The Design and Evaluation of Selection Techniques for 3D Volumetric Displays

Tovi Grossman & Ravin Balakrishnan
Department of Computer Science
University of Toronto
tovi | ravin @dgp.toronto.edu
www.dgp.toronto.edu

ABSTRACT
Volumetric displays, which display imagery in true 3D space, are a promising platform for the display and manipulation of 3D data. To fully leverage their capabilities, appropriate user interfaces and interaction techniques must be designed. In this paper, we explore 3D selection techniques for volumetric displays. In a first experiment, we find a ray cursor to be superior to a 3D point cursor in a single target environment. To address the difficulties associated with dense target environments we design four new ray cursor techniques which provide disambiguation mechanisms for multiple intersected targets. Our techniques showed varied success in a second, dense target experiment. One of the new techniques, the *depth ray*, performed particularly well, significantly reducing movement time, error rate, and input device footprint in comparison to the 3D point cursor.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors, Experimentation.

Keywords: Volumetric displays, 3D interaction, selection.

INTRODUCTION

Volumetric Displays [4] are a new and exciting class of three dimensional (3D) display. They provide a 360 degree viewing angle, and unlike most other 3D technologies, present imagery in true 3D space and do not require users to wear supplementary hardware such as shutter glasses or head mounted displays. These unique properties give them the potential to be beneficial for the display and manipulation of 3D data [2, 8]. To fully leverage the capabilities of these displays, user interfaces must be appropriately designed, and be based on an understanding of the user's capabilities when interacting with this new technology.

One of the most fundamental interface tasks is object selection. In volumetric displays, targets in 3D space must be selected by users who might be located anywhere around the display, a task that is difficult to achieve with existing 2D or 3D selection techniques that are designed for 2D image plane interaction.

Research on selection for 3D virtual reality (VR) environments has introduced two commonly used techniques, which may apply to volumetric displays. Hand extension techniques, or 3D point cursors, directly map the user's hand to the location of a 3D cursor [10, 13, 16]. Ray cursors, or aperture based selection techniques, project a ray from the users hand which is used to intersect and select targets [5, 12, 15]. Studies have shown that for VR environments the ray cursor results in faster selection times [3]. However, this result may not hold in volumetric displays, since all objects are within arm's reach, such that the travel distance required of the point cursor is minimal.

Even if the ray cursor does provide better performance within volumetric displays, it has an inherent problem associated with its use which requires exploration. In dense target environments, the ray may intersect multiple objects, and so the actual target of interest is ambiguous (Figure 1). While techniques for disambiguation have been proposed [8, 10, 12, 14, 19, 23], few have been implemented, and none appear to have been systematically evaluated. Furthermore, the proposed solutions have drawbacks of their own, such as introducing extra buttons or modes for cycling mechanisms [10], or having the system use heuristics to make predictions about the intended target [12, 19], which may not be accurate, or fail under certain environment layouts.

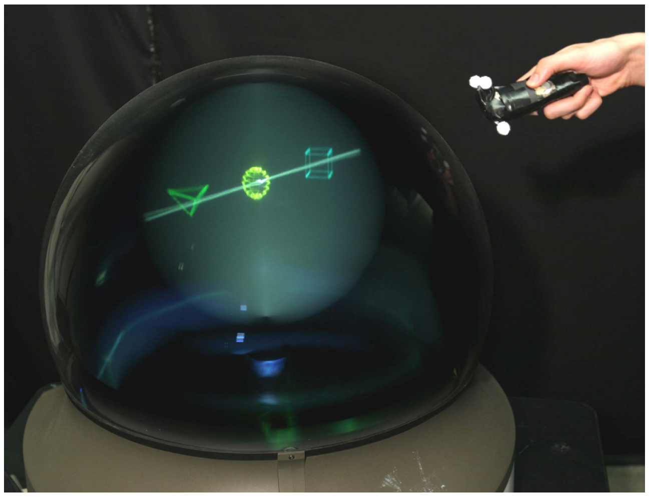
Figure 1. Ray cursor selection in a volumetric display. Multiple targets are intersected, requiring disambiguation.

In this paper, we design and evaluate selection techniques appropriate for volumetric displays. We first implement and evaluate the 3D point cursor and ray cursor in a single target volumetric display environment. Consistent with the VR literature the ray cursor was found to have faster selection times than the point cursor. We then present four different design enhancements to the ray cursor which allow users to disambiguate multiple objects. We evaluate these new techniques in a second experiment, under a dense target environment. Except for the predictive smart ray technique, our new techniques were successful, each with their own beneficial properties. In particular, our depth ray technique significantly reduced movement times, error rates, and input device footprints. We discuss the implications of our work to user interface design, and conclude with some remarks about future lines of work.

RELATED WORK

In 3D virtual environments, selection is categorized as one of the four basic interactions (along with navigation, manipulation, and data input) [13]. A user must be able to specify an object in the virtual world, so they can then manipulate or interact with it. We now give a brief review of selection techniques used in 3D environments, followed by some relevant 2D techniques.

Liang and Green [12] implemented a ray firing selection mechanism that they called "laser gun" selection. A ray is emitted from the user's hand, so the user has control over the start point and orientation of the ray, much like a physical laser pointer. With this technique it was found to be difficult to select small and distant objects due to the required angular accuracy. To alleviate this problem, they created a mechanism called "spotlight selection" where instead of emitting a ray the user emits a conic selection area, with its apex at the user's hand. Other related forms of ray cursors have also been developed, such as aperture based selection [5] and 2D image plane selection [15].

Generally with the ray cursor techniques, only the first intersected object will be selected, even though the ray can intersect multiple objects simultaneously. Under this implementation, it could be very difficult or even impossible to select objects that are further away, depending on the density of the target environment. While no evaluation has been conducted to examine this difficulty, the following techniques have been proposed.

Liang and Green [12] developed a metric for the spotlight selection to decide which object would be selected if multiple targets were within the cone, based on the distance between the target to the apex and central axis of the cone. While this metric may work in some situations, it would fail for objects further away from the user, if there were a number of closer targets along the same line.

An interesting extension to spotlight selection is Shadow Cone Selection [20], which selects targets by sweeping out an area with a cone selection cursor. While useful for group selections, the shadow cone does not provide a disambiguation mechanism, as all targets which are intersected during the entire selection period will be selected.

Hinckley et al. [10] suggest that the ray casting technique could be augmented with a mechanism for cycling through the set of all ray-object intersection points. While this would allow the user to specify the correct target regardless of the density of the environment, it would require extra buttons, it could be cumbersome if there were a large number of targets to cycle through, and it is not clear how the cycling modality would be incorporated into the selection modality. Grossman et al. [8] used forwards and backwards hand movements to cycle through intersected objects. In their implementation, little visual feedback was provided, possibly making it difficult for users to understand how much movement was required to select particular targets.

Olwal et al. [14] describe the flexible pointer, a ray cursor technique which allows users to point to objects which are fully or partially occluded. Users can bend the ray cursor so that it points to their target of interest, without passing through distracter targets. However the technique requires two 6-dof devices to control the cursor, and also requires the user to specify the 3D location of the desired target. Another technique requiring two input devices is iSith [23], where two rays are simultaneously controlled, with their intersection being used to define a target location.

Steed and Parker [19] suggest several new methods for disambiguating multiple targets, such as improving the metrics proposed by Liang and Green [12] for spotlight selection, or gesturing such that the target of interest remains in the selection area over a period of time. While interesting ideas, it is unclear if these techniques were implemented or evaluated.

A more direct method of interaction, in which disambiguation is not an issue, is to use a 3D point cursor which specifies X, Y and Z coordinates for 3D selections [10, 13, 16]. Mine [13] states that in local interactions, a direct mapping from the user's hand to a 3D "virtual cursor or drone" could be used to select an object. The benefit of this method is that it is completely unaffected by the target density of the environment. The problem, however, is that the selections are constrained by three dimensions, resulting in longer selection times. This has been confirmed in a number of studies [3, 6]. Instead of a 3D point cursor, Zhai et al. [24] developed the silk cursor, which is a 3D volume cursor. While using a volume cursor could reduce target acquisition times, it once again produces a difficulty when interacting in dense target environments, as multiple targets may fall within the bounds of the cursor's volume.

Although never implemented, a number of the above techniques were discussed for use within volumetric displays in an exploratory paper with wizard-of-oz prototypes [2].

There has also been recent work in disambiguating multiple targets in dense two-dimensional target environments. The bubble cursor [7] is an area cursor that dynamically changes its size and shape to always capture only the closest target. The splatter technique [17] allows users to spread out overlapping 2D objects, to reduce occlusions. Both these techniques provide inspiration for the dense environment 3D selection techniques which we propose and evaluate.

EXPERIMENT 1: SPARSE ENVIRONMENT

The main goal of this study is to obtain data on known selection techniques for volumetric displays, in a simplified and controlled single-target environment. We will compare the two most popular candidates from the VR literature, the point cursor and the ray cursor. The data which we obtain from this experiment on these two techniques will be used to guide the design of our new techniques, suitable for more realistic, dense target environments.

Although the ray cursor has previously been shown to outperform direct pointing on other platforms [3], it is not clear if this will hold within the volumetric display. Unlike most immersive VR environments, in a volumetric display all targets are within arm's reach, and so the required distance to travel to any target will be minimal. Previous work has shown that the time taken to select an object using the 3D point cursor follows Fitts' Law, and will thus be a function of this travel distance [6]. It is, therefore, of interest to determine if this minimized distance property of volumetric displays is significant enough for the performance of the point cursor to surpass the performance of the ray cursor.

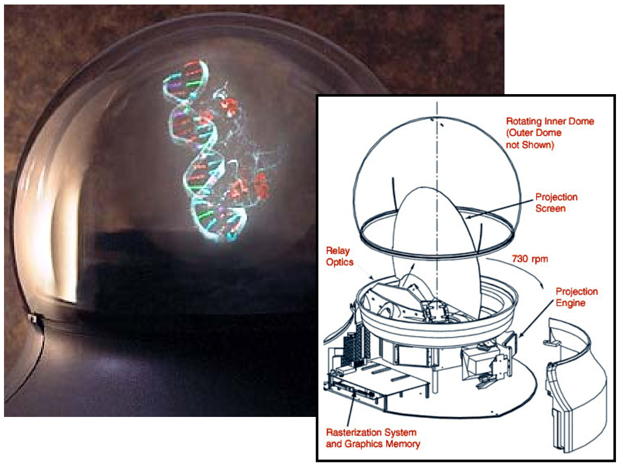
Rotating Inner Dome (Outer Dome not Shown)
Projection Screen
Relay Optics
730 rpm
Projection Engine
Rasterization System and Graphics Memory

Figure 2. Volumetric display.

Apparatus

We used a 3D volumetric display developed by Actuality Systems (www.actuality-systems.com). This display generates a 10" spherical 3D volumetric image by sweeping a semi-transparent 2D image plane around the Y (up-down) axis (Figure 2). There are a total of 198 2D images (slices), each consisting of 768x768 pixels, displayed uniformly around the Y (up-down) axis, resulting in a total of 116 million voxels. The refresh rate of the display is 24Hz. The experiment was run on a 2 GHz Pentium4 computer. For input, an Ascension Flock-of-birds 6-dof tracker equipped with a single button was used, which had a positional accuracy of less than 2mm, and operated at 100Hz.

Participants

Six male and six female unpaid volunteers, ranging in age from 23 to 35, served as participants in this experiment. Participants were screened through the Stereo Optical RADNDOT stereopsis test. One of the twelve participants was left handed and the rest were right handed. Participants controlled the tracking device with their dominant hand.

Procedure

A 3D static target acquisition task was used. Targets were rendered as yellow wireframe spheres. A start target would randomly appear in one of the eight octants of the display. Once this target was selected the trial began, and a goal target would appear at a random position in one of three possible octants which were not adjacent to the starting octant. The distance between the start and goal targets was always set to 5 inches. Subjects were told to complete the trials as quickly as possible, while minimizing errors. The radius of the start target was always 0.45 inches, and the end target took on a radius of either 0.3 or 0.6 inches. For control purposes, users were centered in front of the display and were told not to move their feet during the trials.

The 3D point cursor was displayed as a crosshair with short line segments along the three main axes. The handheld tracker controlled the 3D cursor with a direct one-to-one mapping and a control-display gain of one. To select a target with this cursor, the center of the crosshair had to be positioned inside of it (Figure 3a, b). The ray cursor was displayed as a single line segment, originating at the surface of the display. The input device controlled both the orientation and origin position of the ray, with a direct one-to-one mapping. To select an object with the ray cursor the ray had to intersect the target (Figure 3c, d). For both cursors, selection was initiated by clicking the button.

We also included two common forms of visual feedback, to ensure that the presence or absence of such feedback would not affect the relative performance of the techniques. The feedback which we included were highlighting [13], where a target color would change to red when it could be selected, and shadowing [21], where a 2D projection of both the target and cursor were displayed on a ground plane.

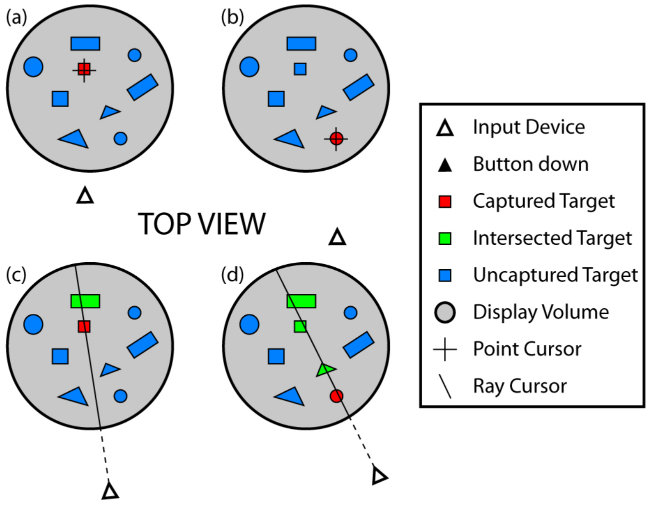

Figure 3. Experiment 1 selection techniques. (a-b) The point cursor, drawn as a crosshair, is controlled by the position of the input device. A target can be selected by positioning the crosshair inside of it. (c-d) The ray cursor, drawn as a line through the display, is controlled with the position and orientation of the input device. The ray cursor will select the first target which it intersects. Note that this legend will also be used for figures 5-8.

Design

A balanced within subjects design was used. The independent variables of interest were cursor type *CT* (point cursor, ray cursor), visual feedback *FB* (none, highlighting, shadowing), and goal target size *SIZE* (0.3, 0.6). The 12 combinations of *CT*, *FB*, and *SIZE* were fully crossed with 8 possible start target positions, each with 3 possible goal target positions, resulting in a total of 288 combinations. Participants were divided into two groups of six. Order of presentation of cursor type was counterbalanced across the groups, with all trials for the first cursor type performed in a single session on one day, and all trials for the second cursor type in a second session on another day. Each session lasted approximately one hour. Within each group, participants were randomly assigned one of the six unique orderings of the three *FB* conditions.

Before each session there was a 2 minute demonstration and warm up. Participants completed all trials for each of the three *FB* values in a session, with trials for each value of *FB* being repeated in four blocks, for a total of 12 blocks per session. In each block, the start and goal target positions, along with the goal target size were presented in random order. A short break was taken between each block.

Results

Task performance was measured by movement time, defined as the time between successful selections of the start and end targets. In our analysis of movement time we removed outliers more than 3 standard deviations from the group mean (1.8% of data) and trials in which errors occurred (11.3% of data). The error rates were not significantly affected by the cursor type or visual feedback.

Analysis of variance indicated that *CT* ($F_{1,\,11}$ = 2180), *FB* ($F_{2,\,46}$ = 105.4), *SIZE* ($F_{1,\,71}$ = 3112), all significantly affected movement time at the $p < .0001$ level. The ray cursor was significantly faster, with overall movement times of 1.27s for the ray cursor, and 1.62s for the point cursor. There was also a significant *CT* x *SIZE* interaction ($F_{1,\,11}$ = 54.84, $p < .0001$), as illustrated in Figure 4. Post hoc analysis shows that ray cursor is significantly faster for both sizes at the $p < .0001$ level. It can also be seen that movement times for point cursor increase to a greater extent than for the ray cursor, when acquiring the smaller target. This interaction is an interesting effect, as it indicates that the ray cursor is less affected by the size of its goal target.

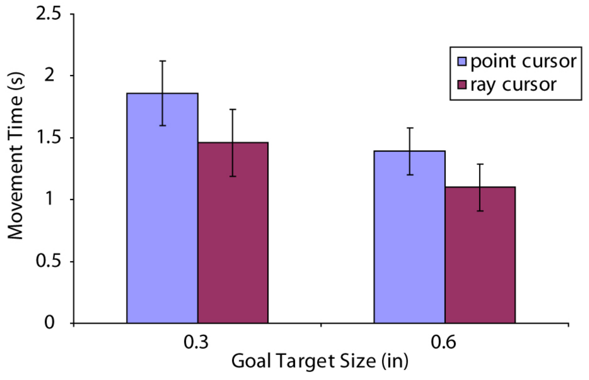

Figure 4. Movement times by target size.

Although there was a significant effect for *FB* on movement time, the feedback did not improve movement times. The times were 1.42s for highlighting and 1.43s for none, which were not significantly different, and 1.53 for shadow, which was significantly higher than the other two feedback conditions ($p < .01$). It is interesting that the highlighting feedback did not improve movement times, showing that the users could suitably perceive when the ray intersected the goal target, and when the point cursor was inside the goal target. The increased movement times in the shadowing condition were likely due to divided attention.

Summary

Consistent with the previous VR literature, we have found that the ray cursor results in significantly faster selection times than the point cursor, even with limited travel distances within the volumetric display. Moreover the ray cursor was less affected by a reduction in target size, likely because a reduced target size means only two dimensions of motor space are reduced, while for the point cursor, three dimensions of motor space are reduced. As for the visual feedback, neither form reduced movement times, or affected the cursors differently.

While the results of Experiment 1 are useful, we are still left with the goal of finding a 3D selection technique which can be effective in both sparse *and* dense target environments. The data clearly shows that the ray cursor is better for sparse environments. As such, we are further motivated to explore enhancements to the ray cursor which provide disambiguation mechanisms for dense target environments. We now provide a detailed description of the new techniques which we have designed.

RAY CURSOR DISAMBIGUATION TECHNIQUES

As previously discussed, a limitation with the ray cursor is that if it simply selects the first target which is intersected, then it could be difficult or even impossible to select occluded targets in a dense environment. This problem is especially important for volumetric displays, as the selection of occluded objects may be more common for the following two reasons.

Firstly, because the display is so close to the user, the user's viewpoint vector will be drastically different from the ray cursor vector, which is emitted from the hand. So even though targets may not be occluded from the user's point of view, they may be occluded relative to the vector of the ray cursor. Secondly, even if targets are occluded from the user's viewpoint, the user will still be able to see them, as all imagery is semi-transparent in the current generation of volumetric displays.

In the following sections, we present four new selection techniques, based on the ray cursor, which allow for the disambiguation of multiple intersected targets. The depth ray, lock ray and flower ray require explicit disambiguation, while with the smart ray the disambiguation is implicitly performed by the system. For the explicit disambiguation techniques we explore both concurrent (depth ray) and sequential (lock ray and flower ray) selection and disambiguation phases.

Depth Ray

The depth ray augments the ray cursor with a depth marker, visualized as a small sphere, existing along the length of the ray (Figure 5a). Along with the standard control of the ray cursor, the position of the depth marker can also be controlled dynamically. The distance between the hand and the surface of the volumetric display is mapped to the position of the depth marker, using an absolute mapping. Moving the hand forwards and backwards will move the depth marker in the same manner (Figure 5b, c).

With the depth ray, all targets which are intersected by the ray are highlighted green. Of these intersected targets, the one which is closest to the depth marker is highlighted red, indicating that it will be selected with a button click. Note that instead of discretely cycling from one target to the next, as suggested by Hinckley et al. [10], and implemented by Grossman et al. [8], we chose to continuously move the depth marker along the length of the ray and select the closest target. This design was inspired by the bubble cursor [7], a 2D selection technique which moves around the screen continuously, and selects the closest target. This technique was shown to outperform the object pointing technique [9], which jumps from one target to the next.

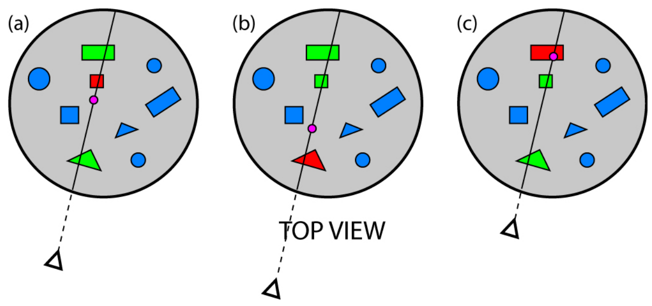

Figure 5. The depth ray. (a) A pink depth marker is used to select the closest intersected target. (b) Moving the input device backwards selects a closer object. (c) Moving the input device forwards selects the further target.

Lock Ray

The depth ray allows users to control the position and orientation of the ray, while simultaneously disambiguating between multiple targets. While this may allow for fast selections, the two phases could potentially interfere with one another. Adjusting the ray position could cause the depth marker to move and vise-versa. As a solution to this, we developed the lock ray, a similar technique, but the selection and disambiguation phases are carried out sequentially, in a two-step process.

With the lock ray, all intersected targets are also highlighted green; however no depth marker is visualized (Figure 6a). To specify the target, the user clicks and holds the button down. At this point, the position of the ray is locked, and only then does the depth marker appear (Figure 6b). The user adjusts the depth marker in a similar manner to the depth ray, and the intersected target which is closest to the depth marker is highlighted red indicating that it can be selected by releasing the button (Figure 6c). Keeping the button down during the disambiguation phase provides a kinesthetically held mode, avoiding confusion between selection and disambiguation phases [18].

The initial position of the depth marker is always the center of the ray, so users will know which way they need to move it to acquire their goal target, even before it is visualized. In the event that the user misses the goal target when the ray is locked, the user's hand can be moved in a direction perpendicular to the ray to cancel the selection.

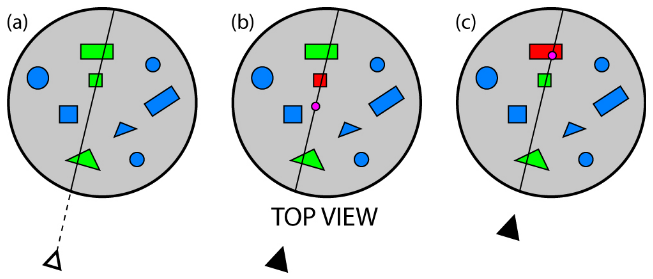

Figure 6. The lock ray. (a) All intersected targets are highlighted. (b) Holding the button down locks the ray and displays the depth marker at its center. (c) The depth marker is controlled with the input device, selecting the closest intersected target.

Flower Ray

The flower ray is another two-step selection technique, similar to the lock ray. The selection phases of the techniques are the same (Figure 7a). However, with the flower ray, when the user clicks and holds the button, all intersected targets animate towards the user's viewpoint, and flower out into a marking menu [11] (Figure 7b). The rationale behind this design is that a marking menu selection should be faster than the disambiguation phase of the lock ray, which is much like selecting an item from a linear menu. This technique is a 3D extension to the splatter technique [17], which spreads out items which are occluded in a 2D layout when the user clicks down.

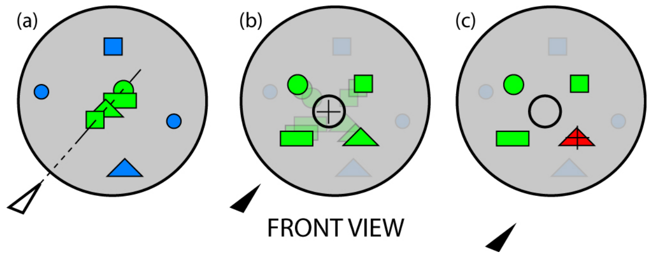

Figure 7. The flower ray. (a) All intersected targets are highlighted. (b) Holding the button down causes all intersected targets to flower out into a marking menu. (c) The input device is used to select the desired target from the marking menu.

When the marking menu appears, a 2D cursor, controlled by the input device, is drawn in the center of the menu. The cursor needs to travel a minimum distance, visualized by a circle, to select any of the targets. Once leaving the bounds of the circle, the target closest to the 2D cursor will be highlighted red, indicating that it can be selected by letting go of the button (Figure 7c). As with the lock ray, a selection can be cancelled if the intended target was not selected and does not appear in the marking menu. To do so, the button is released while the cursor is still inside the bounds of the circle.

While the marking menu will potentially make the flower ray faster than the lock ray, a possible drawback is that users will need to follow the animation and find their intended object in the marking menu. This is not an issue with the lock ray since the disambiguation phase is completed in place. To minimize this effect, we arrange the targets in a clockwise fashion about the marking menu, in order of their depth from the user. The closest target is displayed at the top right, the furthest target is displayed at the top left, and remaining targets are distributed evenly.

Smart Ray

So far, the new techniques which we have described all require an explicit disambiguation phase carried out by the user. We felt that giving user explicit control would be the correct approach, as research in 2D selection interfaces have shown that predictive techniques can be detrimental to performance [7]. However, for the sake of comparison we decided to also include a predictive technique in which the disambiguation phase is performed implicitly by the system. Previously implemented predictive techniques rely on a metric based on the current ray position within the target layout [12, 19]. However such an approach does not guarantee that every target can be selected. The target could occur in an environment such that no matter how the ray intersects the intended target, another target is intersected and selected by the algorithm. As such, we feel it is necessary for the prediction algorithm to be based on the history of the ray cursor's movements.

The design of the smart ray is based on the idea that the intersection of two rays could define a point in 3D space. Instead of taking the intersection of two simultaneously defined rays, which would require a second input device [23], the smart ray takes the intersection of a single ray over a length of time. This technique was recently proposed but not implemented or evaluated [19].

In our implementation, we use an algorithm based on target weights to determine which target should be selected when multiple targets are intersected. Target weights are continuously updated based on their proximity to the ray cursor, and are visualized with small spheres at the center of the target (Figure 8a). The closer the ray comes to the center of the target, the larger the weight increase will be.

As with the previous techniques, all intersected targets are highlighted green. The intersected target with the highest weight is highlighted red, indicating that it can be selected by clicking the button. By using this algorithm, when the ray intersects multiple targets, the user can reposition the ray so that its new position still intersects the intended target (Figure 8b). Even if multiple targets are selected by the new ray position, the intended target will have the highest weight, as its weight has been continuously increasing (Figure 8c).

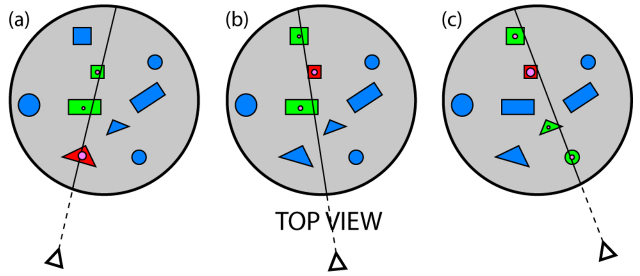

Figure 8. Using the smart ray to select the small square. (a) Target weights are based on the distance from the ray to the target, visualized as spheres in the center of each intersected target. The target with the highest weight can be selected. (b-c) The ray can be repositioned to select an occluded target, by continually increasing its weight.

This technique is similar to the shadow cone [20]. However the shadow cone requires that targets remain intersected during the entire selection. In our initial pilot studies, it was clear that this constraint was much too strong, as it was difficult for users to reposition the ray in a manner that their goal target was intersected the entire time. The smart ray relaxes this constraint. Weights will gradually decrease when the ray is moved away from a target, but as long as the target is reacquired within a reasonable amount of time, it will have the highest weight and can be selected.

EXPERIMENT 2: DENSE ENVIRONMENT

In Experiment 1, we found that the ray cursor is a faster selection technique in comparison to the point cursor in a single target environment. However the ray cursor, in its naïve implementation, is not an appropriate technique for volumetric displays, in a realistic usage scenario, due to the problem of multiple target ambiguity. Motivated by this difficulty, we have presented the design of four new selection techniques, all based on the ray cursor, which provide mechanisms for disambiguating multiple targets.

All of these new techniques have both potential benefits and drawbacks. The depth ray integrates the selection and disambiguation phases, which could minimize times, but could also cause interference between phases. The lock ray explicitly separates the phases, but the disambiguation is accomplished with a linear menu selection. The flower ray provides a marking menu for disambiguation, which should be faster, but users need to follow an animation and find their intended target in the marking menu. Finally, the smart ray provides an implicit and possibly more fluid disambiguation mechanism, but as with any predictive user interface, it could cause frustration if the system misinterprets the user's intent.

In an effort to evaluate the relative effect of these potential benefits and drawbacks, we now present a second experiment, evaluating our four new techniques. The experiment will be conducted in a dense target environment, designed such that it would be virtually impossible to select the target without a disambiguation mechanism. As a result, we omit the naïve implementation of the ray cursor. Instead, we use the point cursor as the baseline for comparison with our new techniques, as its performance should remain unaffected by the density of the environment. This will allow us to identify which, if any, of our new techniques are still faster than using the point cursor, even with the addition of the disambiguation mechanisms. Such techniques, if they perform well, would be appropriate for use within volumetric displays, as they would provide fast selection for both sparse and dense target environments.

Apparatus

Experiment 2 was run on the same volumetric display and computer as in Experiment 1. The tracking technology differed as the experiment was carried out at a later time. The input device was a wireless single-button presentation mouse. Three passive-reflective markers were placed on the device, which were tracked in 3D by a Vicon motion tracking system (www.vicon.com). This allowed us to track both the 3D location and orientation of the input device. The markers were tracked at 120Hz with sub-millimeter precision.

Participants

Eight male and two female new unpaid volunteers, ranging in age from 20 to 25, served as participants in this experiment. Participants were screened through the Stereo Optical RADNDOT stereopsis test. All were right handed and controlled the input device with their right hand.

Procedure

As with Experiment 1, a 3D static target acquisition task was used. To begin a trial, users selected a sphere displayed at the front of the display. After clicking this target, the experiment environment was displayed, consisting of a 3x3x3 array of distracter targets, and a single goal target (Figure 9). The goal target was rendered as a yellow wireframe sphere, and the distracter targets were rendered as blue wireframe tetrahedrons. Since we were mainly interested in the disambiguation component of the techniques, we kept the goal target size constant, with a radius of 0.3 inches. Distracter targets were larger, ensuring that when using the ray cursor techniques to select the goal target, distracter targets would have to be intersected. As in Experiment 1, users were centered in front of the display and were told not to move their feet during the trials.

The goal target was positioned behind one of the distracter targets, either in the left or right row. This resulted in 18 possible target locations (Figure 9). Participants had to successfully select the goal target to complete a trial. Selection errors occurred if the user selected either the wrong target or no target at all.

For consistency across all cursor types, targets were not considered selected by the point, depth, and smart rays until the button was released. A conic selection area was used for the ray cursors, with a 2-degree angle at the apex of the cone. This increased the number of targets which would have to be disambiguated, as more distracter targets would be intersected. However, our initial observations of informal usage showed that the benefit of the conic selection outweighed the cost of having to disambiguate between a few more targets. Although a conic selection area was used, the cursor was still rendered as a single ray.

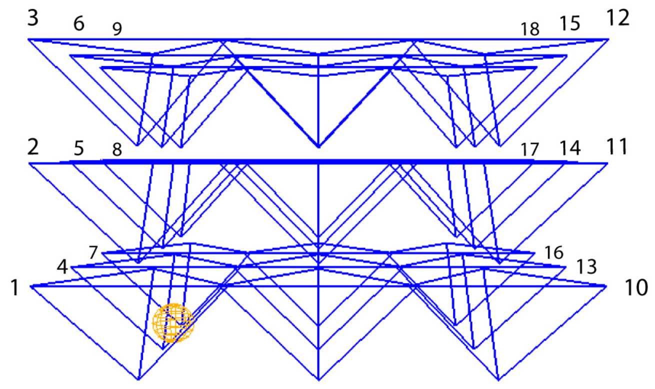

Figure 9. Target environment for Experiment 2, consisting of a sphere goal target and a 3x3x3 array of tetrahedron distracter targets. Target location numbers correspond to which of the 18 distracter targets the goal target is behind.

Design

A repeated measures within-participant design was used. The independent variables were the cursor type *CT* (point cursor, depth ray, lock ray, flower ray, smart ray) and target location *LOC* (1-18). The experiment lasted approximately 90 minutes, and was divided into 5 sessions, with short breaks in between sessions. Each session consisted of all trials for one of the five values of *CT*. Sessions were broken up into 3 blocks of 54 trials, with the 18 target locations appearing 3 times each in random order. This design resulted in 810 trials per participant.

To familiarize participants with the task and selection techniques, eight warm-up trials were performed before each session began. Presentation orders of the selection techniques were counterbalanced using a 10x5 balanced Latin square design. Participants were randomly assigned one of the 10 orderings.

Results

The main dependant measures for the task were trial completion time, error rate, and input device footprint. Trial completion time can be further analyzed into the selection phase time and disambiguation phase time.

Trial Completion Time

In our analysis of trial completion time, we discarded trials in which errors occurred (13.3% of data), and removed outliers that were more than 3 standard deviations from the group mean (1.6% of data).

Repeated measures of analysis showed main effects for *CT* ($F_{4, 36} = 188$), *LOC* ($F_{17, 833} = 16.5$), and the *CT* x *LOC* interaction ($F_{68, 612} = 9.23$) (all $p < .0001$). Average trial completions times were 3.51s for the smart ray, 2.69s for the lock ray, 2.54s for the point cursor, 2.46s for the flower ray, and 2.05s for the depth ray (Figure 10). Post hoc multiple means comparison tests showed that the point cursor was not significantly different from lock ray or flower ray, but all other pairs were significantly different ($p < .001$).

Figure 11 shows the movement times for each cursor by the goal target location. The most prominent effect seen here is that movement times for the smart ray were similar to other techniques for targets on the left side of the display, but much worse when targets were on the right side of the display. This may seem strange since the environment was completely symmetrical. However, because users were right handed, the ray was also coming from the right side. It is clear that due to the arrangement of targets, under this condition, the predictive algorithm broke down.

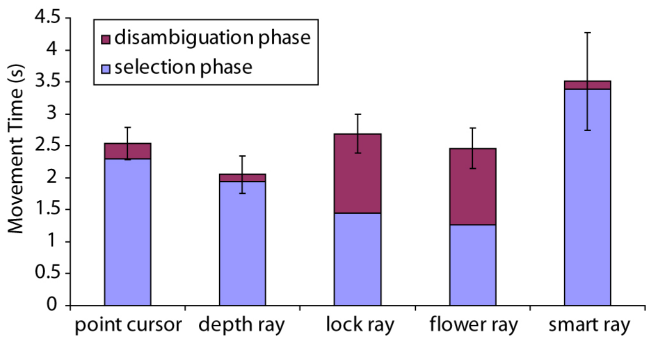

Figure 10. Movement times for each cursor, by the selection and disambiguation phases. Error bars illustrate 1 standard deviation of the total trial time.

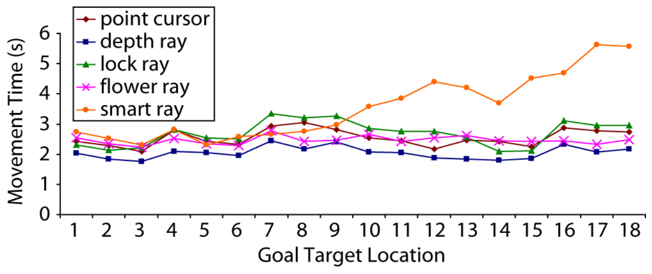

Figure 11. Movement times by target location.

Trial Phase Times

Some interesting effects are seen when we break the data up by the two phases. We define the selection phase as the time until the user clicks the button down. The disambiguation phase is the subsequent time until the button is released. For the point cursor, depth ray and smart ray, the disambiguation phase times will be minimal, only consisting of the time taken to click the button.

Figure 10 breaks the total movement times down by the two phases. Selection phase times were significantly affected by *CT* ($F_{4,\ 36}$ = 433, p < .0001). As expected, the times were slower for the three techniques for which the disambiguation and selection are done concurrently (p < .0001). Of these three techniques, the depth ray was significantly fastest, followed by the point cursor and then the smart ray (p < .0001). The flower ray was slightly faster than the lock ray (p < .01), which is surprising, since the techniques are exactly the same during the selection phase. With the lock ray, we suspect that users were likely planning their disambiguation movements before completing the selection phase, causing the increase in time. In comparison to the flower ray, the selection phase of the depth ray was 0.69s slower, which is the added cost of integrating the disambiguation and selection phases for that technique.

When looking at the disambiguation phases, the lock ray is only slightly slower than the flower ray (p < .05). This shows that with the flower ray, the animation time, and the time to find the target in the marking menu, negates the advantage of using a marking menu. However the advantage with the flower ray is that disambiguation times are more stable, regardless of the target location. Indeed there is a significant *CT* x *LOC* interaction for these techniques ($F_{17,\ 153}$ = 39.5 p < .0001) (Figure 12). It can be seen that the location has much more effect on the lock ray, and less effect on the flower ray. The disambiguation times for these techniques, which are the added costs of a sequential disambiguation phase, are 1.20s for the flower ray and 1.25s for the lock ray. This is much higher than the added cost of 0.69s for the concurrent disambiguation phase of the depth ray, which is why the depth ray was fastest overall.

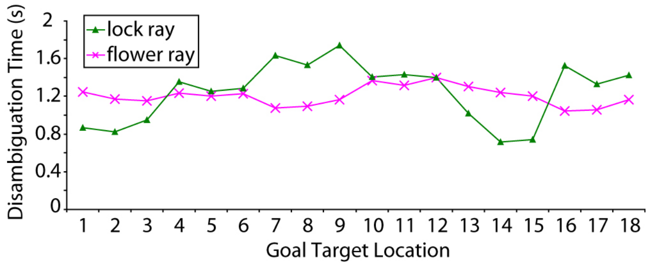

Figure 12. Disambiguation times for the flower ray and lock ray.

Learning

A slight learning effect was seen, with the block number significantly affecting movement times ($F_{2,\ 98}$ = 10.6, p < .0001). Block 1 had the slowest times, averaging 2.78s. Blocks 2 and 3 were significantly faster than block 1, but not from each other, with average times of 2.58s and 2.61s respectively. There was no interaction between the block number and cursor type. This shows that our new techniques were just as easy to learn as the 3D point cursor.

Input Device Footprint

Another variable which we measured was the input device footprint. We measured the length of the total path which the device took to complete the trial. The cursor type had a significant effect on the input device footprint ($F_{4,36}$ = 56.5, p < .0001). Figure 13 illustrates the effect. Post hoc multiple means comparison shows that the lock ray, flower ray, and depth ray all have significantly lower footprints than the point cursor (p < .0001), while the footprint of the smart ray is significantly higher (p < .0005). The reduction of footprint is especially important since a handheld 6-dof device is being used, which can lead to fatigue with extended use [22].

Error Rate

With respect to errors, all of our new techniques performed better than the point cursor. The point cursor had a particularly high error rate of 20.7%. The error rates for the ray cursor techniques were all significantly lower; 13.3% for the depth ray, 11.1% for the lock ray, 10.9% for the flower ray, and 10.4% for the smart ray (all p < .05).

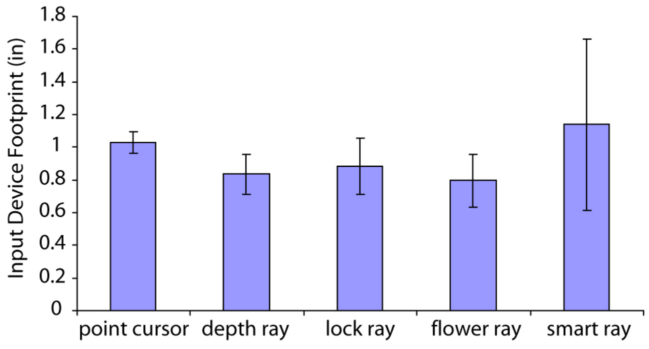

Figure 13. Input device footprints.

IMPLICATIONS TO USER INTERFACE DESIGN

The results of our study have clear implications for the future design of volumetric display user interfaces. A ray cursor metaphor should be used, as it will improve movement times, lower error rates, and reduce the input device footprints, for the common 3D selection task. As for the disambiguation technique, the decision should be based on a couple of considerations.

Most importantly, the input device which will be used should be taken into account. This is because the depth ray requires an input device which has enough degrees-of-freedom to specify the ray *and* specify the depth, since these phases are completed simultaneously. In contrast, the flower ray and lock ray only require enough degrees-of-freedom to specify the ray, since the disambiguation phases are completed independently.

For 6-dof input devices, such as the device used in the experiment, the depth ray would be the most appropriate technique. Such devices could take on the pen shape of a laser pointer, to reinforce the metaphor of emitting a virtual light ray. Alternatively, the system could track the user's index or "pointing" finger, which could specify the location, orientation and depth of the ray, similar to the implementation in Grossman et al. [8].

Another possible option for input would be to use the surface of the display as the input device. Layering the enclosure with a touch and tilt sensitive surface would allow the user to manipulate the ray cursor by either directly touching the display surface or using a stylus. With such a setup, the depth ray would be inappropriate, as it would not be easy to also specify the position of the depth marker. However, the flower ray would work well, as the marking menu stroke could be made along the surface of the display once the selection phase was completed. Similarly, the lock ray could be used, with the depth marker being specified with linear scrubbing on the display surface.

Another input device for which the lock ray or flower ray would be more appropriate is a mouse with extra degrees-of-freedom, such as the 4-dof Rockin'Mouse [1]. The Rockin'Mouse could control the position of the ray through positional movement and the orientation of the ray through tilt. Once completing the selection, subsequent positional movements could be used to specify the depth of the lock ray or make the marking menu selection for the flower ray.

Another factor which should be considered when choosing a technique is the display platform. While our study was focused on selection techniques for volumetric displays, the techniques which we have designed could also be implemented in VR environments. If the depth ray or lock ray were used, then acceleration mappings for the depth marker could be required [16], since intersected objects could drastically range in distance. The flower ray would not be affected by this, possibly making it the most appropriate technique for large VR environments.

Another implication of our results is that predictive techniques, such as the smart ray, should probably be avoided. The smart ray performed poorly based on all measurements. Although in theory the technique should have worked well, the results showed that its performance was highly affected by the location of the goal target within the environment. The poor performance was due to two factors. Firstly, it was difficult for users to keep the ray close to the target while moving from one position to another. Secondly, users preferred to minimize their hand movements, so the change in ray angles was not drastic enough for the technique to work properly.

There are a number of parameters involved with the smart ray algorithm which were chosen in an effort to maximize its usability. Further experimentation could be conducted to optimize these parameters. However, based on our results, it seems unlikely that this would reduce selection times to the extent which our other more successful techniques have. The results for the smart ray were not surprising, as it has been similarly found that predictive selection techniques in 2D can provide poor performance [7].

FUTURE WORK

We have a provided a number of new selection techniques for volumetric displays, which work in both sparse and dense target environments. While three of these techniques performed well, there are areas to explore in the future.

Firstly, in our experiment we used an isotonic 6-dof input device to control all selection techniques. It would be interesting to test our techniques under other input device setups. One notable observation which we made was that users preferred to keep the device close to their body, minimizing hand movements. This may mean that users would find it more tiring if the input device were constrained to the surface of the display.

It may also be interesting to explore multiple pointing devices, as an alternative to the smart ray. This option was considered, however we chose not to include it for the pragmatic reason that we believe that two hands should not be required for a task as common and simple as target selection. While adding a second input device would increase the input bandwidth, this has to be traded-off with an increase in input manipulation complexity.

Along with testing other input devices, future work could also explore other selection techniques. Most interesting would be a 3D extension of the bubble cursor [7], which is a 2D area cursor that disambiguates between multiple targets by changing its capture area dynamically. One possible drawback of the technique is that it could suffer from the same increased input device footprints which we observed for the point cursor. However, its success in 2D environments does warrant the exploration.

It might also be useful to determine how our techniques could be applied to moving selection or tracking tasks. In such scenarios, techniques which use direct hand mappings may perform better, especially those which increase the cursor activation area, such as the silk cursor [24]. Furthermore it would be useful to consider how our techniques could be used for object manipulation. A potential method would be to combine our techniques with direct hand manipulation, similar to Bowman's HOMER technique [3].

CONCLUSION

We have presented an in-depth exploration of selection techniques for volumetric displays. In a first experiment, we found that the ray cursor is significantly faster than the point cursor for single target environments. This is consistent with evaluation of selection techniques on other 3D display platforms, despite the size of volumetric displays.

Based on this result, we were motivated to design enhancements to the ray cursor technique, to provide disambiguation mechanisms such that the new techniques would be suitable for dense target environments. We presented four design alternatives, each with their own unique properties. In a second experiment, we quantitatively evaluated the benefits and drawbacks of each of our new techniques.

The most successful technique was the depth ray, in which users select and disambiguate their target somewhat concurrently. The technique significantly lowered acquisition time, input device footprint, and error rate, in comparison to the 3D point cursor. The lock ray and flower ray also performed well, both reducing input device footprint and error rates, but their acquisition times were not as good.

In summary, we have provided important data on 3D selection techniques for volumetric displays, including new techniques which we have designed which reduce selection times, error rates, and input device footprints. We have discussed the implications of our work to future interface design, and possible extensions to our work. These contributions will be valuable for future designers of interactive volumetric display applications, as object selection will be a fundamental technique for any such application.

REFERENCES

1. Balakrishnan, R., Baudel, T., Kurtenbach, G. and Fitzmaurice, G. (1997). The Rockin'Mouse: Integral 3D manipulation on a plane. *ACM CHI.* p. 311-318.
2. Balakrishnan, R., Fitzmaurice, G. and Kurtenbach, G. (2001). User interfaces for volumetric displays. *IEEE Computer*. p. 37-45.
3. Bowman, D. A., Johnson, D. B. and Hodges, L. F. (1999). Testbed evaluation of virtual environment interaction. *ACM VRST.* p. 26-33.
4. Ebert, D., Bedwell, E., Maher, S., Smoliar, L. and Downing, E. (1999). Realizing 3D visualization using crossed-beam volumetric displays. *Communications of the ACM*. 42(8): p. 101-107.
5. Forsberg, A., Herndon, K. and Zeleznik, R. (1996). Aperture based selection for immersive virtual environments. *ACM UIST.* p. 95-96.
6. Grossman, T. and Balakrishnan, R. (2004) Pointing at trivariate targets in 3D environments. *ACM CHI* p. 447-454.
7. Grossman, T. and Balakrishnan, R., (2005). The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *ACM CHI.* p. 281-290.
8. Grossman, T., Wigdor, D. and Balakrishnan, R. (2004). Multi finger gestural interaction with 3D volumetric displays. *ACM UIST*. p. 61-70.
9. Guiard, Y., Blanch, R. and Beaudouin-Lafon, M. (2004). Object pointing: a complement to bitmap pointing in GUIs. *Graphics Interface*. p. 9-16.
10. Hinckley, K., Pausch, R., Goble, J. C. and Kassell, N. (1994). A survey of design issues in spatial input. *ACM UIST*. p. 213-222.
11. Kurtenbach, G. and Buxton, W. (1993). The limits of expert performance using hierarchical marking menus. *ACM CHI.* p. 35-42.
12. Liang, J. and Green, M. (1994). JDCAD: A highly interactive 3D modeling system. *Computers and Graphics*. 18(4): p. 499-506.
13. Mine, M. (1995). Virtual environment interaction techniques. *UNC Technical Report TR95-020.*
14. Olwal, A. and Feiner, S. (2003). The Flexible Pointer - An Interaction Technique for Selection in Augmented and Virtual Reality. *ACM UIST supplement*. p. 81-82.
15. Pierce, J., Forsberg, A., Conway, M., Hong, S. and Zeleznik, R. (1997). Image plane interaction techniques in 3D immersive environments. *ACM I3D*. p. 39-43.
16. Poupyrev, I., Billinghurst, M., Weghorst, S. and Ichikawa, T. (1996). The go-go interaction technique: non-linear mapping for direct manipulation in VR. *ACM UIST*. p. 79-80.
17. Ramos, G., et al. (2006). Tumble! Splat! Helping Users Access and Manipulate Occluded Content in 2D Drawings. *ACM AVI*. p. 428-435.
18. Sellen, A., Kurtenbach, G. and Buxton, W. (1992). The prevention of mode errors through sensory feedback. *Human Computer Interaction*. 7(2): p. 141-164.
19. Steed, A. (2006). Towards a General Model for Selection in Virtual Environments. *IEEE Symposium on 3D User Interfaces*. p. 103-110.
20. Steed, A. and Parker, C. (2004). 3D Selection Strategies for Head Tracked and Non-Head Tracked Operation of Spatially Immersive Displays. *8th International Immersive Projection Technology Workshop*.
21. Wanger, L. (1992). The effect of shadow quality on the perception of spatial relationships in computer generated imagery. *ACM I3D.* p. 39-42.
22. Ware, C. and Slipp, L. (1991) Using velocity control to navigate 3D graphical environments: a comparison of three interfaces. *Human Factors Meeting*. p. 25-32.
23. Wyss, H. P., Blach, R. and Bues, M. (2006) iSith - Intersection-based Spatial Interaction for Two Hands. *IEEE Symposium on 3D User Interfaces*. p. 59-61.
24. Zhai, S., Buxton, W. and Milgram, P. (1994). The "Silk Cursor": Investigating transparency for 3D target acquisition. *ACM CHI* p. 459-464.