user study.

References

1. Ahlberg, C., and Shneiderman, B., (1994) The Alphaslider: A Compact and Rapid Selector. *Proceedings of CHI '94*, ACM Press, 365-371
2. Buxton, W. (1986). Chunking and phrasing and the design of human-computer dialogues. In Kugler, H. J. (Ed.) *Information Processing '86, Proceedings of the IFIP 10th World Computer Congress*, 475-480, Amsterdam: North Holland Publishers.
3. Buxton, W., Reeves, W., Fedorkow, G., Smith, K. C., & Baecker, R. (1980). A Microcomputer-Based Conducting System. *Computer Music Journal* 4(1), 8-21.
4. Guimbretiere, F. & Winograd, T. (2000) FlowMenu: Combining Command, Text, and Data Entry. *Proceedings of UIST 2000*, ACM, 213-216
5. Hopkins, D. (1991) The design and implementation of pie menus. *Dr. Dobb's Journal*, 16(12), 16-26.
6. Keele, S. W. (1968) Movement control in skilled motor performance, *Psychological Bulletin*, 70, 387-403.
7. Kurtenbach, G. (1988) Hierarchical Encapsulation and Connection in a Graphical User Interface: a Music Case Study, MSc thesis, University of Toronto
8. Kurtenbach, G. & Buxton, W. (1993) The limits of expert performance using hierarchical marking menus. *Proceedings of the CHI '93 Conference on Human Factors in Computing Systems,* New York: ACM., pp. 482-487
9. Pook, S., Lecolinet, E., Vaysseix, G., and Barillot, E. (2000) Control Menus: Execution and Control in a Single Interactor. *CHI 2000 Extended Abstracts*, ACM, pp. 263-264.
10. Reinhardt, A. (1991). First Impression: Momenta Point to the Future. *Byte Magazine*.