

The Design of 3D Haptic Widgets

Timothy Miller*

Robert Zeleznik†

Brown University‡

Abstract

3D interaction with a computer is hard. As human interaction in the real world is quite dependent on haptics, we believe that adding haptics will alleviate many of the problems of 3D interaction with a computer. The naive approach to incorporating haptics would be to simply find existing objects and imitate them. This, however, would not seem to adequately leverage what is possible with a reasonably general force feedback device, as it does not take advantage of the ability to provide forces that are not physically based, or to simulate objects and behavior that could have been physically realized but weren't, or which would have been prohibitively difficult to physically realize. From our extensive experience with designing haptic computer user interfaces and using haptic interfaces in the real world we have abstracted from those interactions certain qualities which we present in this paper. In addition, this paper demonstrates how to use these qualities in the analysis and design of a suite of 3D haptic widgets, most of which are novel and many of which include non-physically based forces.

CR Categories: H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces—Haptic I/O; H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces—Graphical user interfaces (GUI); H.5.1 [Information Interfaces and Presentation (e.g., HCI)]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality

Keywords: 3D Haptic Widgets, 3D Widgets, Widget Taxonomy

1 INTRODUCTION

Interaction with the computer in three dimensions is difficult [6]. As humans depend to a great extent on haptics and the presence of and ability to sense forces in the real world, we expect that adding force-feedback to interfaces will help in the virtual world. The field of ergonomics has described devices in the real world in terms of a set of low-level physical characteristics and also provided recommendations for the use of those characteristics in a given device for

a given task [14, pp. 135–142] [21, pp. 270–272, 641–645]. An analogous set of qualities from animation has been applied to user interfaces [4]. This paper presents an attempt at similarly producing a set of haptic principles that both aid in the evaluation of existing haptic interfaces and also guide further development. However, the set we use is different than any we have seen from the field of ergonomics, perhaps because current force feedback devices are both more limited and more readily flexible within those limitations than easily available and more typical physical controls.

In much current haptic research, what to simulate is given, and the problem is how to simulate it. For instance, in basic teleoperation such as in the nanomanipulator [7], the problem is to simulate the surface and forces reported by a scanning probe microscope, at the same time as controlling the forces output to the probe. As another example, in surgical simulation applications [15] the problem is to simulate the forces that would actually be felt by the surgeon interacting with the simulated tissue. In contrast, for our applications, a task to be performed by the user is given, and we want to choose what to simulate to allow and aid performing that task. Although there has been a scattering of previous work along those lines [1, 9, 12, 13, 16–18, 20], little has been done to find an overall characterization of the sorts of general qualities that using haptics can provide. In addition, previous haptics work in three dimensions [1, 9] appears to address this problem by finding a physical analogy for the task and simulating it (such as simulating a physical button, or a physical sculpting tool). Our approach additionally allows for the combination of physically-based and possibly non-physically-based behavior to produce overall interaction that is not necessarily explicitly based on the real world and may not even be easily achieved with typical real-world devices. Analysis of the resulting behavior and interaction in terms of the system of qualities we present may then lead to better designs and directions for improvement for particular tasks.

Other researchers have approached 3D interaction by developing elaborate 3D widgets [6] making use only of visual feedback. In contrast, our approach uses force feedback as well, and focuses on simpler mechanisms that seamlessly integrate with the environment, rather than on more complex contraptions that the user must consciously manipulate. Previous work has also employed purely graphical techniques such as snapping [2, 3]; these techniques have the problem that they tear holes in the user's input space because it is not possible to specify points near to but not on the snapping point. We believe haptics can address this problem and other similar ones, but this paper is not about why haptics is better; rather, it presents principles for how to use haptics effectively. Therefore we do not directly compare graphical techniques with haptical techniques in this paper.

Force feedback for the widgets described in this paper was implemented using a 1.0-workspace PHANTOM™ [9] with encoder gimbal, made by SensAble Technologies, Inc.; this device has a 6 degree-of-freedom (DOF) input channel for position and orientation of the stylus grip and provides 3 DOF force output.

*tsm@cs.brown.edu

†bcz@cs.brown.edu

‡Department of Computer Science, Box 1910, Brown University, Providence, RI 02912.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1999 Symposium on Interactive 3D Graphics Atlanta GAUSA
Copyright ACM 1999 1-58113-082-1/99/04...\$5.00

2 PRINCIPLES FOR USING HAPTIC FEEDBACK

We identify four uses of haptic feedback, *anticipation*, *follow-through*, *indication*, and *guidance*, as well as a specific usage of anticipation and guidance (for *distinguishing directions*) which is not quite a widget but seems enough of a building block on its own to single out. As it has turned out, the first two of our uses are broadly similar enough to principles from animation [4] that we have adopted the same names for them. Our taxonomy of haptic feedback uses is only meant as a rough guideline, and it is likely that further refinement will be necessary.

2.1 Anticipation

Haptics can be used to provide a breakable¹ force resisting the user's motion and indicating the imminence of a qualitative change in the user's input before the user actually makes it, so that the user has the opportunity to back off from that change if it is undesired. For instance, in pressing a typical button, the user feels a force back indicating that they are pressing the button and that if they press harder they will trigger whatever the button controls. As a perhaps more illustrative example, consider a volume knob with a detented off position. The user feels little resistance to motion through most of the range of the control, where the volume is being adjusted quantitatively, but resistance increases upon nearing the end with the off position, letting the user know that they are nearing a point where the behavior of the control changes qualitatively.

2.2 Follow-through

Haptics can also be used to let the user know that an attempted qualitative change has actually been accomplished, so that they have the opportunity to correct their motion if they don't get this feedback. For instance, consider the volume control knob with detented off position mentioned above: the sudden decrease (and possibly even direction reversal) of the resistance force on entering the off position makes the knob feel as though it has snapped into place, giving positive feedback to the user that the control is indeed off.

2.3 Indication

Haptics can provide an indication that a continuing qualitative condition remains in effect, possibly with quantitative information about the condition. For instance, a joystick with springs to return it to the center position lets the user know that the control is away from the neutral position and also in what direction it's away, and possibly how far.

2.4 Guidance

Haptics can be used to adapt the user's input with a bias towards some set of possible inputs. For instance, a straightedge is an example of guidance because it can be used to guide the user towards those points along a straight line at a particular position. A straightedge may be regarded as a special case of a *jig*, a device for guiding a tool along a specific path or to specific points, such as for drilling evenly spaced holes. As another example, consider a hypothetical radio station tuning dial with detents at user-settable positions (corresponding to favorite stations, for instance). A more subtle example is a control which uses viscous resistance to guide the user to an appropriate rate of change for the control [21, p. 271].

¹Breakable here means that the force can be overcome by the user to "break through" it.

2.5 Distinguishing Directions

Haptics can be used to allow the user to make clear distinctions between locally orthogonal directions. This can be used to map different (but possibly related) controls onto different dimensions of the same input mechanism. One example from the real world is a flight yoke which allows rotation of a control about a shaft and translation of the control along the shaft. The forces resisting the translation are noticeably greater than those resisting the rotation, thus allowing the user to only rotate the control without translating it, for instance, while still being able to fluidly do both. This can be considered a use of anticipation, because the force feedback lets the user know when their input will qualitatively change from rotation to translation. However, it could also be considered an instance of guidance, because the user's motion is being adapted with a bias towards those motions which only control one dimension at a time.

3 3D HAPTIC WIDGETS

We would like to design widgets using these principles in an orthogonal manner. However, the principles must be embodied in terms of forces, and a given force may have additional side-effects or express multiple principles depending on how it is used. For instance, if the user is attempting to very precisely position an object, having follow-through to indicate that the user has indeed started positioning it may jerk the user, reducing accuracy. As another example, consider a groove running along a surface: the same forces that adapt the user's position toward the bottom of the groove can also be viewed as anticipating the qualitative change of leaving the line the user is being guided to. Although we do not have a general technique for handling this problem, our examples of widgets illustrate some of the means that may be used to overcome it.

We discuss both widgets that are newly presented in this paper and also a few widgets that have been previously published; previously published widgets are identified explicitly with citations in the description for each such widget. Each of the widgets is analyzed in terms of our haptic feedback use taxonomy. Our haptic explorations have been performed using a 3D polygonal modeling system as a testbed; this system is designed to use a sketch-oriented interface loosely inspired by the principles of Sketch [23]. All of the widgets newly presented in this paper were implemented in the process of trying to support this task of polygonal modeling.

3.1 Pushbuttons

Virtual haptic pushbuttons have been implemented previously by a number of researchers [1,9,22]. A basic pushbutton force profile [9] consists of an initial springy region where the force increases linearly with displacement, followed by a sudden decrease in resistive force and transition to a *deadband* where the resistive force is constant, followed by a *hard stop* where the force approximates that of a hard surface (usually with a simulated damped spring force). Anticipation is provided by the initial springy region, which lets the user know that they are about to trigger a button and allows them to back off if they don't mean to. Potential for haptic follow-through is actually provided twice, at separate points, in this example: on the sudden popping through the resistive force into the deadband, and on running into the hard stop at the end. Which of these corresponds to the actual triggering of the button, however, is ambiguous, both from the feel of the widget and from the specification of the button. The authors have even encountered this ambiguity in real-world pushbuttons: some trigger when the initial springy region is popped through, while others trigger when the hard stop is reached. Some unfortunate real-world buttons even trigger partway through the deadband region (and can be awkward to use as a result). It seems possible that the design of even something as simple



Figure 1: Cross-sectional geometry of our notches and dimples.



Figure 2: Approximate cross-sectional geometry of our proposed improved notches and dimples.

as a pushbutton could be improved by making the follow-through unambiguous, perhaps by replacing the hard stop with graded increasing resistances throughout the deadband region that slows the user down to a rough stop without any particular transition point. On the other hand, one possibility is to have the triggering happen when the user hits the hard stop, but only if the hard stop has not already been hit without an intervening retreat beyond the pop-through point. This would ensure hysteresis in the button (which probably has to be present anyway in some form to make it a useful control) and provide feedback at both ends of the hysteresis.

3.2 Notches and Dimples

In our test system, most objects are created by sketching strokes that gesturally denote the object and its position; for example, a cube may be created by sketching three perpendicular edges. Most of these strokes are made on existing surfaces in the environment (as in the Sketch system [23] where new objects are always created resting on old ones). *Notches* are made in the simulated haptic surface (but not in the displayed version on screen) along lines of interest: parallel to the axes of the object being drawn on, and perpendicular and parallel to strokes drawn already. These notches have a V cross-section with the ends of the V flush with the surface, as shown in Figure 1. Cone-shaped *dimples* are additionally made into the surface at points of interest: the endpoints of existing strokes, points where the current stroke would be equal in length to an existing stroke, and so forth. Currently the notches and dimples all slope inward at 45°; the notches are currently all 1 mm deep and the dimples all 1.5 mm deep so that they can be felt even when they’re on an existing notch. The intent is to extend the use of these notches and dimples (along with certain widgets described later) to implement a haptic version of snap-dragging [3], and for use in constraint based modeling systems in general.

The analysis of notches and dimples as used in our system in terms of our taxonomy of haptic principles is almost identical, so only notches will be discussed here. The primary haptic principle notches provide is guidance, as the lines they guide the user to only differ quantitatively from other positions on the surface and the user is free to move to any other position. Notches also provide follow-through, as there is a distinct feel when the user has hit the exact bottom of the notch and is therefore exactly where they have been guided to.² Notches also provide undesired follow-through on entering or leaving the outer part of the notch because of the sharp corner to the join of the V with the surrounding surface; since this false follow-through doesn’t correspond with anything in particular in the application, it would be desirable to eliminate it by making the notches have a more rounded falloff to the surrounding surface, as shown in Figure 2.

²It might be objected that our definition of follow-through said it only applied to qualitative change, but we have argued that the use of notches is for a quantitative change; in fact the notch guides to a quantitatively different position, but whether the user is accepting the guidance or not is considered a qualitative difference.

3.3 Line Detents

In the process of sketching an object which isn’t flat, the user needs to draw a line which doesn’t lie on the surface they started on. In order to continue to provide guidance during the drawing of this line, we introduce the notion of a *line detent*. Whereas normal detents guide a control to specific points along a dial, say, we consider a generalization of that idea to be any roughly similarly-feeling snapping guidance to a point, curve, or surface. This includes both the notches and dimples above, as well as this widget, which has a force profile much like a typical button without a hard stop, only going out radially from a line. That is, exactly on the line defining the position of the widget, the user feels no force. Moving slightly away from the line, the user feels a force increasing linearly with distance from the line and directed towards the line, providing anticipation that the user is about to leave the line. Beyond a certain threshold distance, the force suddenly drops to a lower level (follow-through that the user has indeed left the line) and remains at that strength (but always directed towards the line, providing guidance back to the detent) as long as the user is far enough away from the line that the force that would be produced by the springy region is greater (thus providing hysteresis). Currently, the spring constant is set to 1 N/mm and the threshold to 1 mm, with the force dropping outside the threshold to half its maximum, and the line detents are placed perpendicularly to the surface being drawn off from when the user lifts off that surface.

3.4 Infinitesimal Ridges

All previous haptic rendering systems we have seen render polyhedra such that the user cannot get exactly to the edge of a face without falling off. However, users want to be able to have alignment between their objects. Other systems [2, 3] use graphical snapping to aid this task. In order to facilitate this precise alignment in our system, infinitesimal ridges were added to all the edges of objects simulated in our system. These ridges, implemented similarly to those used in [11, 12], simply prevent the user from falling off the edge of the object as long as the user is touching the object, but offer no resistance if the user is off the surface of the object, by however small an amount. Thus, it feels to the user as though there are little ridges at the edges of the objects; the user can easily touch a face of an object and zip right to a corner, knowing that the ridges will stop them before falling off and guide them right to the spot.

3.5 Haptic Virtual Sphere

Anderson [1] implemented a haptic sphere for controlling rotation and scaling. In the use of this “sphere craft”, the user’s position is generally constrained to move smoothly on the surface of a sphere, but the user can enlarge or shrink the sphere by exerting sufficient force in or out. Movement along the surface of the sphere produces a corresponding rotation of the virtual camera, with movement in and out controlling zoom. The primary principle evident here is distinguishing directions, as discussed above, but the continuing sphere constraint also serves as indication that the user is still in the sphere craft manipulation mode and tells something about where the user is relative to the center of the sphere.

We have also tried several versions of a haptic virtual sphere for camera and object control. The primary differences between ours and Anderson’s are that ours appears in the environment, in registration with the thing to be manipulated, and that we have tried a variety of additional control mappings, some of which are described in the next section. Our implementation maintains an effective user position on the sphere, similar to the idea variously named “God Object”, “ghost point”, “proxy”, and “surface contact point” in the literature [9, 19, 22]. A spring force is applied to bring the user’s

true position toward the effective user position, proportional to the distance between them; the spring constant, however, is larger in the direction radially away from the center of the sphere than in the directions tangential to the sphere. The effective user position moves so that it is never further than a certain threshold from the true user position; the threshold value is again larger in the radial direction than in the tangential, and the radial and tangential displacements are computed separately so that the new effective position remains at the same radius if the user hasn't moved far enough radially, for instance. This is similar to the implementation of friction in [20]. Currently, the values for the radial and tangential spring constant and threshold are, respectively, 0.7 N/mm, 2 mm, 0.2 N/mm, and 0.5 mm.

3.6 Sphere With a Twist

The biggest problem with using the virtual sphere to control orientation is the mismatch between the 2 DOFs of the sphere surface and the 3 DOFs of rotation, resulting in an inability to directly control twist around the axis passing through the user's position. People work around this problem by using looping curves to twist [5], but, since the PHANTOM has other DOFs, it seems reasonable to try to use them. Thus, as an alternative to the control mapping for the resizable sphere described by Anderson, we have another version that maps radial displacement of the stylus tip to twist around the axis through the center of the sphere and the user's position. (It is worthwhile to note that, in contrast with the mapping of radial displacement to scaling, this mapping does not result in the user's position maintaining correlation with a position in the world if the widget is used to control the camera.) This does indeed solve the problem of being able to control twist, but the result does not feel very intuitive to us.

We implemented another variant of the widget where the twist is controlled by that part of the rotation of the stylus which is around the axis v from the sphere center through the user's position. (We convert the orientation change of the stylus into a vector u whose direction is the axis of rotation and whose length denotes magnitude of the rotation, finding the length of its projection onto the radial axis v , i.e. $u \cdot v$, and twisting the sphere by that amount.) Two special cases will be given to illustrate this extension to the method; for ease of exposition, in all of these cases the tip of the stylus will be taken to remain fixed at the front of the virtual sphere (closest to the viewpoint). In the first case, the body of the stylus points straight out from the sphere, along v and thus towards the viewer. If the stylus is twisted clockwise, say, around v , the sphere will twist by the same amount (Figure 3(a)). If, instead, the stylus body is rotated so that it points straight left, at 9 o'clock, with no twist, then the virtual sphere will not move (Figure 3(b)). In the second case, the body of the stylus points straight up, at 12 o'clock. If the stylus is twisted around its body in this position, without moving from the 12 o'clock position, the virtual sphere will not move (Figure 3(c)). If instead the stylus body is rotated counterclockwise by 90° so that it now points to 9 o'clock, the virtual sphere will rotate with it, also by 90° (Figure 3(d)). This technique came about because one author of this paper noted that he felt like he was intuitively trying to twist the regular haptic sphere by rotating the stylus in his hand in sympathy with the desired twist. Although the result does not take further advantage of force feedback principles beyond the basic idea of distinguishing motions, one user does feel that it provides a significant and intuitive advantage over either of the other mechanisms. However, opinion is divided among the few other people who have tried it, and more testing is necessary to find out whether it offers any general advantage over the basic technique.

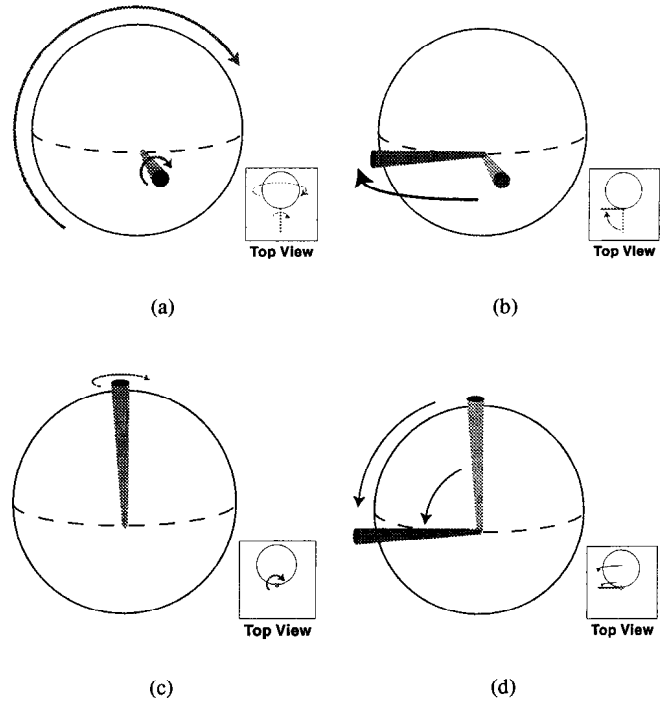


Figure 3: Examples of the virtual sphere with a twist.

3.7 Press to Slide

Normally, moving the stylus across the surface of a virtual object results in simply feeling the object. In our system the user can in addition press harder into the surface of the object to slide it parallel to that surface. This is similar to what happens in the real world, where one can feel across the top surface of an object on a desk, or press harder to slide it across the desk, except that sliding can be done in any plane in our system and does not require having an object underneath. The implementation currently provides no haptic feedback whatsoever in addition to the basic surface forces, and therefore does not make use of any of our principles. It is likely that they could be made use of fruitfully here, providing anticipation through two different spring constants applied for different user force levels, perhaps follow-through via a popping sensation when the dragging mode is engaged, and indication through a difference to the sliding feel, perhaps by using different friction coefficients, or viscosity instead of friction. In addition, a future extension is planned to allow the user to click to drag the object perpendicularly to the constraint plane, applying the distinguishing of directions principle to make the resistance in the perpendicular direction different.

3.8 Viscosity-Driven Switches

Our previous research has considered the idea of using flicking motions of the 2D cursor to switch application modes, similar to marking menus [8] without having to click. A drawback of this approach is that the mode switches are often triggered accidentally; we address this issue by attempting to use the haptic principle of anticipation to let the user back off from the mode switch if it is undesired, along with corresponding follow-through. This was done by adding an unusual non-physical variant of viscosity to the motion of the stylus in free space. At speeds below 200 mm/s, no viscous force is added. At greater speeds, a force is added oppo-

site to the direction of movement, and proportional to the square of the amount the speed is above 200 mm/s, with a proportionality constant currently $20 \text{ N s}^2/\text{m}^2$. This provides the desired anticipation. Above 600 mm/s the viscous force suddenly drops off to nothing again, providing follow-through. Hysteresis is provided by not starting the viscous force again until the speed has dropped below 500 mm/s. A quadratic increase in resistance was chosen because attempts made with linear increases resulted in either there being too little anticipation or else having the user's motion slowed unacceptably during normal movements (or else having the speed required to trigger the switch be too high to achieve comfortably).

3.9 Plastic Deformation Vertices

Occasionally in a polygonal modeler, it is desirable to manipulate vertices freely in space. Simply mapping the position of the haptic device to the position of the vertex results in an undesirable jittering due to shakiness in the human hand position. Our solution is to have the user's position virtually attached to the vertex position by a spring; when the user has moved further than a certain threshold away from the vertex the vertex moves to be exactly that threshold distance away from the user, similarly to the resizing behavior of the haptic virtual sphere above, and to the implementation of friction in [20]. This provides both guidance because it adapts the user's input to be biased towards the current position, and anticipation of when the user is about to make a change to the position of the vertex. At a threshold of 1 mm and spring constant of 0.4 N/mm this feels as though one is smoothly deforming a plastic object, and the feeling of control over the position is better than without.

3.10 Failed Widgets

3.10.1 Touch to Draw

In our current system, strokes are initiated by clicking the physical button on the PHANTOM stylus when the user is touching a surface. Our original method of starting strokes was simply to touch the surface to begin drawing, analogous to drawing with a pencil or pen in the real world. However, we were frustrated with the difficulty in drawing with this method compared with drawing strokes with a real pencil, and found that the click-to-draw method was noticeably easier and faster to use (though still not as good as a pencil). From our observation, it seems that one of the most noticeable sources of difficulty with touch-to-draw was that it was difficult to anticipate when one was about to touch the surface and therefore begin drawing. We hypothesize that in the real world there are additional visual cues such as stereo vision which would give a better sense of where the user is relative to the drawing surface and thus provide visual anticipation of when they are about to begin drawing.

3.10.2 Break Detent to End

Another early idea that didn't work out was, in fact, one of our original ideas for applying haptic anticipation: something like our line detent would be provided when the user started drawing a stroke, and the end of the stroke could be indicated by merely breaking through the detent along the surface, also signaling the beginning of the next stroke. (This was originally designed in the context of the gesture to create a cube, which requires three perpendicular strokes.) This would seem to provide exactly what we've talked about for anticipation: a resistive force that lets the user know when they're about to make a qualitative change in their gesture from continuing one stroke to starting another. Unfortunately, it resulted in greater inaccuracies of drawing and a lessened feeling of control than ending the stroke by either releasing the stylus button or lifting

off the surface. We were unable to find a level of force for the detent anticipation which provided sufficient force that the user could indeed feel the anticipation but not too much force that the user couldn't break through it without disturbing their endpoint. This is therefore a case where there is a tradeoff between anticipation and accuracy.

4 CONCLUSIONS AND FUTURE WORK

We have presented a small number of principles for using haptic feedback and shown how they apply to existing haptic widgets, as well as haptic widgets presented here for the first time. We have found these principles to be surprisingly robust and general in the course of our discussions while developing these widgets; in fact, as it seems that most if not all of our principles can be applied to purely graphical user interfaces as well, one interesting avenue for future work is to try to improve the design of such interfaces using these principles to see how far the principles can be extended and whether new ones become evident.

Most examples of guidance we have encountered involve some degree of anticipation as well. One future widget we would like to explore is a *line wiggler*. It can be difficult to draw a curve with a uniform amount of wiggling in it, controlling both the overall shape of the curve and perhaps occasional variations in the wiggles. Applying a sinusoidal force to the user's position while they are drawing a curve could aid in this task, letting the user focus on broad motions to specify the overall shape, but still allowing the user to resist the forces to produce occasional variations. This would be an example of guidance with no aspect of anticipation at all.

Other possible future widgets include a haptic version of the deformation rack [6], presumably by using distinguishing directions to simultaneously control multiple less-than-three-dimensional parameters, such as bend and taper. Yet another possible widget is the use of textures in free space to indicate various modes the interface may be in.

5 ACKNOWLEDGMENTS

Thanks to John Hughes for his help with forming the exposition of Section 3.6. This work is supported in part by the NSF Graphics and Visualization Center, Alias/Wavefront, Advanced Networks and Services, Autodesk, Microsoft, Sun Microsystems, Silicon Graphics, Inc., and TACO. Our system was implemented using some of the geometrical manipulation code from EusLisp [10] ported to Common Lisp.

References

- [1] Tom Anderson. FLIGHT: A 3D Human-Computer Interface and Application Development Environment. In J. K. Salisbury and M. A. Srinivasan, editors, *The Proceedings of the Second PHANTOM Users Group Workshop*. December 1997. Published as MIT Artificial Intelligence Laboratory Technical Report AITR-1617 and Research Laboratory for Electronics Technical Report No. 618.
- [2] Ashlar. Vellum. Computer Program. <http://www.ashlar.com>.
- [3] Eric A. Bier. Snap-Dragging in Three Dimensions. In *1990 Symposium on Interactive 3D Graphics*, pages 131–138. ACM SIGGRAPH, 1990.
- [4] Bay-Wei Chang and David Ungar. Animation: From Cartoons to the User Interface. In *Sixth Annual Symposium on*

- User Interface Software and Technology*, pages 45–55. ACM, November 1993.
- [5] Michael Chen, S. Joy Mountford, and Abigail Sellen. A Study in Interactive 3-D Rotation Using 2-D Control Devices. In *Computer Graphics (SIGGRAPH '88 Conference Proceedings)*, pages 121–129. ACM SIGGRAPH, August 1988.
 - [6] Brookshire D. Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-Dimensional Widgets. In *1992 Symposium on Interactive 3D Graphics*, pages 183–188, 230–231. ACM SIGGRAPH, 1992.
 - [7] Mark Finch, Mike Falvo, Vernon L. Chi, Sean Washburn, Russell M. Taylor II, and Richard Superfine. Surface Modification Tools in a Virtual Environment Interface to a Scanning Probe Microscope. In *1995 Symposium on Interactive 3D Graphics*, pages 13–18, 203. ACM SIGGRAPH, April 1995.
 - [8] Gordon Kurtenbach and William Buxton. User Learning and Performance with Marking Menus. In *Proceedings of ACM CHI '94 Conference on Human Factors in Computing Systems*, pages 258–264. 1994.
 - [9] Thomas Harold Massie. *Initial Haptic Explorations with the Phantom: Virtual Touch Through Pointer Interaction*. Master's thesis, Massachusetts Institute of Technology, February 1996.
 - [10] Toshihiro Matsui and Isao Hara. EusLisp version 8.00 Reference Manual. Technical Report ETL-TR-95-2, Electrotechnical Laboratory, Agency of Industrial Science and Technology, Ministry of International Trading and Industry, 1-1-4 Umezono, Tsukuba-city, Ibaraki 305, Japan, January 1995. Available from <http://www.etl.go.jp/~matsui/eus/euslisp.html>.
 - [11] Timothy Miller. Implementation Issues in Adding Force Feedback to the X Desktop. In *The Proceedings of the Third PHANTOM Users Group Workshop*. 1998.
 - [12] Timothy Miller and Robert Zeleznik. An Insidious Haptic Invasion: Adding Force Feedback to the X Desktop. In *User Interface Software and Technology*. ACM, November 1998.
 - [13] Stefan Münch and Martin Stangenberg. Intelligent Control for Haptic Displays. *Computer Graphics Forum*, 15(3):C217–C226, September 1996. Proceedings of EUROGRAPH-ICS'96.
 - [14] David J. Osborne. *Ergonomics at Work*. John Wiley & Sons, New York, second edition, 1987.
 - [15] R. O'Toole, R. Playter, W. Blank, N. Cornelius, W. Roberts, and M. Raibert. A Novel Virtual Reality Surgical Trainer with Force Feedback: Surgeon vs. Medical Student Performance. In J. K. Salisbury and M. A. Srinivasan, editors, *The Proceedings of the Second PHANTOM Users Group Workshop*. December 1997.
 - [16] Louis Rosenberg and Scott Brave. Using Force Feedback to Enhance Human Performance in Graphical User Interfaces. In *CHI 96*, pages 291–292. ACM SIGCHI, April 1996.
 - [17] Louis B. Rosenberg. FEELit Mouse: Adding a realistic sense of FEEL to the Computing Experience. <http://www.force-feedback.com/feelit/white-paper.html>, October 1997.
 - [18] Louis B. Rosenberg and Scott Brave. The use of force feedback to enhance graphical user interfaces. In Mark T. Bolas, Scott S. Fisher, and John O. Merritt, editors, *Stereoscopic Displays and Virtual Reality Systems III*, pages 243–248. 1996. Proc SPIE 2653.
 - [19] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. The Haptic Display of Complex Graphical Environments. In *Computer Graphics (SIGGRAPH 97 Conference Proceedings)*, pages 345–352. ACM SIGGRAPH, 1997.
 - [20] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Ziles. Haptic Rendering: Programming Touch Interaction With Virtual Objects. In *1995 Symposium on Interactive 3D Graphics*, pages 123–130. ACM SIGGRAPH, 1995.
 - [21] Mark S. Sanders and Ernest J. McCormick. *Human Factors in Engineering and Design*. McGraw-Hill, New York, sixth edition, 1987.
 - [22] SensAble Technologies, Inc. GHOST™ Software Developer's Toolkit Programmer's Guide.
 - [23] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: An Interface for Sketching 3D Scenes. In *Computer Graphics (SIGGRAPH 96 Conference Proceedings)*, pages 163–170. ACM SIGGRAPH, August 1996.