Multi-Finger Gestural Interaction with 3D Volumetric Displays

Tovi Grossman, Daniel Wigdor, Ravin Balakrishnan
Department of Computer Science
University of Toronto
tovi | dwigdor | ravin @dgp.toronto.edu
www.dgp.toronto.edu

ABSTRACT
Volumetric displays provide interesting opportunities and challenges for 3D interaction and visualization, particularly when used in a highly interactive manner. We explore this area through the design and implementation of techniques for interactive direct manipulation of objects with a 3D volumetric display. Motion tracking of the user's fingers provides for direct gestural interaction with the virtual objects, through manipulations on and around the display's hemispheric enclosure. Our techniques leverage the unique features of volumetric displays, including a 360° viewing volume that enables manipulation from any viewpoint around the display, as well as natural and accurate perception of true depth information in the displayed 3D scene. We demonstrate our techniques within a prototype 3D geometric model building application.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]: Interaction styles; I.3.6 **[Methodology and Techniques]**: Interaction techniques.

Additional Keywords and Phrases: volumetric display, 3D interaction, multi-finger and two-handed gestural input.

INTRODUCTION
Viewing imagery on volumetric displays [7, 12], which generate true volumetric 3D images by actually illuminating points in 3D space, is akin to viewing physical objects in the real world. Viewers can use their inherent physiological mechanisms for depth perception to gain a richer, more accurate understanding of the virtual 3D scene. These displays typically have a 360° field of view, and the user does not have to wear hardware such as shutter glasses or head-trackers. As such, they are a promising alternative to traditional display systems for viewing in 3D.

Although these displays are now commercially available (e.g., www.actuality-systems.com), current applications tend to use them as a non-interactive output-only display device, much like one would use a printer. In order to fully leverage the unique features of these displays, however, it would be desirable if one could directly interact with and manipulate the 3D data being displayed. It should be noted that the vast literature on interaction within 3D virtual reality environments is clearly relevant, and we do indeed draw upon this body of previous work. However, volumetric displays present interesting challenges which demand special attention, such that a user interface appropriate for a virtual reality environment may not work well on a volumetric display, and vice versa. For example, in a virtual reality environment, virtual objects could be at a very large distance from the user's virtual position, whereas all virtual objects in a volumetric display are always within arm's reach of a user. However, despite this apparent accessibility, the imagery inside a volumetric display is enclosed by a protective transparent enclosure, which means the user cannot reach in and grab objects, whereas in most virtual reality environments, the user and their hands are often (virtually) inside the environment. These differences can significantly impact the usability of an interface, making it worthwhile to investigate new interaction styles specifically suited to volumetric displays.

In this paper, we investigate interaction techniques for volumetric display interfaces, through the development of an interactive 3D geometric model building application. While this application area itself presents many interesting challenges, our focus is on the interaction techniques that are likely generalizable to interactive applications for other domains. We explore a very direct style of interaction where the user interacts with the virtual data using direct finger manipulations on and around the enclosure surrounding the displayed 3D volumetric image. In our implementation, the enclosure is a hemisphere (Figure 1). If the enclosure was instead a cuboid or cylinder [7], some of the mappings we use may need modification, but the overall ideas would remain applicable.

Figure 1. User working with a volumetric display, with finger input tracked using a camera-based motion tracking system.

UIST '04, October 24–27, 2004, Santa Fe, New Mexico, USA.

RELATED WORK

Much research done on interactive 3D virtual environments to date has relied on stereoscopic displays, either immersive VR systems [5], or non-immersive fish tank VR systems using LCD shutter stereo-glasses [18-20]. Unfortunately, unlike volumetric displays that generate true 3D voxels, these displays can create a conflict between the two mechanisms that give humans stereoscopic vision – convergence and accommodation (see [16] for a review of human stereoscopic vision). By providing two different images for each eye, stereoscopic displays satisfy convergence, but the single image plane is insufficient for accommodation. The result is a tendency for some users to experience nausea and dizziness [13]. Volumetric displays also do not require the use of head tracking, and its associated problems with lag and poor accuracy, that is often used with stereo displays to provide motion parallax.

From a technology viewpoint, volumetric displays can be very broadly classified into three categories. Holographic displays [12] generate 3D images by using microscopic patterns on a physical imaging plane to control the diffraction of light. Static techniques [7] create emissive voxels by directly exciting points within a physical 3D substrate. Swept volume techniques spin a 2D time-varying image about an axis at a sufficiently high speed to enable the human viewer's visual system to perceive a 3D volumetric image by fusing together the successive 2D images into a 3D whole. The technical details of these display implementations are outside the scope of our paper and we refer the reader to [7, 12] for more details.

Given that volumetric displays have not been easily available until recently, there has been relatively little research on how to use such displays effectively in an interactive manner. A speculative paper [1] discusses possible interaction scenarios for volumetric displays, using wizard-of-oz mock-up prototypes to demonstrate various techniques for selection, displaying text and menus, and manipulating objects. However, they did not have or make use of a real volumetric display and as such did not demonstrate any working implementations of their ideas.

From an interaction perspective, the most relevant prior art is in the virtual reality community, that has long explored interactive 3D environments, albeit using various 2D display technologies. This includes work on virtual object selection, virtual object manipulation, menu and command selection, and various 3D widgets.

One of the most basic tasks in any application is object selection. For applications in 3D environments, it may seem obvious to implement selection based on the position of a 3D point cursor [9, 14, 15, 17]. However this requires a user to position the cursor, and thus their hand, in a specific 3D location. Even by altering the mappings between the hand and cursor to reduce the necessary movements[17], it can still be a tedious task to perform repeatedly [3, 4]. The main alternative to using a point cursor is the ray casting selection cursor [11], where a virtual ray is emitted from the user's hand position, so the user has control over the start point and orientation of the ray, much like a physical laser pointer. The first object it intersects is typically selected. Because multiple objects could be intersected, ray casting can present ambiguities not seen with a point cursor. To alleviate this, Hinckley et al [9] suggest that the ray casting technique could be augmented with a mechanism for cycling through the set of all ray-object intersection points. We use a similar approach for object selection in our application.

In many virtual environments, (e.g., [11]), objects are manipulated using a 6-dof tracker. This approach allows for straightforward mappings where the position and orientation of virtual objects correspond directly to the tracker's movements. Others [14] have used direct gestural interaction where hand movements are mapped directly to object movement. The HOMER technique [3] combines ray-casting selection with subsequent direct manipulation: after an object is selected, its position and orientation is manipulated as though it were attached to the hand directly. In Charade [2], freehand gestures were used to manipulate 2-dimensional computerized objects in an augmented reality system.

Conner et al [6] present a set of 3D widgets that allow for indirect interaction with virtual objects through a mediating virtual widget with clickable elements. For example, a translation widget would have virtual handles representing the three primary axes that could be dragged to move the corresponding virtual object in that direction. Many current applications (e.g., MAYA, 3D StudioMax) for 3D modeling and animation make extensive use of such 3D widgets since they can be easily operated with status-quo mouse & keyboards input.

In this paper we will discuss the use of a two dimensional menu placed on the surface of the display. A similar idea was included in [14], where a 2D menu was embedded in the virtual environment. The menu they developed floats in 3D space and includes various widgets such as radio buttons, sliders, and dials. The user interacts with the menu using a ray cursor, so that the user does not have to make large reaching movements. In our implementation, we place the menu on the surface of the display so that the user can also directly reach and interact with it. In the JDCAD system [11] a ring menu was used for item selection, where the items were arranged along the circumference of a circle, and could be rotated until the item to be selected was directly in front of the user.

In short, our literature survey reveals significant work in the general area of 3D interaction that we can build upon in our designs for interactive volumetric displays. However, little of this prior art is directly related to volumetric displays per se. In particular, volumetric displays provide a fixed display area around which to centre interactions, which make it fundamentally different from traditional virtual environments. Thus, interfaces specific to volumetric displays is a ripe area for further exploration.

DESIGN PRINCIPLES

Volumetric Display as the Sole Display

As alluded to in the introduction, one could simply use a volumetric display as an output-only device to display 3D imagery that is created and manipulated using traditional 2D computational environments. In this use scenario, the volumetric display will indeed enable users to better view a 3D scene, but it will be a passive viewing experience, much like watching a movie. We believe that the enhanced 3D viewing capabilities of volumetric displays make it imperative that we begin to explore using it not only to view 3D images, but to also create and interact with those images directly on the volumetric display itself. Thus, in our exploration, we focus on how one might use the volumetric display as the exclusive platform for doing all manipulations with the displayed 3D data. It is critical that we understand the issues surrounding interaction with this class of display in isolation, before attempting to possibly integrate it into environments with multiple heterogeneous displays each with their own strengths and weaknesses.

Multi-Viewpoint and Out-of-Viewpoint Operation

On 2D or stereoscopic displays, users have a single viewpoint of the 3D scene at a given time. As a result, users have to rotate the scene frequently to view the parts occluded from the current viewpoint, or to enhance depth perception through motion. Head tracking can enable more fluid viewpoint changes, but only within the limited range of the display's field of view. Furthermore, most interaction occurs relative to the current viewpoint. In contrast, volumetric displays allow the user to walk around it, or move their head appropriately, to dynamically adjust their viewpoint in a fluid, unobtrusive manner much like they would when looking a physical object in the real world. Furthermore, users can also reach around and interact with the 3D scene from all directions around the display, regardless of their current viewpoint (assuming a moderately sized display). These properties can allow for new interaction techniques beyond what is possible in other display environments and should be exploited to maximal benefit. We attempt to leverage these properties where appropriate in the design of our interaction techniques.

Direct Touch and Gestural Input

One can imagine using many possible input devices for interaction with volumetric displays. For example, in previous empirical work evaluating selection techniques on a volumetric display, we used a 6-dof tracker to control a 3D point cursor [8]. However, the nature of the display, with 3D imagery floating within the enclosure, tends to evoke a strong tendency for people to touch it. Indeed, we have observed countless visitors to our lab attempting to point to parts of the displayed 3D scene by touching the surface of the display's enclosure, or gesturing with their fingers over it. This anecdotal evidence suggests that direct touch and gesture based input could be particularly suited as an input modality to enable rich, high quality, interaction. We explore this style of input throughout our interface designs.

SYSTEM HARDWARE & SOFTWARE

Display Device

We use a 3D volumetric display from Actuality Systems (www.actuality-systems.com). It generates a 10" spherical 3D volumetric image by sweeping a semi-transparent 2D image plane around the up axis (Figure 2). A total of 198 2D slices (images) of 768x768 pixels each are uniformly displayed around the center axis, resulting in a total of 116 million voxels. The display has a refresh rate of 24Hz, and was driven by a 2 GHz Pentium4 computer on which our application software ran. The display can be viewed from any direction around the hemispheric dome, without requiring the user to wear any hardware. Thus, multiple people can view the imagery simultaneously, each from a different viewpoint, while maintaining the context of their shared physical surroundings. We note that the imagery in the current generation display cannot be rendered with full opacity. Furthermore, any sort of hidden-surface removal would require head-tracking for a single user, and would be impossible with multiple users. While this restriction to semi-transparent imagery can be a limitation for some applications, it does not hinder our ability to investigate user interface issues and we can indeed sometimes take advantage of this inherent transparency in our designs.

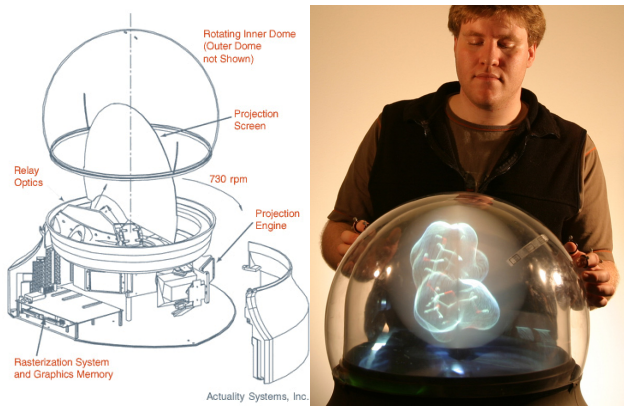

Figure 2. Volumetric display

Finger Tracking

A Vicon motion tracking system (www.vicon.com) is used to track the positions of markers placed on the user's fingers. The Vicon system uses several high-resolution cameras to track the 3D location of multiple passive reflective markers in real time. In addition to tracking the location of the markers in 3D space, the system can uniquely identify and label each marker according to its position on a user's fingers. The 3D coordinates of these labeled markers can then be streamed in real-time to other applications. Our prototype uses four cameras for tracking, two of which are seen in Figure 1. In our current work, we track markers on the index fingers of both hands, and the thumb of the user's dominant hand (Figure 3).

We use the labeled marker data, in conjunction with knowledge of the precise topology and 3D spatial location of the display's enclosure in the tracking volume, to simulate an enhanced touch sensitive display. Our system

categorizes the precise positional information of the tips of the two index fingers and thumb into one of three discrete states: "down" – when touching the surface of the display's enclosure, "hovering" – when within 6 cm of the surface, and "up" – when more than 6 cm from the surface. We also detect static postures and dynamic gestures of the fingers by examining the relative distance between the markers.

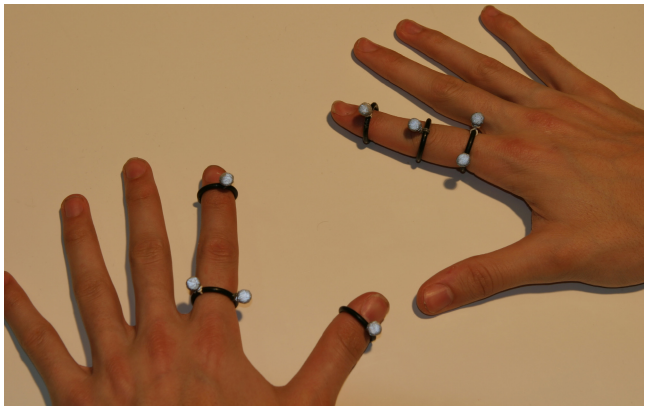

Figure 3. Markers used for tracking finger positions. Multiple markers enable us to track bending of the index fingers, as well as finger tip position and orientation.

Note that it is technically possible to make the display's enclosure directly touch sensitive with current transparent resistive overlay technology, but the cost would likely be prohibitive given low production volumes. As such, we use this motion tracking system to simulate a touch sensitive display surface. Furthermore, this tracking system allows us to explore postural and gestural input that would not be possible with only a simple touch sensitive overlay. We also note that while current generation tracking technology requires markers for robust tracking, improvements to computer vision techniques may reduce or possibly even eliminate the need for markers in the future.. While the inconvenience of using markers does marginally detract from the overall usability of our prototype system, this tracking system allows us to explore advanced freehand interaction techniques today, *before* marker-free tracking becomes widely available. As such, this hardware setup should be viewed simply as an enabling technology for our prototype, rather than one that would be used in any future real implementation of our interface ideas.

Software

Our application software was written in C++ and OpenGL, with a custom OpenGL driver specific to the volumetric display. Marker tracking and labeling was performed using Vicon's standard tracking software, and the data streamed in real-time to our application. The markers were tracked at 120Hz. We could not detect any perceptible latency in the marker data or in the movement of virtual finger representations relative to the actual finger movements.

COMMAND INPUT

Given that we intend to perform all interaction on and around the display itself, and wanted to avoid using additional input devices like keyboards, we implemented two techniques to facilitate command input using the fingers: *surface menus* and a set of *postures and gestures*.

Surface Menus

Similar to interfaces for 2D touch screens, we display frequently used commands as buttons on the surface of the display. We call this *surface menus* (Figure 4). Because there is 3-10 cm gap between the edge of the display volume and its enclosing surface, these options could not be digitally displayed directly on the touchable enclosure. Since this gap will likely not be present in future implementations, we felt it was reasonable too simulate the buttons using pre-printed acetate overlays (Figure 4). We provide two surface menus, one for each hand. The buttons on the non-dominant hand's surface menu are used for kinesthetically held transient modal commands. These provide context for the dominant hand's interaction with the system while the non-dominant hand's index finger is down on the appropriate button (much like the use of a "shift" key in a regular keyboard). The buttons on the dominant hand's surface menu are for other frequently used commands, and are executed with a quick tap.

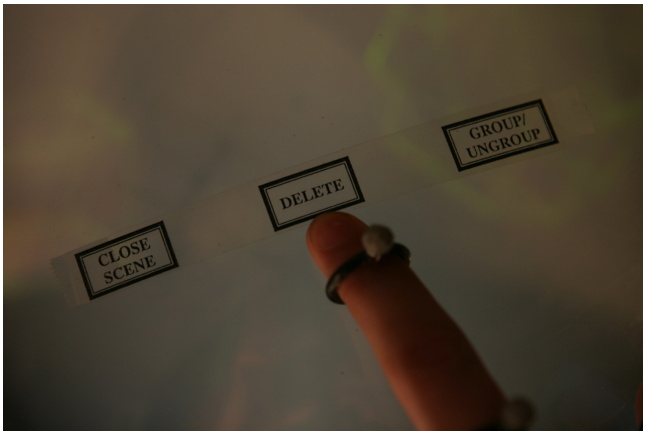

Figure 4. Surface menus. Physical buttons taped onto the display's surface can be tapped to execute a function or tapped-and-held to maintain a mode.

When either index finger hovers over any surface menu button, a caption associated with its function is displayed to the user, similar to the bubble help in conventional GUIs .

In our implementation, the locations of the surface menus were fixed. However, if such menus were to be displayed digitally, their location could be dynamically adjusted. It would be desirable to orient the menus relative to the user's body position, such that they would always appear on either side of the user for easy access. We implement dynamic orientation for the button captions, such that the caption is always facing the user, regardless of the user's position around the display. The user's position is estimated by examining the position and orientation of their fingers.

Postures and Gestures

While surface menus provide a nice mechanism for command input, there are instances when it could be inconvenient to have to touch the surface menu buttons to invoke a command. For example, if the user is

manipulating a virtual object, it may be easier to enter commands using other finger movements. We also wanted to experiment with more than one command input mechanism, to enable later determination of optimal solutions. Accordingly, we developed a set of hand postures and gestures which can be carried out on or off the surface of the display. We infer the set of postures based on the shape of the fingers, while the set of gestures is determined based on the dynamic characteristics of the fingers' movement over time. Figure 5 illustrates this set of postures and gestures. The commands associated with each posture and gesture will be described as we progress through the paper explaining the various interaction techniques.

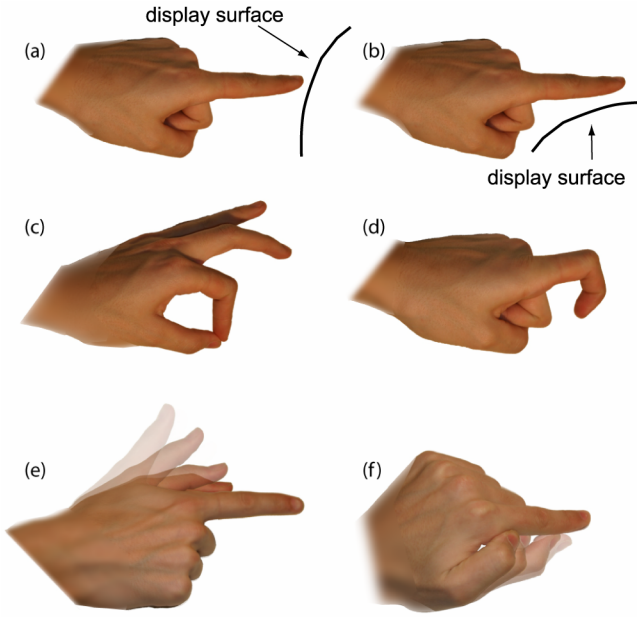

Figure 5. Postures and gestures. (a) point posture: index finger points towards the display. (b) flat posture: index finger is parallel to display surface. (c) pinch posture: tips of index finger and thumb brought together. (d) curl posture: tip bent towards base of finger. (e) trigger gesture: thumb presses against index finger (f) scrub gesture: thumb scrubs along index finger in either direction.

INTERACTION TECHNIQUES

For interacting in three dimensions, we need to support a variety of basic operations such as file visualization and browsing, selection, translation, scaling, and rotation. The following techniques were developed in the context of a 3D model building application, but can be generalized to other volumetric display applications.

SurfaceBrowser

In order to allow for basic file operations of load, save, organize, copy, and delete we developed a simple file/object management mechanism called *SurfaceBrowser* (Figure 6). The SurfaceBrowser displays various objects by organizing them into cells of a 2D array. Four such arrays, or *pages*, are then projected around the entire inner surface of the display, allowing the user to easily interact with the objects by touching the surface of the enclosure directly above them. The pages either contain *models* or *scenes*. Models are primary shapes used in building more complex scenes. The contents of each cell rotate slowly to aid in their visualization. We have currently implemented display support only for 3D models and scenes since they are the primary data types of interest in our 3D model building application. However, support for other data types such as images could easily be implemented within the same SurfaceBrowser framework.

While the SurfaceBrowser is displayed, it can be rotated by scrubbing the non-dominant hand's index finger along the surface of the display. This feature allows the user to bring regions of interest closer to him/herself, although this is not strictly necessary in a volumetric display since the user could walk or move their head around the display to look at various parts of the SurfaceBrowser. In a sense, this rotation technique supports "lazy" operation, which may be desirable in some situations.

A flat arrow cursor is displayed below the dominant hand's index finger, which is used to perform basic operations with objects in the SurfaceBrowser. Touching the surface with this finger while the cursor is above an object (either model or scene) selects it, lifting the finger deselects it. While selected, an object can be moved from cell to cell by dragging within a page, or copied by dragging to another page. The object can also be dragged into a trash can area at the top of the display to delete it. A quick tap on any object opens it. When this occurs, the selected model or scene smoothly animates from its 2D form on the surface of the display, to its 3D shape at the center of the display. We use smooth animated transitions throughout our prototype to provide users with a sense of continuity as they move from action to action.

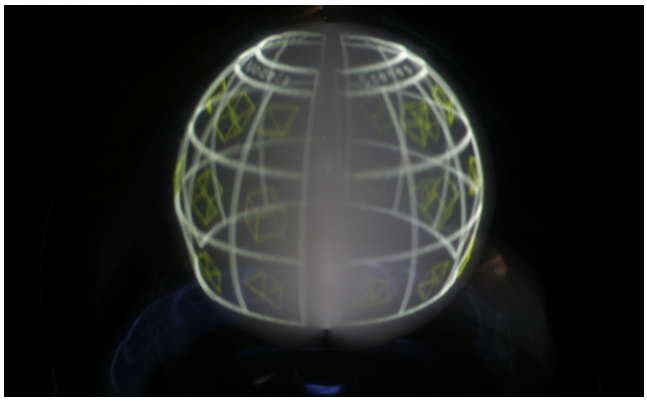

Figure 6. SurfaceBrowser. The distortion at the centre of the display is an artifact of the physical display mechanism.

Model Transformations

Once a model has been opened, we allow for rotation, translation, and scaling. While we did explore simultaneous rotation and translation for six-degree of freedom manipulations, we limited the interactions to distinct modal operations to enhance precision. We now describe the interaction techniques that allow 3D transformations to be applied to the models.

Rotate

Rotation is initiated by touching the display with the dominant hand's index finger. The finger is then dragged across the surface of the display, and this movement is transformed into rotation of the model, as if there were a stick connecting the finger to the model's center (Figure 7a,b). This provides two degrees of freedom of rotation. A third rotational degree of freedom is achieved by twisting the hand, while the index finger is still down. This rotates the model about the vector defined from the finger to the model (Figure 7c). Rotation stops when the finger is removed from the surface of the display.

Note that we deliberately chose rotation axes that are defined by the display's hemispheric surface and the user's hands, in order to keep the mappings simple. Although our hemispheric volumetric display does have a well defined up-down axis, it does not have any inherent left-right or front-back axes. As such, although we do define a global three axis coordinate system (with admittedly arbitrary choices of left-right and front-back axes) that we use for snapping operations described later, we did not want to impose this global axes triad on the user for the basic transformation, particularly since the user could be performing these transformations while standing anywhere around the display.

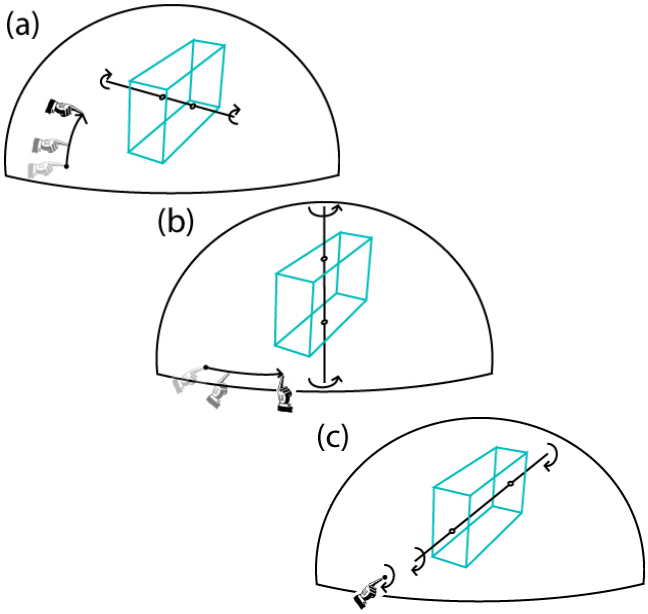

Figure 7. Rotating a model. (a, b) Moving finger across the surface rotates the model as though its centre was attached by a stick to the fingertip. (c) Twisting the hand about the fingertip rotates the model about the vector from its centre to the fingertip.

Translate

Translation is imitated by assuming a pinch posture (Figure 5c) with the dominant hand. While pinched, moving the hand in any direction moves the model the same distance in that direction. The metaphor here is that of picking up an object with a pinch grip and moving it. When translating the model away from the user, it is possible that the user's hand may collide with the display. Thus, we provided added functionality to translate the model towards or away from the user in a relative manner, accomplished through the scrub gesture (Figure 5f). The direction of the scrub gesture determines the direction of the model's translation along the vector defined by the index finger. When the dominant hand leaves the pinch posture, and is not scrubbing, translation stops. Since entering or exiting the pinch posture can cause unwanted translations, we provide a clutching mechanism to freeze the model momentarily to ensure precision translations. This freezing action occurs whenever the non-dominant hand's index finger is in the curl posture (Figure 5d). The metaphor is that of curling the fingers around an object to hold it still, as one might do in the physical world.

Scale

Unlike translation and rotation which are performed with a single finger, scaling is a bimanual technique. To scale a model, both index fingers are placed on and dragged along the surface of the display. Sliding the fingers further apart on the surface increases the scale, while sliding them together decreases it. The object is scaled uniformly along all dimensions. This is similar to the technique presented in [10] for scaling in a 2D drawing program. Scaling stops when the user lifts either finger from the surface.

Visual Feedback for Transformations

While performing any of the transformations, a colored 3D icon is drawn at the center of the model, indicating which transformation is currently being applied (Figure 8). The icons are displayed oriented towards the user's current hand positions to facilitate viewing. If the transformation has not yet been initiated, the icon is white, indicating to the user that the posture of their hands is close to that required to begin the corresponding transformation. For example, if both of the user's hands are close to touching the surface of the display, the scale icon will be displayed in white. Once both fingers make contact with the surface the widget will become colored. This provides a nice way to guide users into appropriate postures for transformation actions, and also to reduce accidental triggering of transformations.

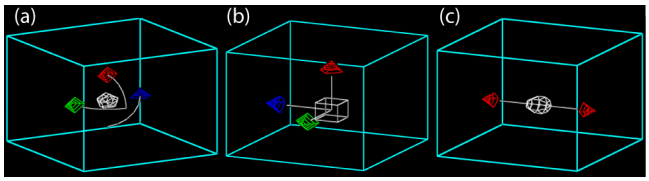

Figure 8. Visual feedback for transformations. (a) Rotate. (b) Translate. (c) Scale

Constrained Transformations

The rotate and translate transformations discussed allow users to simultaneously control three degrees of freedom. In most common 3D graphics applications, such transformations are limited to one or at most two degrees of freedom primarily because they are performed using 2D input devices within a two-dimensional perspective viewpoint. While allowing users to simultaneously move or rotate objects along three axes simultaneously on a

volumetric display is very powerful, it is sometimes useful to constrain transformations to a particular axis for precision movements. To support this, we created a mechanism where users can add or remove axes to which subsequent transformations will be constrained.

Axis Definition

The constraint axis specification mode is entered and maintained while the non-dominant hand's index finger is held on the "axis" surface menu button. While in axis mode the dominate hand can create, activate, or deactivate constraint axes. When in a pointing posture, a white constraint axis preview line is displayed as the finger hovers over the surface of the display. The position and orientation of the preview line matches the vector of the finger, so it appears as though it is a ray being emitted from the finger tip. If the preview line is close enough to one of the global primary axes, or the centre or object axes of any models in the scene, it will snap to that axis. This aids in precise positioning of the constraint axis. If the finger taps the surface while still in the pointing posture, the constraint axis will be added, and is displayed as a thick red line extending through the display (Figure 9). Tapping either end of the constraint axis will deactivate that constraint axis, and it will appear as a short white tick mark. Multiple deactivated constraint axes can exist at a time. The tick marks will appear as long as the non-dominant hand maintains the system in axis mode, and tapping any of them in the flat posture will activate that constraint axis. The active constraint axis is deactivated if another constraint axis is created or activated, so that only one constraint axis can be active at a time.

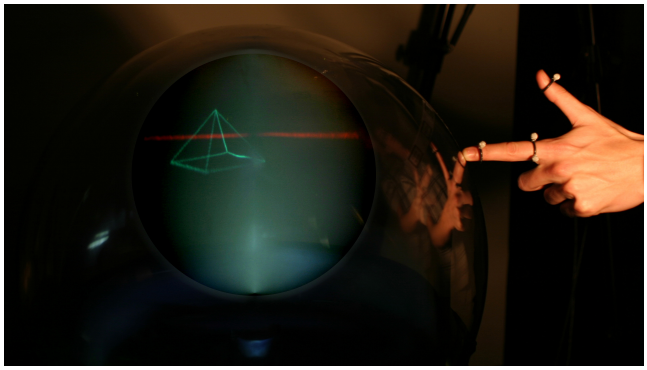

Figure 9. Constraint axis. These can be placed arbitrarily by pointing to define the axis and touching the surface of the display to add it to the scene.

Constrained Transformations

If a constraint axis is active, then all transformations will be constrained to it. For rotation, only the component of the movement of the index finger that is perpendicular to the constraint axis is applied to the rotation of the active model. This causes the model to rotate about the defined axis. A large cylindrical widget is drawn perpendicular to the constraint axis, providing feedback for the user as to where their finger should be dragged for effective rotation. Similarly, when translating, only the component of movement parallel to the constraint axis is used. The translation is thus constrained to that axis. By default our scale function is a one degree of freedom operation, where objects are scaled uniformly along all dimensions. However, when a constraint axis is active, the scale operation is constrained to apply only along that axis.

From Models to Scenes

Now that the main interaction techniques for manipulating individual models have been described, we will discuss the techniques involved in combining multiple models to build up scenes.

Add model

Additional models can be added to the current scene by depressing the "Add Model" surface menu button with the non-dominant hand's index finger. When this is done, the SurfaceBrowser is shown around the perimeter of the display, while the current scene continues to be rendered in the centre (Figure 10). As always, while the non-dominant hand's finger remains down, it can be dragged across the surface to rotate the SurfaceBrowser, and a model is added to the scene by tapping it. The SurfaceBrowser disappears when a model is added, or when the non-dominant hand's finger leaves the surface of the display.

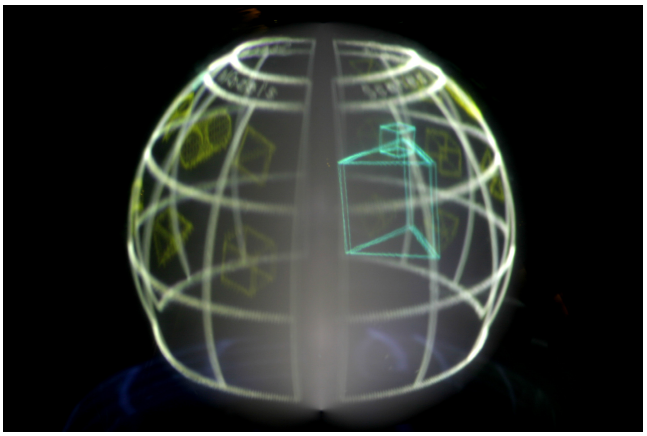

Figure 10. Adding additional models to the scene. The surface browser allows context to be maintained while making a selection that affects the scene.

Selection

With multiple models in a scene, we must select which models are to be manipulated before performing any further transformations. We support model selections with a scene using a ray cursor [11]. In a recent (as yet unpublished) experiment, we found that a ray cursor was more effective than other techniques for a static selection task within a volumetric display. The ray cursor can be used by tapping-and-holding on a surface menu button with the non-dominant hand. A virtual "light ray" appears to emit from the user's finger and is rendered as a yellow line on the volumetric display (Figure 11). When the ray intersects a model, the model is highlighted by changing color, and a trigger gesture (Figure 5e) is used to select or deselect it. Models turn blue once they are selected, as a visual indicator of their state. If the ray cursor intersects multiple

objects, only one of them will be initially highlighted, but hand movements can be used to cycle through the possibilities. Moving the hand forward will highlight the model further from the hand, and moving the hand backwards will highlight the model closer to the hand. Subsequent actions affect only the selected models.

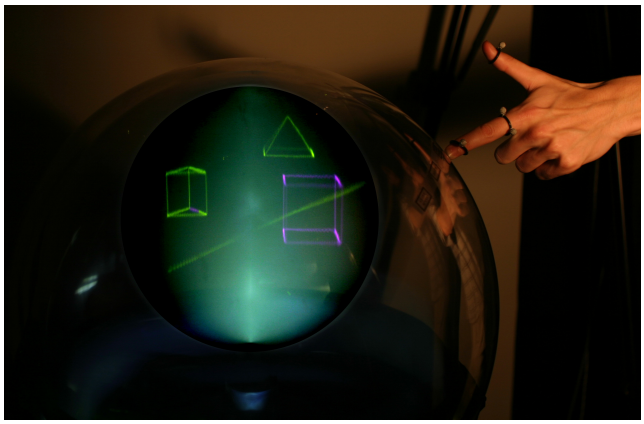

Figure 11. Selection using a ray cursor.

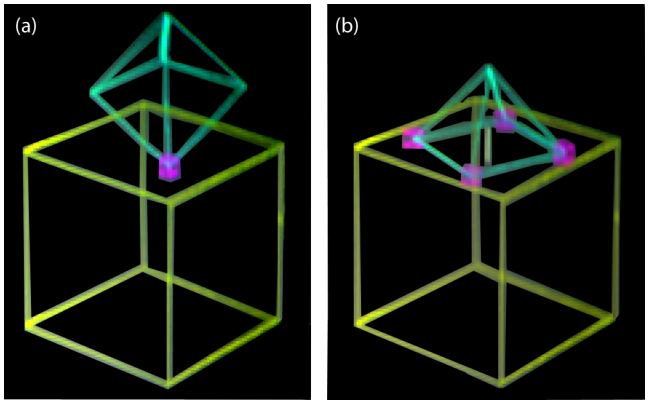

Figure 12. Snapping. (a) Purple marks indicate snapping of a vertex to a face. (b) White tick marks indicate snapping of parallel faces.

Operations on Multiple Models

The transformation techniques for single models described earlier can be used to simultaneously apply transformations to multiple selected models. When more than one model is selected, the centre of rotation, translation, and scaling operations is computed as the average of the centers of all selected models. The relative positions of the selected models remain unchanged by any transformation.

Multiple objects can be grouped together by pressing the "Group/Ungroup" surface menu button when they are selected. Once a group has been formed, selecting or deselecting a model selects or deselects its entire group. Objects can subsequently be ungrouped by pressing the "Group/Ungroup" button while a group is selected.

Single models or groups of models, once selected, can be deleted from a scene by pressing the "Delete" button.

Snapping/Collision

To aid in building complex scenes, the system supports collision detection and snapping between models. By default this is disabled, but by pressing the "Snap" surface menu button, collision detection and snapping are enabled. When translating, objects cannot intersect one another. Instead, the vertices of the moving object will snap to the face of any stationary object which it intersects. Purple marks are displayed to indicate the point at which a vertex snaps to a face (Figure 12a). If the model is then translated to any side of the face, it will snap to the edge. Objects can also snap while rotating. The object will snap such that the normal of any of its faces matches the normal of any nearby face. A small tick mark is displayed through parallel snapped faces to indicate that snapping has occurred. (Figure 12b).

DISCUSSION

In our research, we uncovered several interesting principles and issues unique to volumetric displays:

- True 3D displays remove a layer of abstraction between input and display space, and thus tend to better afford gestural interactions.
- Because the display space is limited by the physical enclosure, all objects are within arm's reach. As a result, traditional 3D interaction techniques don't necessarily apply, necessitating the development of new techniques.
- Given that the display area is within an enclosure, gestures on and above its surface can be quite directly mapped to actions within.
- Multiple users can view the scene simultaneously while maintaining context of their shared physical surroundings, allowing for richer multi-user experiences.
- With current display technology, head-tracking would be required for hidden surface removal with one user, and impossible with multiple users.

The interaction techniques we developed make maximal use of the three dimensional nature of the display and input system. While the direct finger tracking allows users to perform high degree of freedom operations with multiple fingers, the display technology allows users to accurately visualize the virtual 3D manipulations with excellent depth perception. Taken together, this allows for interesting interactions not possible in traditional input and display combinations. For example, free-form transformations allow for simultaneous three degree of freedom translation and rotation, allowing quick and accurate object placement in a 3D environment without the need for constant viewpoint rotations. Combining this with snapping and collision detection, users can quickly build-up 3D scenes from a set of primary models (e.g., Figure 13). To improve precision, users can quickly define arbitrary constraint axes. Not only can the constraint axis be defined from any viewpoint, but it can also be adequately visualized, without the need for adjusting a camera position as would be required in traditional 3D virtual environments.

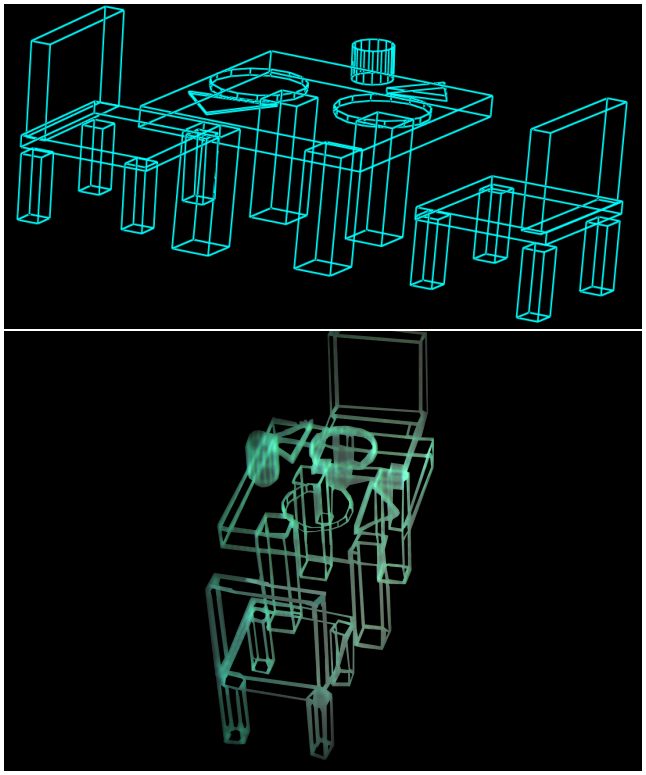

Figure 13. A "table setting" scene built using our system. (top) The scene rendered in high quality to show the precise composition of its parts. (bottom) Photograph of the scene as it is displayed in the volumetric display.

While some of the interaction techniques which we implemented have been adapted from research on virtual reality systems, others have been designed specifically for volumetric displays. For example, the use of a ray cursor has been demonstrated previously, but we extended the technique to using the index finger to directly control the ray's orientation and position as well as using the thumb to perform a trigger gesture to confirm selections. Similarly, we extended previous work on virtual rotations to a new technique that enables precise rotations about all three axes simultaneously by moving the fingers directly on the surface of the display.

While the model manipulation techniques make use of up to three degrees of freedom of input, our mechanism for selecting between models and scenes, the SurfaceBrowser, is constrained to the inner surface of the volumetric display. This, in effect, creates a two dimensional viewing plane, wrapped around the inside surface. We deliberately designed the SurfaceBrowser in this manner, rather than making use of the full 3D display volume, because it frees up the internal display area for simultaneously viewing models selected from the SurfaceBrowser. Thus, a user is able to view their current work area and make selections from a menu that affects that area, preserving context. Because our volumetric display has a non-uniform gap between the surface and the rendered 3D volumetric image, we felt it was necessary to add a cursor icon to give the user feedback as to where their finger position was being mapped into the surface browser. As volumetric display technology improves, this gap will quite likely be reduced, and the cursor could be removed from the SurfaceBrowser, resulting in a sense of even more direct interaction.

It is important to note that although the techniques we developed were presented in the context of a 3D model builder, they are equally applicable to any interactive 3D application which uses a volumetric display. In a sense, our techniques are 3D analogues of the standard WIMP techniques used on 2D desktops that work reasonably well for a broad range of 2D applications. For example, the SurfaceBrowser can be thought of as a standard file browser for managing any kind of 3D, or even 2D, data on a volumetric display.

CONCLUSIONS and FUTURE DIRECTIONS

We have presented a suite of gestural interaction techniques for use with a 3D volumetric display. To allow for high fidelity direct user input, we demonstrated the use of a real-time motion capture system to simulate a touch-sensitive display surface, detect hover just over the surface of the display, and track user hand positions when away from the display. We believe that this is an interesting use of motion tracking systems, departing from its traditional use in offline motion capture to provide movement data for animation, games, and human movement analysis. While the techniques presented here can form the basis for highly interactive use of such displays, there are clearly many more interesting research challenges that remain to be explored.

With improvements to the display technology, it would likely be possible to display surfaced and textured models. When such rendering is available it would be of interest to explore more advanced object manipulations. For example, our existing techniques for directly interacting with the display surface could be built upon to perform various sculpting operations. Manipulations used when working with physical clay such as pulling, pushing, squeezing, and stretching could be adapted to gestural interaction on the display surface to perform various surface deformations on virtual models.

To allow for such advanced gestures, it may be useful to move from tracking the thumb and two index fingers which we implemented in our system, to full tracking of all fingers and the palm of both hands. This would clearly increase the possible set of postures and gestures. However there would be an obvious trade off. The small and simple gesture set which we developed allowed for a simple fluid interface requiring relatively little prior training of the few users who have tried the system. Increasing the complexity of the gesture set would require explicit mechanisms to reveal and teach novice users about the various possible interface actions. We plan to conduct more formal usability testing of our gesture set and interaction techniques in the near future.

Lastly, the display's 360° field of view makes it an obvious platform for exploring collaborative multi-user interaction in 3D environments. Given that users would not need to wear head mounted displays or special glasses, they would be able to view the three dimensional data while maintaining the context of their surrounding environment and other users, facilitating human-human communication in conjunction with human-computer interaction. However, because the work area is a shared three-dimensional display, unique issues arise when multiple users attempt to share the space. For example, at what orientation should text be displayed? What kinds of strategies can be applied to allow two or more people to work together using gestural interaction techniques on such a display? The rich literature on collaborative computing will provide guidance, but these prior ideas will have to be adapted and refined for appropriate use with this new display technology, a challenge we believe is worthy of further exploration in the future.

ACKNOWLEDGEMENTS

We thank John Hancock for technical support, Daniel Vogel and Maya Przybylski for help with images and video production, members of the Dynamic Graphics Project lab (www.dgp.toronto.edu) at the University of Toronto for valuable discussions, and Actuality Systems for technical maintenance of the display.

REFERENCES

1. Balakrishnan, R., Fitzmaurice, G., & Kurtenbach, G. (2001). User interfaces for volumetric displays. *IEEE Computer*. March 2001. p. 37-45.
2. Baudel, T., & Beaudoin-Lafon, M. (1993). Charade: remote control of objects using free-hand gestures. *Communications of the ACM*. 36(7). p. 28-35.
3. Bowman, D., & Hodges, L. (1997). An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. *ACM Symposium on Interactive 3D Graphics*.
4. Bowman, D.A., Johnson, D.B., & Hodges, L.F. (1999). Testbed environment of virtual environment interaction. *ACM VRST'99 Symposium on Virtual Reality Software and Technologies*. p. 26-33.
5. Buxton, W., & Fitzmaurice, G.W. (1998). HMD's, Caves, & Chameleon: A human-centric analysis of interaction in virtual space. . p. 64-68.
6. Conner, B., Snibbe, S.S., Herndon, K.P., Robbins, D., Zeleznik, R., & Dam, A.v. (1992). Three dimensional widgets. *Computer Graphics*. 22(4). p. 121-129.
7. Ebert, D., Bedwell, E., Maher, S., Smoliar, L., & Downing, E. (1999). Realizing 3D visualization using crossed-beam volumetric displays. *Communications of the ACM*. 42(8). p. 101-107.
8. Grossman, T., & Balakrishnan, R. (2004 - in press). Pointing at trivariate targets in 3D environments. *ACM CHI Conference on Human Factors in Computing Systems*.
9. Hinckley, K., Pausch, R., Goble, J.C., & Kassell, N. (1994). A survey of design issues in spatial input. *ACM UIST 1994 Symposium on User Interface Software and Technology*. p. 213-222.
10. Kurtenbach, G., Fitzmaurice, G., Baudel, T., & Buxton, W. (1997). The design of a GUI paradigm based on tablets, two-hands, and transparency. *ACM CHI 1997 Conference on Human Factors in Computing Systems*. p. 35-42.
11. Liang, J., & Green, M. (1994). JDCAD: A highly interactive 3D modelingsystem. *Computers and Graphics*. 18(4). p. 499-506.
12. Lucente, M. (1997). Interactive three-dimensional holographic displays: seeing the future in depth. *Computer Graphics issue on Current, New, and Emerging Display Systems*. May 1997.
13. McCauley, M.E., & Sharkey, T.J. (1992). Cybersickness: Perception of self-motion in virtual environments. *Presence*. 1(3). p. 311-318.
14. Mine, M. (1995). ISAAC: A virtual environment tool for the interactive construction of virtual worlds. UNC Chapel Hill Computer Science Technical Report TR95-020.
15. Mine, M. (1995). Virtual environment interaction techniques. UNC Chapel Hill Computer Science Technical Report TR95-018.
16. Patterson, R., & Martin, W. Human Stereopsis. *Human Factors*. 34(6). p. 669-692.
17. Poupyrev, I., Billinghurst, M., Weghorst, S., & Ichikawa, T. (1996). The go-go interaction technique: non-linear mapping for direct manipulation in VR. *ACM UIST Symposium on User Interface Software and Technology*.
18. Ware, C., & Balakrishnan, R. (1994). Reaching for objects in VR displays: Lag and frame rate. *ACM Transactions on Computer-Human Interaction*. 1(4). p. 331-356.
19. Ware, C., & Lowther, K. (1997). Selection using a one-eyed cursor in a fish-tank VR environment. *ACM Transactions on Computer Human Interaction*. 4(4). p. 309-322.
20. Zhai, S. (1995). Human performance in six degree-of-freedom input control. Ph.D. thesis. University of Toronto.