Haptic State-Surface Interactions

Rick Komerska and Colin Ware
Data Visualization Research Lab
Center for Coastal and Ocean Mapping
University of New Hampshire
Durham, NH 03824
komerska@ccom.unh.edu
colinw@cisunix.unh.edu

Abstract

Haptic devices, such as the PHANTOM [1] (SensAble Technologies, Inc.) can be used to develop object interactions where various interaction states and state transitions are implemented through forces, rather than through menu selections, as in typical user interfaces. This parallels certain real-world interactions such as sliding an object over a plane or pressing hard to destroy (delete) something. We present a set of techniques that we call haptic state-surface interactions that are designed to make interactions with 3D objects more fluid and natural. We develop the example of drawing a polyline on a curved or flat surface. Control points are selected by touching them and this enables them to be slid across the surface. Simply lifting up the stylus and breaking contact releases them. Points are deleted by pushing them through the surface. Points are cloned by applying force so that they "click-down". We have also developed state-plane techniques that use pop-up orthogonal planes to allow for the positioning of points anywhere in 3D space. We conducted two experiments to evaluate the state surface technique for the task of laying out a spline curve on a curved surface. The first experiment did not show any significant benefit over more conventional methods but lead to a redesign of the state surface interface. The second experiment showed the modified state surface interaction method to be superior in terms of interaction speed and user preferences to the alternatives.

Introduction

In the everyday world when we want to position an object we reach out, grasp it, move it, then release it. In the computer graphics world, to move an object we typically move the cursor to a position over the object, depress the mouse button, move the object using the mouse, and release the mouse button. However, it gets more complicated when operations for adding points and deleting them are required. In widely used drawing packages, such as Adobe Illustrator, there are states for selecting an object, selecting control points, for adding points and for deleting points. Each of these states is accessed via a menu selection and some of them require visiting submenus. It is probably not an exaggeration to say that one of the main barriers to entry of many drawing and 3D sculpting packages is the learning of the many states of the system.

When it is necessary to position objects in 3D things become even more complex. Typically in CAD interfaces this is accomplished via independent plan and elevation view positioning operations. The development of 3 and 6 degree-of-freedom input devices has made it possible to carry out the 3D position task in single movement. However, paradoxically, with this freedom has come a realization that constraints limiting the degrees-of-freedom are often useful for 3D interaction. It is very hard to position objects accurately in 3D; it is easier and faster to position an object against a constraining surface [2]. Such considerations have lead researchers to add physical "props" to virtual environments [3]. For example, Lindeman et al. [4] added a paddle that users physically held in their left hand and saw in the virtual environment in the form of a computer graphics proxy. Using their right hand, users could position objects on this paddle taking advantage of the plane constraints provided. They showed that this improved performance on a number of tasks, including object docking.

In addition to ease of use, there may be design factors that make it desirable to operate much of the time with two degrees-of-freedom with respect to a constraining surface even though the goal may be the creation of a 3D curve. For example, Grossman et al. [5], in an experimental system for designing the principal curves of an automobile design, found it useful to allow users to draw out curved lines on a curved plane that had been previously defined. This deliberate reduction in task degrees-of-freedom was done even though the interface used a six degree-of-freedom input device capable of simultaneously adjusting the position of curve points in 3D space.

The availability of force feedback devices, such as the PHANTOM, makes it possible to create simulated virtual surfaces supporting interaction anywhere in space. In the present work, we have been exploring methods that combine the advantages of surface forces in supporting constrained positioning with a new method for adding state information to the surface. This reduces the need for menu selections and, hopefully, also makes the system easier to learn and use.

A more pragmatic motivation for our work came from a system we have been developing to plan paths for autonomous undersea vehicles (AUVs) using the PHANTOM force feedback device in a Fish Tank VR setup [6]. AUVs commonly "fly" at a constant height above the seabed from one waypoint to the next, and thus should be constrained to a constant height surface. Our initial version of this interface required one button on the PHANTOM stylus to select from a menu and another to add new waypoints. It was also necessary to use a button to move an object, although objects could be selected by simple contact. These conventional object movement and menu interactions seemed to violate the expectation that adding haptics should allow for more direct manipulation of objects without the use of buttons and menus.

The solution we have developed was partly inspired by the pop-through mouse. This work by Zeleznik et al. [7] added a third button state by allowing light force to be registered for one state while firm force caused the button to "pop-through" to another position and another state. In our work we have used this idea in combination with artificial constraint surfaces. Other ideas came from the way Lindeman et al's system used passive haptics to support selection; simply touching the paddle (held in the left hand) with the forefinger of the right hand caused selection of objects on that plane. Lifting the finger from the paddle resulted in release of the object. The essence of our method is to use force states to support interaction modes and transitions between them to support actions such as selection, cloning and deletion.

Haptic state surface

Through a process of iterative design we found that we could comfortably encapsulate four states based upon a surface normal reaction force applied at the PHANTOM stylus tip. We initially used the force profile shown in Figure 1, although we did adjust this slightly based upon the results of our first experiment. These states are as follows:

The ABOVE_SURFACE state occurs when the input device is above the surface by more than 2 mm. No interaction is possible.

The ON_SURFACE state involves light spring forces keeping the stylus tip from leaving the surface or from pushing deeper into it. In this state, objects can readily be slid laterally over the surface. The force gradient (spring constant) used is 0.5 N/mm. When the height is greater than 0.5 mm above the surface, no force is applied.

The IN_SURFACE state results from a downward force exceeding 1.0 N and less than 2.5 N. This causes the stylus to "click-through" like Zeleznik et al's mouse [7] into a stable position between 2 and 4 mm below the visible surface. The force gradient used is 1.0 N/mm.

The BELOW_SURFACE state results from a downward force exceeding 2.5 N. This causes the stylus to break through the state surface, resulting in no forces at all on the stylus tip.

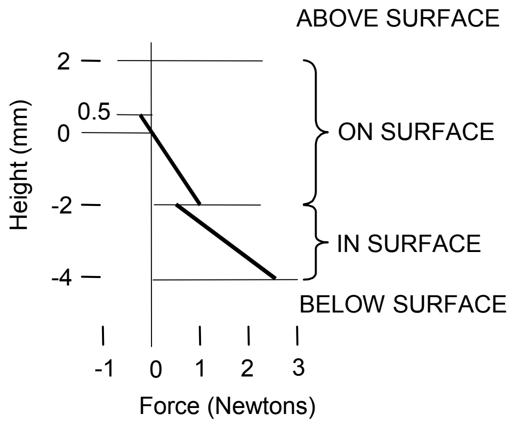


Figure 1. The depth force function and states used in the state surface.

To support interaction with objects on the plane, we change states based on the state transition diagram shown in Figure 2.

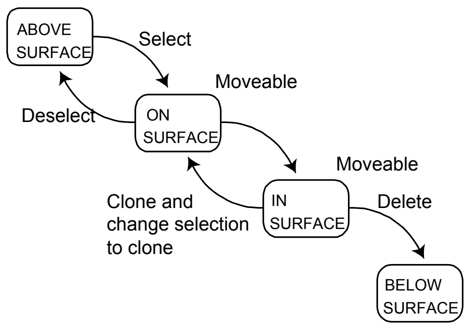


Figure 2. Transitions between states trigger actions such as object selection, object cloning and object deletion.

From the user's perspective, the state interface behaves as follows: to select an object we touch it with the proxy stylus tip. We are now in the ON_SURFACE state and can freely slide it around the surface. The surface feels slightly sticky, because of the small force that holds the stylus tip in the plane.

To duplicate an object, we press down lightly and we feel the stylus click into the IN_SURFACE state. Releasing the extra force causes the cloning operation and the new cloned objects can be slid over the surface to a new position. The user perceives this click down and up as a single operation.

To delete an object, greater force is used, pushing the object right through the surface, into the BELOW_SURFACE state. To continue with interaction, it is necessary to move the stylus back above the surface.

Viscosity and constraint grid

Through informal evaluation we came to the conclusion that adding a certain amount of resistance to movement was conducive to more precise positioning. To accomplish this we added a force vector opposing the motion of the stylus tip, when the stylus tip was in contact with the surface. The function we used was a kind of hybrid between simple viscosity – where the force is proportional to the velocity, and a fixed sliding friction – where the force is constant.

$$\vec{F}_{\text{viscosity}} = -1.2 \cdot \vec{v}$$

$$\text{if } \left\{ \left|\vec{F}_{\text{viscosity}}\right| > 0.3 \right\} \text{ then } \left|\vec{F}_{\text{viscosity}}\right| = 0.3$$

Here *v* is the stylus tip velocity given in units of meters per second (or millimeters per millisecond) and the force output is in Newtons. When the velocity is less than 0.25 m/s, we get the viscosity effect, but when the velocity is greater than this the force is constant, mimicking friction.

As part of the interface we also implemented an optional constraint grid that causes the stylus tip to seek gridlines and the intersections of gridlines. The constraint grid is implemented through the following two lateral functions in the plane of the screen.

$$F_x = 0.4 \cdot \left(\left\lfloor \frac{x}{gap} + 0.5 \right\rfloor - \frac{x}{gap} \right) \qquad F_y = 0.4 \cdot \left(\left\lfloor \frac{y}{gap} + 0.5 \right\rfloor - \frac{y}{gap} \right)$$

Where *x* is the position in the x direction, *y* is the position in the y direction and *gap* is the grid spacing. Summing the forces due to the viscosity and the constraint grid makes it easier to make small adjustments from one gridline to the next.

Creating and editing a spline curve

Thus far we have described how state surfaces can be used to select, duplicate, move and delete points on a surface. We have also extended the technique to the creation of natural cubic spline curves on a surface as illustrated in Figure 3. To initiate a spline curve, we either use a cloner object or menu selection from a haptically enhanced pie menu [8, 9].

Once a new spline curve is instantiated, the first click down on a surface creates the first control vertex and locks it to the surface. This point is now active and can be moved on the surface. Subsequent clicks down and up create new control vertices and makes each active and movable in turn. As with individual points, vertices can be deselected by lifting the stylus tip from the surface. They can be selected again by touch. Vertices can also be inserted into the middle of the spline by selecting an existing control vertex and clicking down. Pressing down harder deletes them.

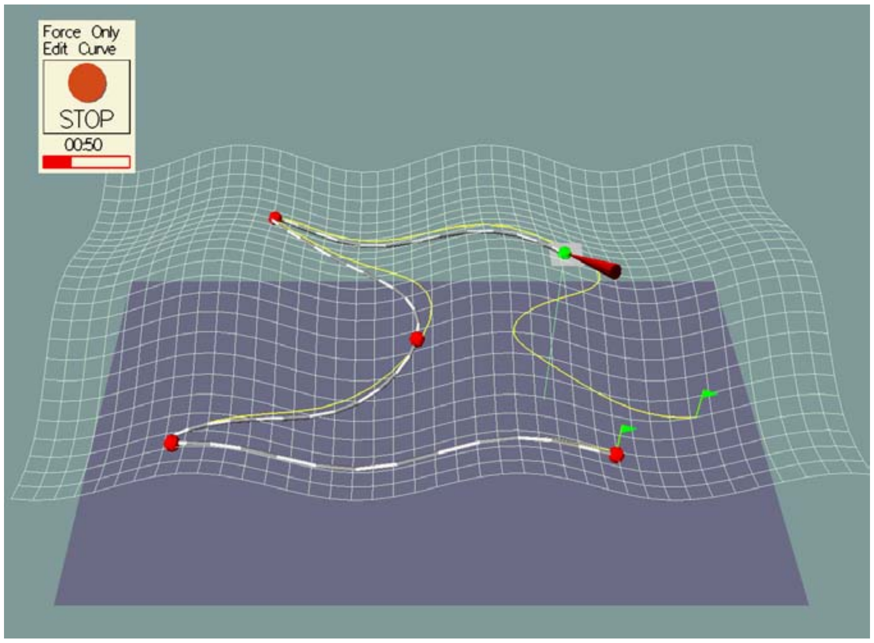


Figure 3. A spline curve being laid out on a curved surface using state-surface interaction.

State planes for 3D positioning

The state-surface methods we have described thus far work well for interaction where an object is being positioned on an existing surface. But what about positioning points freely in space? In order to support 3D positioning of points or small objects in 3D space, and also to allow for the construction of arbitrary 3D curves, we developed the technique we call state-plane interaction. This uses orthogonal planar state-surfaces that popup at when a control point is touched. The particular plane that appears depends on the orientation of the stylus at the moment of contact. The plane that appears is the one most nearly orthogonal to the stylus shaft.

To begin a state plane interaction, we select a root “reproducer” object. Touching it causes a plane to appear through the object. As we shall see, this plane can be horizontal or vertical, but for the moment let us assume that it is a horizontal plane. Clicking down (normal to the state plane) on the root object causes the object to spawn a new instance of that object and this can now be moved around on the plane in exactly the same manner as we have described in state surface interaction. We can also repeatedly click down to create as many instances as we like of the objects.

Unlike state surfaces, state planes are artifacts that are not permanent parts of the environment; they pop up only when needed as positioning guides and they disappear when the stylus proxy is lifted from the object. Using state planes to position objects arbitrarily placed in 3D space is straightforward. Two selections and movements must be made. Typically one will be made with respect to a horizontal plane and a second will be made with respect to a vertical plane. In some instances, the requirement to make two movements instead of a single movement in 3D is inefficient. However, the technique makes it easy to create a set of identical objects all in the same plane. Also, the state plane itself gives a useful cue to, for example, show whether the current object is above or below other objects in the local environment. Optionally, the state planes can also have grid constraints added to them to facilitate precise 3D placement.

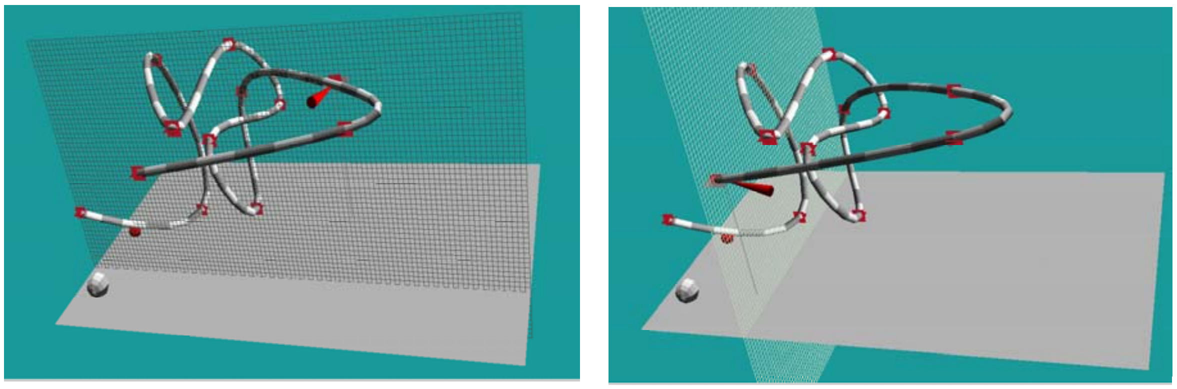


Figure 4. State planes can be used for 3D positioning of spline control vertices.

We have also created a variant of the state surface for 3D spline curves. These use the same techniques that have been described for state surfaces, combined with state planes to allow for the construction of arbitrary curves in 3D space. Any of the spline control vertices can be selected and moved, and adding or deleting of control points is achieved by the same click-down and click-through methods. Using the state surface method to edit a 3D spline curve is illustrated in Figure 4.

Haptic Fish Tank VR environment

The environment we have constructed to implement these ideas is a Haptic Fish Tank VR setup, with a mirror designed to allow the user to place his or her hand in the workspace with the virtual objects (see sidebar). To support some of the design alternatives, we added a second switch located in line behind the built-in switch on the PHANTOM stylus. This second switch provides the ability to pop up context sensitive menus, in a similar fashion as the right-hand button on a mouse.

Evaluation

We carried out two experiments. The first was designed to compare our state-surface technique with reasonable alternatives using menus. Although the results showed no advantage, we learned a number of invaluable lessons that we used to redesign the interface, producing something that was demonstrably superior. The goal of the second experiment was to evaluate this more refined version of the interface.

Experiment 1: State surfaces v. menus

For our first evaluation study we implemented three different interfaces supporting the task of drawing a spline curve on an undulating curved surface. In the first, there was no haptic support for interaction and pie menus were used for all operations instead of state surface interactions. The second was like the first except that, haptic force was provided to allow users to feel the curved surface on which the curve was laid out. The third variation used fully implemented state-surface interaction and required no use of menus (except to start the trial).

We tested two common variations on the task.

Task 1: Create new curve

This task was to creation of a new spline path to match a target spline drawn on the curved surface as illustrated in Figure 3. The subject was required to first lay down a new curve that approximated the target curve. They could then go back and edit the control vertices to attain an optimal match. The target spline was one of a set of 60 used for all conditions. Target splines were non self-intersecting and could have either 4 or 7 control vertices. An attempt was made to ensure the target splines would be relatively easy for the subject to match.

Task 2: Edit curve

This task emphasized the editing of an existing spline. A new target spline curve appeared on the screen together with the curve that the subject had laid down in the first task. The task was to reshape the existing curve to match the new target curve. An optimal match always involved adjusting the position of all control vertices and adding or deleting several points. If the first curve (task 1) had 4 vertices, the second curve (task 2) always had 7 and vice versa.

Haptic Pie Menus

For the conditions in which menu interactions were required, we used a haptically enhanced pie menu as illustrated in Figure 5 [8, 9]. These enhancements included a haptic plane coincident with the menu plane supporting the stylus on the menu. A circular detent force centered within each pie wedge and activated upon stylus entry into the wedge helped to differentiate options. Depressing the back button on the PHANTOM stylus activated menus. Option selection was performed by moving the stylus proxy tip into the appropriate wedge and releasing the button, as shown in Figure 5. If the user changed his or her mind and decided not to make a selection, pulling back off the menu with a force greater than 1.1 N deactivated it and caused it to disappear.

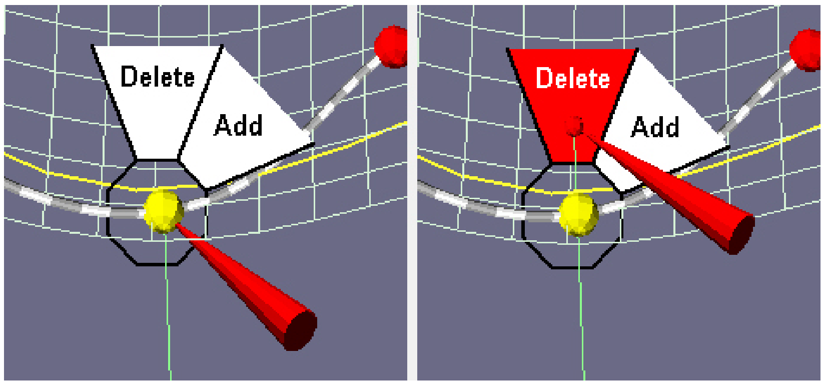


Figure 5. Contextual haptic pie menu.

The experimental conditions were as follows.

Condition 1: Menu & Button (no surface force)

This interface was the best we could design for laying curves against surfaces in 3D environments without haptic support. The subject used the PHANTOM stylus as a 3D positioning device, but no forces were active, except for the ones to support menu interaction.

A new curve was started by means of a menu selection. This generated the first point, which attached itself to the proxy tip. Moving the point was accomplished by simply moving the stylus; pressing the button released the point, created a new point and attached it to the tip. This move-click operation could be repeated to quickly lay out a new line. A menu selection was required to end the curve. To edit the curve, the subject first selected a spline control point. To select a point, the tip of the stylus proxy was first moved within 4.5 mm of the point center causing the point to be highlighted (change color from red to yellow). Selection could then be made by depressing the front button, and was visually indicated by changing the point color to green. When moving the point, it was not necessary for the user to keep the tip of the stylus on the surface; a vertical line was drawn through the tip of the stylus proxy through the point being moved (along the surface) to show the correspondence. Releasing the button released the point. Point addition and deletion was via a menu selection while on a selected point.

Condition 2: Menu & Button (with surface force)
This interaction was essentially the same as in the first condition but with a surface the user could feel. This was the same as the haptic state surface used in the state surface condition, except that it merely served as a support in moving and positioning objects on the surface. Haptic surface viscosity was implemented but grid constraints was not. The menu interactions and method for selecting and moving points were the same as in condition 1.

Condition 3: State Surface (no button)
The state surface interaction methods were used to select, add and delete points as described in the introduction to this paper. Adding and deleting of points was done through the state surface. Movement of the point along the surface was done automatically while the pen tip stayed on the surface; lifting the pen tip off the surface anchored the selected point. The menu was used only to begin a new spline curve.

The experiment environment is shown in Figure 3. A 3D virtual surface was displayed, with the target spline fixed on the surface. The subject's curve was shown as a checkered spline tube, with the control vertices shown as colored spheres. Color was used with the spheres to indicate pen attachment state; red indicated the point was not attached, yellow indicated the pen tip was near the point and the subject could select it, and green indicated the subject was actively manipulating the point. A task panel was displayed in the upper left corner, and showed the current interface condition and task. A haptically enabled round button was used to start and end each experimental trial. The running task time was displayed under the button, as was the progress bar for the entire experiment.

Method
Subjects began each trial by pressing the task bar button labeled "START". This started the task timer, and the subject then carried out the task. When the subject felt that they had best matched their spline to the target spline, they used the PHANTOM to again press the task bar button (now relabeled "STOP"). This stopped the timer and ended the task.
In making their curve matches, subjects were told there were three requirements. They were to work as fast as possible, as accurately as possible, and using the fewest number of control vertices on the matching spline curve. Warning messages encouraged adherence to these requirements. These messages were displayed when (1) the task time exceeded 70 and 120 seconds for 4 and 7-point target splines, respectively, (2) when the mean error exceeded 0.5 mm, or (3) when the number of control vertices exceeded the optimum by 3 or more.

Subjects were trained by taking them through each force condition in both tasks, two or three times. They were told that they would not need more than 10 control vertices to match any curve. The three interface conditions were presented in a different random order to each subject. There were five task pairs given in each condition set. These were alternated so that if on task 1 the subject had 4 control vertices on the target, there would be 7 vertices for task 2 and for the next task 1 there would also be 7. However, we regarded the first of each set of 5 to be a kind of training refresher allowing the subject to get used to the change of interface. This result was discarded. Subjects were notified of condition changes via the task panel and a set of audible beeps.

In summary the experimental factors were as follows:

> **Conditions** (3) Menu & Button (no surface force) / Menu & Button (with surface force) / State Surface (no button)
> **Tasks** (2) New curve / Edit curve
> **Number of target points** (2) 4 / 7
> **Trial block** (2) First / second

Thus we had 5 x 2 x 3 = 30 trials in a trial block. The entire set was replicated (with a different random order of conditions) to allow us to look for learning effects. This yielded a total of 60 trials of which 48 were used in the analysis.

We measured time taken to make the match, mean error and the number of control vertices used by the subject.

Subjects

There were 16 subjects (12 males and 4 females) who were either undergraduate students or staff at the Center for Coastal and Ocean Mapping. All but one was right handed. Subjects were paid for participating.

Results

An analysis of variance (ANOVA) revealed the different interfaces to be significantly different (F(30,2) = 7.46; $p < 0.002$). The Menu & Button (with surface force) condition was the fastest with a mean time of 45.8 seconds. The Menu & Button (no surface force) condition was next with a mean time of 47.3 seconds and the State Surface (no button) was slowest with a mean time of 52.4 seconds. According to the Tukey post-hoc Honestly Significant Difference (HSD) test, the state surface condition was significantly slower than the others, whereas the two menu & button conditions did not differ significantly from each other.

An ANOVA was also performed on the accuracy. There were only quite small variations in accuracy but again, the state surface method fared the worst. The mean error for the state surface was 0.396 mm whereas it was 0.366 for the other two conditions. This difference was statistically significant (F(30,2) = 6.14; $p < 0.006$).

Experiment 1: Discussion

The results failed to support the use of state surface interaction techniques but we learned a number of valuable lessons. Although many subjects liked the concept of the state surface interaction, there were several perceived implementation problems that had to do with the need to stay always in contact with the surface. Subjects would sometimes skip off the surface and lose the connection with the point. Also, it was difficult with the state surface interaction for subjects to cleanly disengage from a control point, due the "sticky" nature of the surface. A third problem was that the method used to click down and add new points was prone to error, subjects would often inadvertently add new points by applying two much force while moving a point along the surface. This was exacerbated by our use of an undulating surface. Subjects generally liked the state surface method as a means of deleting points. One of them said that it felt like popping a balloon.

Interface refinements

Based upon our observations of and feedback from subjects, we redesigned the state surface interaction. We implemented the following changes (see Figure 6 for the modified force profile).

1. We removed the "sticky force" on the surface. The problem with the sticky force was that it caused a control point to move as stylus was lifted from it.
2. We made a number of changes to the state surface interaction forces. We kept the same spring constants (0.5 N/mm and 1.0 N/mm), but adjusted the state transition displacement distances. This effectively increased the ON_SURFACE to IN_SURFACE transition force from 1.0 to 1.25 N. This reduced the likelihood of accidentally cloning a point.
3. We reduced the IN_SURFACE to BELOW_SURFACE transition force from 2.5 to 2.25 N. This made it easier to delete points.
4. We spent considerable effort re-designing the method for releasing objects for the State Surface (no button) technique. Having removed the sticky force, users would be more likely to accidentally release an object as the stylus skipped while dragging. To prevent accidentally dropping objects but still allow a clean lift-off we implemented the following de-selection method as illustrated in Figure 7. The location of the pen tip is tracked during this liftoff; if the tip leaves through the top of the inverted cone, the vertex is anchored. Alternately, if the tip leaves through the side of the cone, the subject regains movement control of the point but can now "fly" with the pen tip in a similar fashion as in the menu & button conditions. While the tip is inside the cone, the vertex visually remains anchored at the liftoff point and is colored yellow to indicate it's in a transition state. We spent considerable effort in designing the release angle of the cone and settled on 39 degrees.
5. We created a hybrid state surface interaction technique that allowed the user to optionally utilize the stylus button to "lock-on" to a point. It was observed that in the first experiment subjects tended to drop points because they skipped off the surface when making rapid movements. In addition, many subjects mentioned that they missed the sense of control the button provided. We experimented with a lock-on mode that allowed a subject the option to, after selecting a point and depressing the front button), move the stylus above the surface and maintain the attachment to the object via a vertical line (as occurred in conditions 1 and 2). This allowed for large-scale movements with less risk of adding or deleting the point inadvertently.

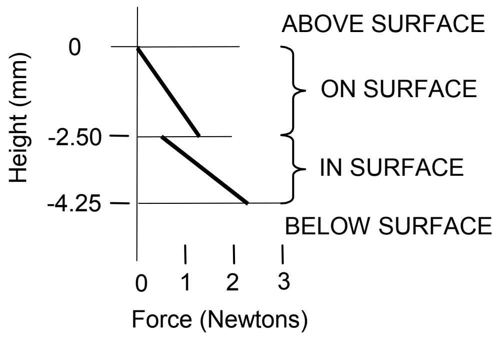


Figure 6. Modified force function and states.

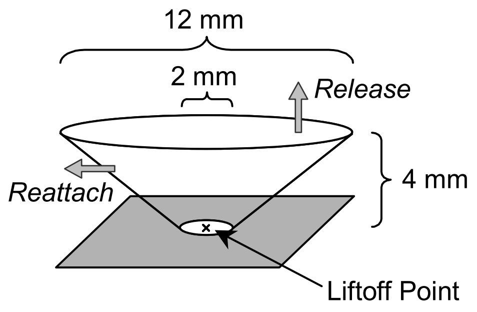


Figure 7. Vertex selection transition region.

Experiment 2: Evaluating the refined interface

In our second experiment, we compared four interface conditions.

Conditions 1 and 2: Same as Experiment 1.

Condition 3: State Surface (no button)
This was like condition 3 of the first experiment but the state surface interaction included refinements 1-4 described above. Selecting, adding, deleting and moving points could all be accomplished by state surface interactions.

Condition 4: Hybrid State Surface (with button option)
This interface was the same as that in condition 3 of the first experiment but with the lock-on method described earlier in the interface refinements section.

In addition we made a number of minor changes to the experimental procedure

1. We used the line-editing task and dropped the create-new-curve task. In most design tasks editing operations consume more effort than the initial drawing.
2. We modified the menu option layout for the Add/Delete menu. We moved the options to opposing directions (Add at 12 o'clock, Delete at 6 o'clock) to address accidental option selections due to their close proximity.
3. We simplified the target splines to simplify the task.

In summary the experimental factors were as follows:

Conditions (4) Menu & Button (no surface force) / Menu & Button (with surface force) / State Surface (no button) / Hybrid State Surface (with button option)
Number of target points (2) 4 / 7
Trial block (3) First / second / third

Subjects were trained in a similar fashion as in the first experiment. The four conditions were given in a different random order to each subject. There were five trials given in each condition set, alternating between 4 and 7 target points. As with the first experiment, we discarded the first trial of each 5 trial condition set. Thus we had 5 x 4 = 20 trials in a trial block. The entire set was replicated twice, with a different random order of conditions for each block. This yielded a total of 60 trials of which 48 were used in the analysis.

We measured time taken to make the match, mean error and the number of control vertices used by the subject.

Subjects
There were 16 subjects, 11 of whom had taken part in the first experiment, which occurred 2 months previously. There were 10 males and 6 females. All subjects were right handed.

Results
The analysis of variance showed a highly significant main effect for condition (F(45,3) = 6.83; $p < 0.001$). The results revealed that the two modified state surface methods were now the fastest. The State Surface (no button) condition was the fastest of all with a mean time of 35.7 seconds. The Hybrid State Surface (with button option) condition was slightly slower, with a mean time of 36.1 seconds. The Menu & Button (with surface force) condition had a mean time of 39.0 seconds and the Menu & Button (no surface force) was the slowest, with a mean time of 39.5 seconds. These results show the state surface techniques to be between 8-11% faster than the menu-based techniques. According to the Tukey HSD test, the conditions formed two groups; the state surface conditions were faster than the menu & button conditions. Within these groups, the differences were not significant.

An ANOVA was also performed on the accuracy. Here we found an almost significant (F(45,3) = 2.8; $p < 0.051$) main effect for condition. The mean errors for the State Surface (no button) and Hybrid State Surface (with button option) were 0.317 mm and

0.323 mm, respectively. The mean errors for the Menu & Button (no surface force) and Menu & Button (with surface force) were 0.337 mm and 0.335 mm, respectively.

Subjects overwhelmingly preferred one of the two state-surface interaction methods. All but two of the subjects ranked the state-surface interaction methods in first and second place.

Discussion

We believe that the majority of improvement in the state-surface technique was due to removing the surface attractive or "sticky" force present in the first experiment. Also contributing to this improvement were (1) increasing the surface force required to create a new point, and (2) enabling large distance moves without constraining the user to maintain the pen tip on the surface. Subjects experience from the first experiment may have also have helped.

All but two of the users preferred the state surface, even though they also liked the haptic pie menus, which were a novelty. Within the group that preferred state surfaces, users were fairly evenly split in their preference between the single button and button-less techniques, although several users became frustrated with the button-less technique. This occurred when they wanted to let go of a selected vertex and the system would not let them.

When deleting using the state-surface techniques, many subjects would "wind up" and "punch down" through the point. Most subjects found this to be very satisfying but not very accurate; sometimes the subject missed the desired point and had to repeat the delete effort, or accidentally deleted an adjacent point. Repeating the delete operation in the case of a missed point was relatively quick though and no subjects complained about it. It is interesting to note that even with the extra time spent in repeating missed delete operations, the state-surface interaction times were significantly faster than when using the menu.

Conclusion

We have presented a set of interaction techniques that enable common operations used in graphical design to be accomplished by haptic force states. Depending upon the normal force at the tip of the stylus against a virtual surface, different systems states are set. By changing the amount of force applied against the surface, the user can effect transitions between states, and this can be used to accomplish actions such as object selection, object movement, object deletion and object cloning. We were successful in encoding four force states in a single surface in a way that users do not find apparently confusing. Indeed, they appear to find this interface more natural and faster to use than the menu-based alternative that we also implemented.

The success of state-surface interactions may partially derive from the fact that they embody haptic metaphors for common operations. Subjects found the touch and move interface analogous to touching and sliding real-world objects. Pushing through for deletion is metaphorically like destroying something by crushing it, or perhaps simply pushing it so deep into the surface that it is lost. The click-down point cloning technique is similar to the use of a rubber stamp that, when repeatedly pressed down, clones the stamp pattern.

Our first empirical assessment of state surfaces failed to show any benefit of the state-surface method in comparison with a more conventional menu-based interaction style. However, based on user feedback we were able to refine the interface. Removing the surface stickiness provided the major benefit and, together with our adjustment of other parameters, resulted in an interface that is both preferred and measurably faster. The fact that relatively small changes in the state surface interface made all the different in creating a demonstrably faster and widely preferred interface points to the importance of getting the force profiles just right. Not the least of the advantages of state surface interaction methods is that it removes the need for a second button on the stylus, an important consideration given that the PHANTOM comes with only one button.

It is possible that advances in force feedback technology would make some of our techniques obsolete. In everyday haptic interaction with the world we typically grasp small objects between thumb and forefinger to move them. Ideally, haptic input devices would support force feedback for a pinch grip, thus providing a very natural way of selecting and deselecting points. But we would still need to invent ways of adding and deleting points. Moreover, each extra degree-of-freedom in force feedback systems adds greatly to the cost. It is possible to purchase two degree-of-freedom force feedback joysticks for relatively modest amounts. Three degree-of-freedom devices are much more expensive and four or more degrees-of-freedom currently costs tens of thousands of dollars. Because of this, we see scope for force states to be used in haptic interfaces for quite some time to come. We believe that they can, properly designed, provide a faster and more natural way of interacting with virtual objects.

Acknowledgements

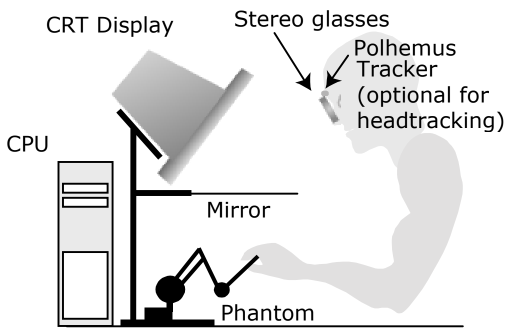
The authors gratefully acknowledge the support of NSF Grant 0081292 and NOAA. We would also like to thank Matthew Plumlee and Roland Arsenault of the Data Visualization Lab for their support, Hannah Sussman for helping oversee the experiments, and CCOM researchers and students who participated in the study.

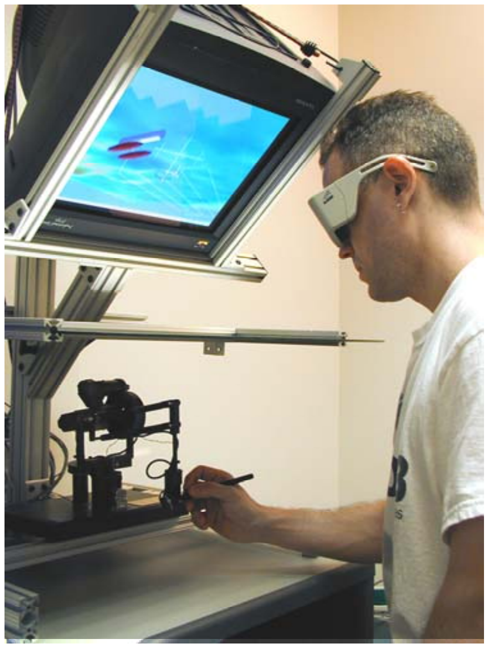
References

[1] T. H. Massie and J. K. Salisbury, "The PHANTOM Haptic Interface: A Device for Probing Virtual Objects," *Proceedings ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Chicago, IL, 1994.
[2] Y. Wang and C. L. MacKenzie, "The Role of Contextual Haptic and Visual Constraints on Object Manipulation in Virtual Environments," *Proceedings ACM CHI 2000*, The Hague, The Netherlands, 2000, pp. 532-539.
[3] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell, "Passive Real-World Interface Props for Neurosurgical Visualization," *Proceedings ACM Human Factors in Computing Systems (CHI '94)*, Boston, MA, 1994, pp. 452-458.
[4] R. W. Lindeman, J. L. Sibert, and J. N. Templeman, "The Effect of 3D Widget Representation and Simulated Surface Constraints on Interaction in Virtual

Environments," *Proceedings IEEE Virtual Reality 2001*, Yokohama, Japan, 2001, pp. 141-148.
[5] T. Grossman, R. Balakrishnan, G. Kurtenbach, G. Fitzmaurice, A. Khan, and B. Buxton, "Creating Principal 3D Curves with Digital Tape Drawing," *Proceedings CHI 2002 Conference on Human Factors in Computing Systems*, Minneapolis, MN, 2002, pp. 121-128.
[6] R. Komerska and C. Ware, "Haptic-GeoZui3D: Exploring the Use of Haptics in AUV Path Planning," to be presented at the *13th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Durham, NH, 2003.
[7] R. Zeleznik, T. Miller, and A. Forsberg, "Pop Through Mouse Buttons: A Simple Hardware Change and its Software UI Impact," *Proceedings 14th Annual ACM Symposium on User Interface Software and Technology (UIST 2001)*, Orlando, FL, 2001, pp. 195-196.
[8] R. Komerska and C. Ware, "A Study of Haptic Linear and Pie Menus in a 3D Fish Tank VR Environment," submitted for publication.
[9] R. Komerska and C. Ware, "Haptic Task Constraints for 3D Interaction," *Proceedings 2003 IEEE Virtual Reality - Haptics Symposium*, Los Angeles, CA, 2003, pp. 270-277.
[10] C. Ware, K. Arthur, and K. S. Booth, "Fish Tank Virtual Reality," *Proceedings INTERCHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993, pp. 37-42.

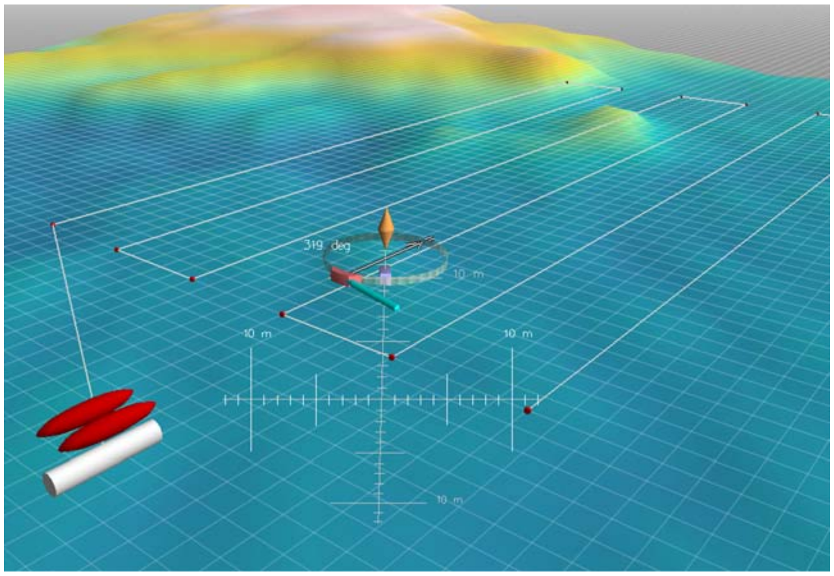
The Haptic Fish Tank





A mirror and tilted monitor makes it possible to co-register the visual and the haptics environment. The user sees the virtual scene beneath the surface of the mirror and uses the PHANTOM device to interact with virtual objects. Stereoscopic viewing is important for good eye hand coordination and it is important that the hand and the virtual computer graphics are co-registered [10]. The user's head position can be tracked to estimate the eye positions and compute the correct perspective view continuously as the user shifts his or her head position. This creates the small but high quality virtual environment that we call the Haptic Fish Tank. The user's hand is not visible, but a graphical proxy for the stylus of the PHANTOM is shown to provide visual feedback for guided hand movements.

In our application, the user's task is to plan the path for an autonomous undersea vehicle. We use haptically enhanced 3D widgets to rotate and tilt the scene. The grid surface shown supports the state surface interactions described in this paper.


319 deg
10 m
10 m
10 m
10 m