Look-That-There: Exploiting Gaze in Virtual Reality Interactions

Robert C. Zeleznik Andrew S. Forsberg Jürgen P. Schulze

Brown University, Providence, RI
{bcz,asf,schulze}@cs.brown.edu

Abstract

We present a suite of interaction techniques that fundamentally leverages the user's gaze direction to provide a range of potential benefits over existing techniques such as reduced arm fatigue, more powerful interaction, and more specialized interaction. Because measuring true gaze direction is problematic, we instead approximate gaze with a non-linear mapping of head orientation that reduces neck strain when looking up or down.

Given the immaturity of gaze-assisted VR interaction, we chose to prototype interaction designs across a variety of fundamental VR tasks that includes 3D point specification, 3D movement, and environment navigation. For each basic task we created a range of exemplary gaze-based techniques that populate three classifications: "Lazy" interactions that minimize or obviate hand movement, "Helping Hand" techniques in which gaze augments conventional interaction as if with an extra hand, and "Hands Down" manipulations in which gaze offloads the hands so that they can operate specialized devices such as a tablet.

Specifically, this paper presents Look-That-There, a technique for moving objects in a virtual environment that does not require hand movement, in addition to gaze-based techniques for selecting menu items, specifying arbitrary 3D points or regions, and orbiting and flying.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction Styles; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality

Keywords: gaze direction, tablet PC, HCI, virtual reality, interaction techniques

1 Introduction

A common concern with VR environments is that although the visual imagery is generally quite compelling, the techniques for interaction with the environment are often unsatisfactory. In many situations, techniques are fatiguing because of the weight of hand-held props and the need for frequent arm movements. In other cases, interaction may seem clunky when compared to desktop or tablet interfaces that exploit sophisticated tactile (e.g., keyboard) and gestural (e.g., stylus) input.

One possible design approach to reducing fatigue and improving the feel of VR interaction is to reduce the encumbrance of input devices through miniaturization or elimination (e.g., using optical tracking). However, these solutions are often difficult to implement, don't fully address issues of fatigue since frequent hand movement may still be required, and do not by themselves enable effective utilization of specialized devices such as tablets.

We instead chose to explore a different approach that was inspired by the observation that much of the input generated by the hand in VR interaction is redundant with the information already provided by the viewer's gaze. Despite the body of work dedicated to exploiting gaze in desktop 2D user interfaces, particularly for the physically challenged, we found only scant references to the use of gaze in virtual environments. With the exception of gaze-based navigation, little consideration has been given to show how gaze can be used to perform common virtual environment tasks, or to demonstrate its synergistic benefits when combined with other interactions. Nonetheless, gaze-based interaction seems to have potential as user evaluations of gaze-based navigation have been promising. Both Bowman [Bowman et al. 1997] and Mine [Mine 1995] have reported positive usability results with gaze-based flying and Chung's orbital mode [Chung 1994] respectively.

Thus the work we present is an attempt to further populate the VR design space with a gamut of gaze-based selection, manipulation and navigation techniques that demonstrate a range of potential benefits. We classify our techniques into three categories that correspond to different approaches to harnessing gaze:

- *Lazy*: By offloading existing hand-based pointing interactions to gaze, hand movements can be minimized or eliminated. This potentially reduces arm fatigue and supports people with physical impairments.
- *Helping Hand*: By treating the user's gaze as an additional "hand", existing hand-based interactions can be extended. This not only allows more parameters of an existing interaction technique to be simultaneously adjusted, but also allows gaze optionally to be used for picking when hand-based picking is inconvenient (e.g., you can see a target, but there is no clear path from your hand to the target unless you raise your hand to point from your eye).
- *Hands Down*: Instead of merely offloading existing interactions from the hand onto gaze, gaze can provide a 3D context for hand-held devices that do not intrinsically support 3D interaction. This facilitates the design of new interactions that can extend sophisticated tactile interactions, based on tablets, keyboards or other specialized devices, into full-fledged 3D interactions.

The point of this work is not quantify whether these interactions are better or worse than previous interactions but instead to shed light on a range of unexplored design possibilities. Although these designs are compelling in their own right, they also provide a menu of research opportunities for further enrichment of VR interaction with gaze.

2 Previous Work

A well-known use of gaze in virtual environments is Mine's gaze directed steering [Mine 1995]. In addition, Mine presented the concept of a Look-At menu in which gaze direction highlights a menu item which is selected by pressing a physical button. Both of these techniques are in the spirit of our work since they reduce arm fatigue and have been easily integrated with other interactions in real virtual environments. However, these techniques just scratch the surface of what is possible with gaze-based interaction. A lesser known interaction that we would classify as a Lazy gaze technique is Chung's orbital mode [Chung 1994], in which a common hand-based rotation of an object is offloaded to gaze such that a viewer can essentially rotate an object in front of them by simply turning their head.

More recently, studies have been performed which compare gaze for selecting objects in virtual environments with other selection techniques. Tanirvedi et al. [Tanriverdi and Jacob 2000] studied gaze vs. arm-extension grasping and found that performance with gaze was faster. Cournia et al. [Cournia et al. 2003] compared gaze with ray casting and found that both were comparable. These studies are relevant to our work because they indicate that some of the problems that we encountered when using head direction to approximate gaze might easily be rectified.

Head Crusher selection demonstrates a design in which pointing is partially offloaded to gaze as both the head and hand locations define a picking ray [Pierce et al. 1997]. The basic Head Crusher design is interesting because even though it offloads something from the hand to gaze, it does not provide the reduction in fatigue benefit of many other Lazy techniques since it actually increases arm movement compared to wand-based picking. However, some Head Crusher variations do exhibit the typical Helping Hand benefits since hand posture can be used to define selection scope which is more difficult with other hand-only interactions.

Finally, there is a long history of offloading hand-input onto gaze in desktop 2D user interfaces (e.g., [Jacob 1990]). Thus in certain cases where everything in VR is represented as a surface and there is no need to specify locations that are not on a surface, the desktop techniques can be directly extended into VR. However in general, gaze-based VR interaction must consider the harder problem of choosing arbitrary points in 3D that are not necessarily on a visible surface. In addition, we want to go beyond just offloading hand input onto gaze and consider how both can be used together—this has not been a primary focus of desktop eye-tracking research.

3 Gaze Directed Techniques

We explored the design potential of gaze directed interfaces by considering three fundamental problems of VR interaction: navigation, pointing/selecting, and moving. For each of these basic tasks, we either prototyped or designed interaction techniques suitable for an immersive four-wall (3 walls and a floor) CAVE environment that would demonstrate the three categories of gaze techniques. In the following subsections we present the basic interactions in terms of how they reflect upon our three principle benefits of VR interaction.

In general, our approach to designing gaze-based techniques was first to address techniques that fall into the Lazy category by merely offloading an existing hand-based interaction technique to gaze. Then we would consider how we might improve an existing interaction if we had an extra "helping hand" available. Interestingly, gaze sometimes turns out to be a better "helping hand" than an actual second hand, such as for automatic speed adjustment while flying. Finally, we would consider how to redesign the interaction if our hands were down out of the environment and holding a wireless, untracked TabletPC. We chose a TabletPC over other devices because of its generality (i.e., it represents the ability to bring virtually any desktop application into a virtual environment) and because we hoped to directly use its sophisticated gestural interaction instead of having to try to replicate it with generally more limited and clunky VR input devices.

Our use of a tablet in VR differs from previous efforts because of the way we combine gaze with tablet interaction. Others have used tablets as a nested 2D surface that you look at directly in an immersive environment (Gorillas in the Bits [Allison et al. 1997], Virtual Notepad [Poupyrev et al. 1998], PDA [Watsen et al. 1999], transparent tablet [Wohlfahrter et al. 2000]). We wanted to explore the complementary concept of hands down (or heads up) interaction, where the tablet is held or rests at waist level to support convenient 2D drawing but the user's gaze is directed forward towards the immersive virtual environment. In some cases, no visual display of the tablet is needed, while in others a representation of the tablet surface needs to be provided as a heads up display. In either case, we were most interested in those interactions where the 2D tablet interactions directly mapped to 3D operations that depended on the viewer's gaze direction.

We also note that we were not able to prototype any techniques using true gaze measurements. Instead, we used head orientation as a rough approximation to gaze, even though we believe that this approximation makes accurate picking slower and increases (neck) muscular strain. Since we do not want to make assumptions about the general viability of gaze tracking in immersive virtual environments, we attempted to address the artifacts of our approximation to gaze tracking when possible. For example, in all of our prototypes, we employed a non-linear mapping function for the vertical angle of the viewing direction so that lower and higher angles could be specified with less neck strain. The downward viewing angle α of the gaze direction is amplified by the factor ϕ:

$$\alpha = \begin{cases} \alpha > \frac{\pi}{12}: & \alpha = \alpha * (1+\phi) \\ else: & \alpha = \alpha * (1+\phi * |\alpha| * \frac{12}{\pi}) \end{cases}$$

In our Cave, we use an amplification factor ϕ of 0.2.

3.1 3D Point Selection

The selection of a point with gaze direction requires at least one additional parameter, because the gaze ray specifies only a line and requires, for instance, the distance from the head to specify a point.

We distinguish two types of environments in which to select a point with head direction: structured environments, which contain virtual objects that can be used to intersect the gaze ray with, and unstructured environments, which do not contain any reference objects.

3.1.1 Structured Environments

Lazy techniques. A representative example of selecting a point with gaze in a structured environment is the selection of a menu item, similar to the same task at the desktop. The menu item intersected by the viewer's gaze ray is highlighted. However, since users often felt lost because of mismatches between their actual gaze and our approximated gaze vector, we also draw a cursor at the intersection of the menu panel with the gaze vector.

Although others have used dwell time to activate menus with gaze, this was not appropriate for our environment, presumably because

Table 1: Overview of gaze directed techniques with examples.

	Principles of using gaze		
Task	**Lazy**	**Helping hand**	**Hands-down**
Point selection, structured environment	menu selection	magnified menu selection	tablet based menu picking
Point selection, unstructured environment	cursor at fixed distance from head	marker placement	tablet w/sideways motion; placing markers while changing properties
3D movement	hands-free Look-That-There	hand or gaze Look-That-There	tablet based movement
Terrain navigation	point-and-fly; orbital mode	speed and orbit control	tablet based navigation control

of our approximated gaze vector. In our testing, head orientation remained constant while users read menu items and so we would either have to use an inordinately long dwell time or we would suffer from the Midas touch problem of inadvertent menu activation. Even with true gaze, we expect that common dwell times of .25 seconds might be less satisfactory than our approach of explicit menu activation with a wireless button or other alternatives that would truly isolate gaze picking from hand actions such as winking or subtle vocalizations.

After a number of trials, we found that it was possible to accurately target menu items, but the strain of the interaction was unsatisfactory, especially for smaller menu items. We considered magnifying menu items in a manner inspired by Apple's Dock in OS X in which menu items are magnified as the cursor moves over them; however that technique only makes things easier to read but does not make them easier to target since the actual target area of buttons never changes. So we instead implemented a highlight magnification mechanism (see Figure 1) in which menu items actually become bigger when intersected by the gaze ray thus increasing their pick region at the cost of reducing the pick region of neighboring menu items. The menu item reverts to its original size when it is no longer intersected by the gaze ray. We found that a scale factor of 1.5 reduced the strain associated with picking our menu items but did not make it appreciably harder to move between neighboring menu items.

We used menu selection within a large grid of icons to calibrate the non-linear mapping function we used throughout our prototypes. Without the mapping function pilot users complained about strain when looking at objects near the bottom and top of the menu. After implementing our non-linear mapping function, we found users were able to select items throughout the menu without noticeable discomfort.

A potential problem with gaze-based menu interaction is that viewers must focus on the user interface and not on the virtual environment (i.e., the task). To address this issue, we prototyped an interface in which scalar-valued menu items can be mapped to the scroll wheel of a hand-held mouse. This allows the viewer to simultaneously change a parameter and observe its effect on the 3D environment. Mapping a value to the scroll wheel is straightforward—the viewer gazes at the scalar-valued menu item and activates it just as if it were a regular menu item (e.g., by clicking the mouse button, or using a subtle vocalization). Instead of performing a regular menu action, the scalar value is mapped to the scroll wheel of the mouse. Thus, the user can focus on the 3D environment while manipulating the scroll wheel to change the mapped parameter's value. The scroll wheel retains its mapping until the user maps a new value to it.

Helping Hand techniques. We did not prototype any helping hand interactions for menu selection, but we propose that in environments with lots of menus, it might be appropriate for gaze direction, perhaps with a dwell factor, to be used to magnify groups of neighboring menu items (similar to the magnification technique just described). Such an interaction might be more natural for reading menu items than pointing to magnify would be and could be considered a helping hand for subsequent menu selection with a hand-based pointer.

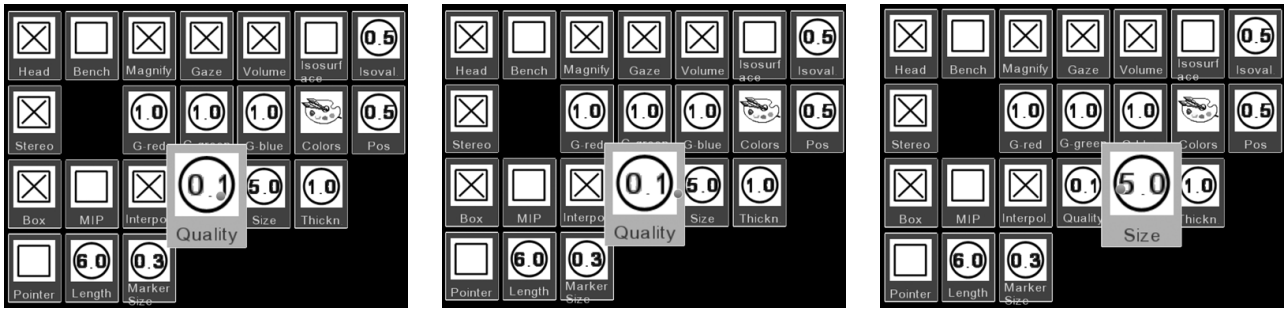
Hands Down techniques. Although it is clearly possible to render menu items directly on the display of a TabletPC, such an approach introduces an undesirable distance between the menu control and the subsequent action in the virtual environment. Therefore, we map the tablet surface to an approximately 30-degree region centered on the viewer's field of view (FOV). Thus a user selects a menu item by looking at it and hovering their stylus over the tablet to establish a mapping between the tablet and their FOV. As the stylus moves over the surface, a cursor moves over objects within the FOV essentially the same way mouse movement maps to a monitor cursor. Clicking on the tablet selects the menu item.

3.1.2 Unstructured Environments

Lazy techniques. Picking a point in an unstructured environment requires information about the distance from the head. A "lazy" way of doing this is to use a fixed distance just as with a hand-held wand. However, moving one's arm around with a wand to point to things is a common interaction even in the real-world, but moving one's head around to point to things is unnatural and so we do not propose any lazy techniques for point selection in an unstructured environment.

Helping Hand techniques. An existing technique for pointing to an arbitrary location is to point with a virtual wand associated with a 6DOF hand-held tracking device. Wand length must be adjusted to reach distant points or to make it more convenient to identify near points—however, making wand length adjustments is typically indirect and can be cumbersome in very large virtual spaces. Therefore, we prototyped a helping hand technique in which gaze is used to adjust wand length. The hand tracker casts a visible ray of unlimited length in the direction the hand points to, similar to the beam of a laser pointer. The system then calculates the intersection of this ray with an invisible plane that passes through the user's eyes and extends along the viewing direction (see Figure 2). The wand length is automatically adjusted to extend to that intersection point. If the plane and the pointer line do not intersect, the marker position remains unchanged.

We chose a plane for the intersection, because our previous implementation, which calculated the intersection of the pointer ray and a ray along the viewing direction, produced too unstable results. We think this was because the viewer had to deal with two degrees



(a) (b) (c)

Figure 1: Selecting a menu item with gaze. The selected item is magnified. It remains selected until the cursor leaves the magnified region. In the above sequence, the cursor (indicated as a small sphere) moves from the Quality widget (images a and b) to the Size widget (image c). Image b shows that even though the cursor is already above the Size widget, the previous widget remains activated.

of freedom to position the head. With the horizontal plane, there is only one significant degree of freedom left, which is the vertical angle into which the user looks. For typical hand orientations, the intersection point moves away from the user when they look up, and toward the user when they look down. This is similar to the intuitive way of looking at objects far and close.

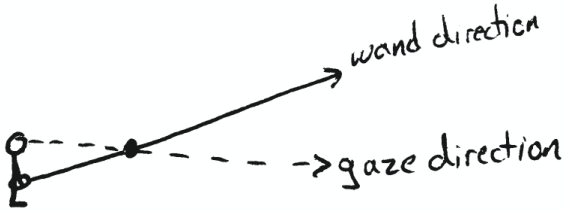


Figure 2: 3D point selection with gaze. A 3D point is specified by the intersection of the viewer's gaze with the ray emitted from their wand. Since these rays do not in general intersect, we actually compute the intersection of the wand ray with a plane through the user's eyes that contains their gaze vector.

Hands Down techniques. We developed two Hands Down methods to select 3D points: an extension of the previously described marker placement, and a tablet PC based method.

The extended marker placement method consists of the above helping hand method, but in addition allows marker properties (color, size, opacity) to be manipulated with the non-dominant hand. By holding a wireless desktop mouse with a mouse wheel in their non-dominant hand, the user can change a marker property by turning the mouse wheel even as the marker is being placed. By gazing at a marker property menu, the user can change the mapping of the mouse wheel at any point during the marker placement interaction.

The other Hands Down method we implemented uses a Tablet PC and a pen (see Figure 3). A rectangular frame, which is shown in the viewing direction, indicates the Tablet PC's drawing surface. When the pen gets near the tablet, a cursor shows up in the rectangle in the viewing direction to indicate the pen's position on the tablet. This allows the user to interact with the tablet without looking at it. Now the user can draw a circle gesture on the tablet, which the system extrudes along the gaze direction to result in a 3D line. By going a step to the left or right, the user sees the line from the side and their gaze determines a 3D point along the ray. The stylus can be used to adjust the location of the 3D point along the ray.

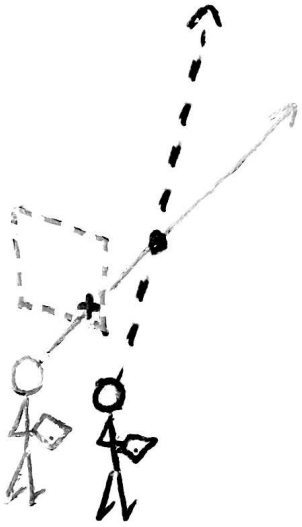


Figure 3: This figure illustrates the two-step procedure for selecting a 3D point with gaze and a hand-held tablet. In the first step (grayed out), the user gazed at a point of interest and used the tablet to fine-tune the gaze direction by offsetting it from the center of the heads-up representation of the tablet (dotted rectangle). This ray is frozen when the pen is pressed. Second, the 3D point is selected by gazing at the desired point on the frozen ray and lifting the pen. The second vantage point is exaggerated in this illustration—moving just a few inches often works well.

3.2 3D Movement

One of our initial motivations for using gaze in a virtual environment was based on a notion of laziness — we wanted to move objects throughout an environment without using a hand-held tracker of any sort. We developed a technique, "Look-That-There", in which gaze is used first to target an object and then to target a destination, such that no hand-based direct manipulation is required. This contrasts with the Head Crusher techniques in which gaze and hand-tracking are used together to select and manipulate objects.

3.2.1 Lazy techniques

Our first prototype addressed the design problem of how to signal object selection and placement with a minimum of effort and without introducing awkward application modes. Initially, we dedicated one hand-held button to triggering selection of the object intersected by the viewer's gaze and a second button to place the selected object at the new intersection of the viewer's gaze with the environment. Although this technique works, it requires users to be aware of the different button functions and the selection state. So instead, we designed an alternative in which we mounted one button on top of the other as a pop-through button[Zeleznik et al. 2002]. In this configuration, pressing lightly gaze-selects an object and pressing firmly moves that object to the new gaze intersection point. We believe that this method is simpler and less error prone because it avoids statefulness when the buttons are released.

3.2.2 Helping Hand techniques

By attaching a tracker to the hand-held pop-through buttons, we were able to explore an alternate design in which gaze is used as a helping hand. In this configuration, the user can target and select an object with a conventional laser pointer by pressing lightly on the button and then manipulate it while continuing to press the button. If the button is released the manipulation ends just like a conventional laser based technique, but if it is pressed more firmly the object and the end of the laser pointer will snap to intersection of the user's gaze with the environment. In essence gaze is like an additional hand that points to the target location while the primary hand is occupied with holding the selected object. However, since it may often be necessary to refine the placement of an object, pressing firmly on the button also automatically adjusts the laser pointer direction and length so that it spans from the user's hand to the target and enables the target location to be adjusted. Figure 4 illustrates this technique.

A combined implementation is also possible that allows users to freely choose between a range of variations of this technique. For example, the initial selection of the object can be made with gaze if the user's hand is at rest by their side, or with a hand-held pointer that automatically appears if they raise their hand to their waist. If the selection is made with gaze, a laser pointer is automatically created from their hand to the selected object so that it can be manipulated, thus potentially avoiding the fatigue of frequent arm lifting to point to objects. By pressing firmly on the button, a destination location can be chosen with gaze if the user's hand is at rest by their side, or it can be selected with the hand-held laser if their hand is raised. The primary drawback to selecting the destination location with the hand is that the object being manipulated may obscure target locations and the ray that could otherwise terminate at the selected object must instead extend through the object so that distant targets can be selected, even if no destination target will be used. Thus the selection and its destination can be targeted both with gaze, both with hand pointing, or one gaze and the other pointing.

3.2.3 Hands Down techniques

We have begun to experiment with using a hand-held tablet as part of the manipulation process. In this approach as with the navigation techniques previously described, the surface of a hand-held tablet is mapped to lie along the user's gaze vector so that gestures can be used on the tablet to select, cut, copy or paste objects. Once selected, an object can be moved by either fixing its location relative to the viewer's gaze allowing coarse-grained "carrying" of objects, or by fixing the tablet's mapping within the virtual environment to support fine-grained depth adjustment. In the former case, the object is effectively fixed as if on a pole to the user's head so that they can carry it to some other location in the environment. In the latter case, fixing the tablet mapping essentially allows the user to place objects in 3D with the hands down technique for 3D point location.

3.3 Navigation

The notion of VR navigation covers a broad domain which we have limited for the scope of this paper to terrain navigation. Within terrain navigation "point-and-fly" has emerged as a popular technique in which the user points a tracked wand in a direction and specifies a rate-of-travel by pushing an analog joystick on the wand forward or backward. The joystick may also be used to rotate the viewer's orientation left or right by pushing it to one side or the other.

While useful in many situations there are two drawbacks to this technique. First, the flying speed may be too small or large for the distances a user needs to travel since the analog joystick control only offers a fixed number of speeds between not moving and some typically arbitrary "full speed". Second, the rotation is around the user's current position and there is no control for orbiting around a region of interest.

3.3.1 Lazy techniques

The direction vector used in this technique can be offloaded to gaze and results in gaze-directed navigation described by Mine [Mine 1995]. However, this technique does not provide a solution to orbiting. A Lazy technique for orbiting, Chung's orbital mode, is to offload hand-based object rotation to head (gaze) rotation.

3.3.2 Helping Hand techniques

Alternatively, gaze can offer a "helping hand" by augmenting point-and-fly with a greater control of speed and the ability to orbit a point of interest. We start with the basic point-and-fly technique. Control over flying speed is achieved by considering the distance to the point on the terrain the user is gazing at. In our implementation, we choose a maximum forward speed (i.e., when the joystick is fully forward) that will let the user reach the point the they are gazing at in two seconds, regardless of how far away it is.

A point of interest can be orbited by gazing at it and then moving the joystick left or right. The left or right movement captures the point of interest and enters the orbiting mode. The further left or right the joystick is moved the faster the rate of rotation. The user can also still move forward or backward along their gaze direction if the joystick is moved forward or backward.

It is interesting to note that in this case we believe it is more effective to use your gaze as a "helping hand" than one's second hand. Not only is the second hand freed from holding an input device but it may also be slower to have to point at an orbit location that is already being gazed upon.

3.3.3 Hands Down techniques

We developed a Hands Down tablet technique in which gestures on the tablet are used to specify the orbit location and flying speed,

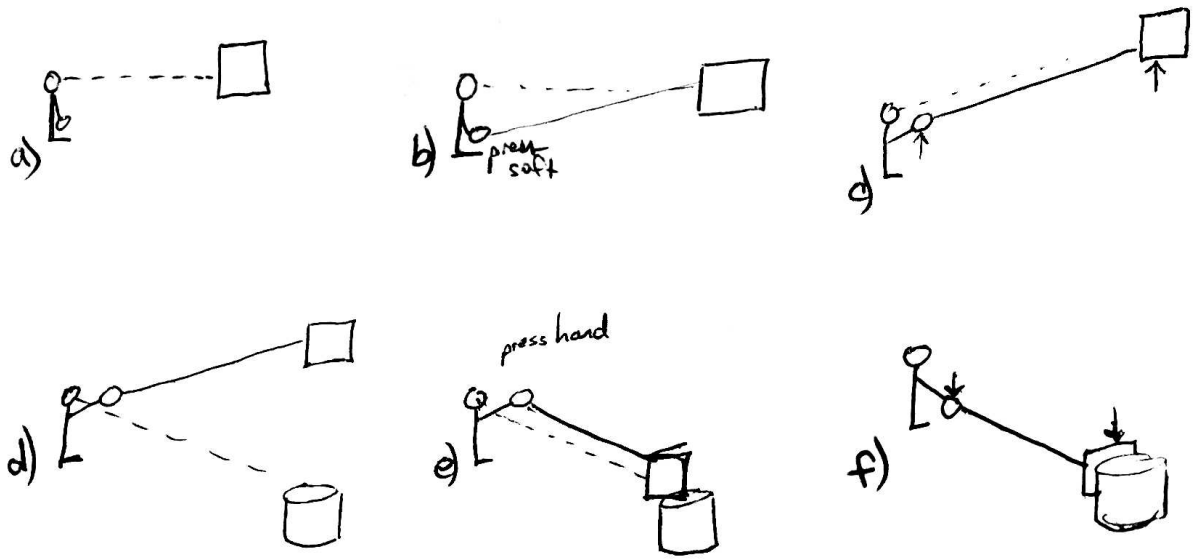


Figure 4: Look-that-there: (a) The user gazes at an object and (b) lightly presses and holds the button to (c) move the selected object. Gazing at another object (d) allows moving the object to it by (e) firmly pressing the button. Holding the button (f) allows adjusting the new position.

and which is extensible so that additional application gestures can be used as well.

Specifically, the technique works as follows. While gazing at some area of the terrain, the user presses on the tablet and drags forward or backward to move forward or backward, respectively. Dragging left or right causes the user to orbit about the point they were gazing at just before dragging left or right. (We did try letting the orbit point freely move with the user's gaze, but found that was less intuitive than fixing it.) The further left or right the stylus is dragged, the higher the rate of rotation. When the user moves the stylus so there is no longer a horizontal component to the mark, the orbiting stops. Since some left or right drift is common when trying to drag the stylus only forward or backward, a buffer area with a width of about a quarter of an inch on the tablet was implemented to help the user control when orbiting mode was invoked (see Figure 5).

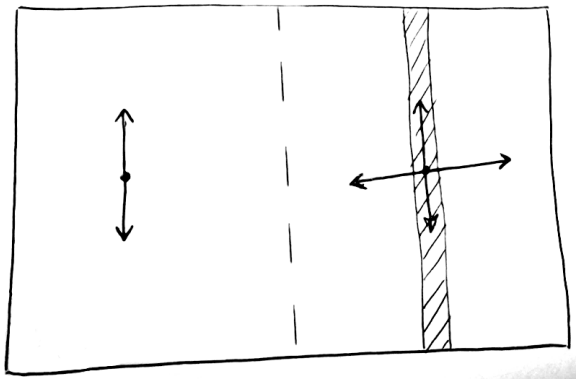


Figure 5: Navigation schematic. The tablet is logically divided in half for navigation. Pressing and dragging in the left half changes the user's elevation. Pressing and dragging in the right half flies and orbits. The vertical strip in the right half is a "dead zone" in which no orbiting occurs until the stylus is outside of it. Small gestures can also be drawn to invoke commands without a perceptible change in position.

Additional commands can be specified using handwriting recognition. For example, drawing an "o" fixes the point the user is gazing at until a "n" is drawn. This differs from the orbiting mode described above where the orbit point was set each time the user dragged the stylus left or right of the center position. Because users are told to draw small letters, the marks (which are drawn in the same way as the flying and orbiting marks are) do not have a perceptible navigation effect.

In addition to flying and orbiting, we also wanted to support a control for changing elevation. Our tablet surface was about 10 inches wide by 8 inches high. When holding the tablet while standing, it was natural to rest the base of one's hand on the surface and draw in only an approximately two-inch diameter circular subset of the full display. Therefore, we were able to logically divide the tablet into two halves—the left and the right. The right half was used for the flying and orbiting marks described above, and single-stroke gesture recognition. The left half was used for a second drawn mark that controlled elevation. When drawing on the left half of the tablet, pressing and dragging forward or backward increased or decreased, respectively, the user's elevation.

Unlike the joystick, the tablet does not give haptic feedback to the user such as snapping into a center position when returning from a press left or right. We tested two ideas to help address this. First, a visual is overlayed on the scene near the top of the Cave display to give feedback as to where the pen is relative to its starting point. Second, we positioned four rubber bands on the tablet (see Figure 6) which apply a force to the pen tip when it is not in the centered position.

4 Future Work

There are a number of areas worthy of further investigation. In particular, it would be quite interesting to compare our implementations using an approximated gaze vector with an implementation that measured true gaze. Although it seems likely that completely accurate gaze measurement would improve most of our techniques, it is less clear what the practical benefit would be using current

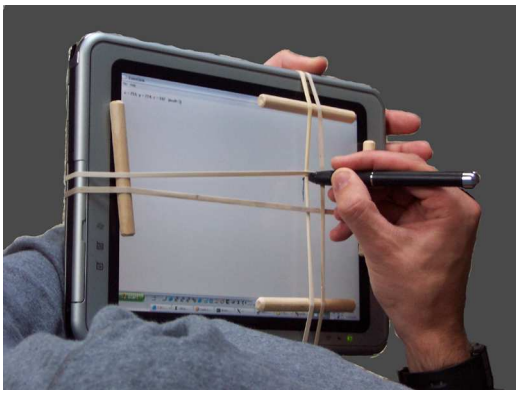


Figure 6: Rubber bands on a Tablet PC provide forces that return the stylus to a "home" position.

gaze-tracking technology particularly in Cave-based environments where it is difficult to unobtrusively observe eye movements. On the other hand, it would be interesting to test gaze-based interaction in Fishtank VR environments where robust gaze tracking might be more practical.

We would also like to consider hybrid techniques such as a Hands Down tablet interaction in which the stylus could also be used as a tracked 3D pointer. A challenge for this technique would be to ensure that conventional stylus interaction is not compromised by tracking the stylus, for example with wires or added bulk.

In this paper, we discussed a few technique designs that we have not yet prototyped and we discussed other techniques that we have not tested against real user populations. In either case, we are relatively confident that the techniques are "usable", but we have little basis for estimating what user preference would be for gaze-based interaction versus other techniques. Therefore, to better understand the applicability of gaze interactions, we think it will be important to conduct relative usability evaluations.

5 Conclusion

We have presented a theory for why gaze-based interaction might be beneficial in virtual environments and we have developed a classification scheme that is useful for developing novel gaze-based techniques. Through various implementations, we have shown Lazy techniques in which existing interactions are offloaded to gaze, Helping Hand techniques in which gaze allows additional parameters of existing techniques to be adjusted, and Hands Down techniques in which we developed novel ways to incorporate tablet-based interaction into virtual environments. Although we have not formally evaluated these techniques, we believe that they provide an important design option both for making VR interaction more effective and more accessible.

References

ALLISON, D., WILLS, B., HODGES, L., AND WINEMAN, J. 1997. *Gorillas in the Bits.* Virtual Reality Annual International Symposium '97 (VRAIS), pp. 69–77.

BOWMAN, D., KOLLER, D., AND HODGES, L. 1997. *Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques.* Virtual Reality Annual International Symposium '97 (VRAIS), pp. 45–53.

CHUNG, J. 1994. *Intuitive Navigation in the Targeting of Radiation Therapy Treatment Beams.* Ph.D. dissertation, UNC-Chapel Hill, Department of Computer Science, May '94, UNC-CH Department of Computer Science Technical Report TR94-025.

COURNIA, N., SMITH, J., AND DUCHOWSKI, A. 2003. *Gaze- vs. Hand-Based Pointing in Virtual Environments.* CHI'03 Short Talk.

JACOB, R. 1990. *What You Look at is What You Get: Eye Movement-Based Interaction Techniques.* Proceedings of ACM CHI'90, pp. 11–18.

MINE, M. 1995. *Virtual Environment Interaction Techniques.* UNC Chapel Hill Computer Science Technical Report TR95-018.

PIERCE, J., FORSBERG, A., CONWAY, M., HONG, S., ZELEZNIK, R., AND MINE, M. 1997. *Image Plane Interaction Techniques in 3D Immersive Environments.* Proceedings of the '97 Symposium on Interactive 3D Graphics, pp. 39–44.

POUPYREV, I., TOMOKAZU, N., AND WEGHORST, S. 1998. *Virtual Notepad: Handwriting in Immersive VR.* Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS '98), pp. 126–132.

TANRIVERDI, V., AND JACOB, R. 2000. *Interacting with Eye Movements in Virtual Environments.* Proceedings of ACM CHI'00, pp. 265–272.

WATSEN, K., DARKEN, R., AND CAPPS, M. 1999. *A Handheld Computer as an Interaction Device to a Virtual Environment.* Proceedings of the 3rd Immersive Projection Technology Workshop (IPTW '99), Stuttgart, Germany.

WOHLFAHRTER, W., ENCARNACAO, L., AND SCHMALSTIEG, D. 2000. *Interactive Volume Exploration on the StudyDesk.* Proceedings of the Fourth International Immersive Projection Technology Workshop, Ames, Iowa.

ZELEZNIK, R., LAVIOLA, J., ACEVEDO, D., AND KEEFE, D. 2002. *Pop Through Button Devices for VE Navigation and Interaction.* Proceedings of VR'02.