Making VR Work: Building a Real-World Immersive Modeling Application in the Virtual World

Mark Mine

Walt Disney Imagineering
1401 Flower Street
Glendale, CA 91201
mark.mine@disney.com

Arun Yoganandan

Walt Disney Imagineering
1401 Flower Street
Glendale, CA 91201
ayogan2@gmail.com

Dane Coffey

Walt Disney Imagineering
1401 Flower Street
Glendale, CA 91201
dcoffey86@gmail.com

ABSTRACT

Building a real-world immersive 3D modeling application is hard. In spite of the many supposed advantages of working in the virtual world, users quickly tire of waving their arms about and the resulting models remain simplistic at best. The dream of creation at the speed of thought has largely remained unfulfilled due to numerous factors such as the lack of suitable menu and system controls, inability to perform precise manipulations, lack of numeric input, challenges with ergonomics, and difficulties with maintaining user focus and preserving immersion. The focus of our research is on the building of virtual world applications that can go beyond the demo and can be used to do real-world work. The goal is to develop interaction techniques that support the richness and complexity required to build complex 3D models, yet minimize expenditure of user energy and maximize user comfort. We present an approach that combines the natural and intuitive power of VR interaction, the precision and control of 2D touch surfaces, and the richness of a commercial modeling package. We also discuss the benefits of collocating 2D touch with 3D bimanual spatial input, the challenges in designing a custom controller targeted at achieving the same, and the new avenues that this collocation creates.

Author Keywords

Virtual Reality; 3D user interaction; Selection; Graphical User Interfaces; Controller design; Immersive 3D modeling.

ACM Classification Keywords

H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems – Artificial, augmented and virtual realities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
SUI '14, October 04 - 05 2014, Honolulu, HI, USA
Copyright 2014 ACM 978-1-4503-2820-3/14/10...\$15.00.
<http://dx.doi.org/10.1145/2659766.2659780>

INTRODUCTION

From making cars to designing theme park rides, pre-visualization and virtual prototyping have become an integral part of the design process. Commercial-off-the-shelf (COTS) desktop modeling tools like Maya and SketchUp are extensively used in such processes to design and develop 3D assets. Unfortunately, when using these tools, a wealth of spatial information is unavailable to the designer since they are constrained to viewing and interacting through a 2D window. Conversely, there is compelling evidence that tracked 3D interaction offers a much more intuitive and quicker way to work in 3D [18], but unfortunately most immersive modeling applications are built from scratch and do not provide the wealth of tools and plugins that COTS applications provide. In this paper, we share our experiences trying to bring those two worlds together by converting the COTS application SketchUp into a virtual reality application: VR SketchUp. Beyond that, we

Figure 1. SketchUp VR Interface
Top: Floating VR GUI
Lower left: Non-dominant Hand Controller (ND-HC),
Lower right: Dominant Hand Controller (D-HC)

endeavor to develop interaction techniques that can run across a spectrum of displays, ranging from the desktop, to head-mounted displays to large CAVE environments, while responsively adapting to the unique input mechanisms and interaction styles afforded by each. That way, designers can switch between these displays based on their current stage in the process such as design, review, showcase etc., but still retain the same familiar tools and capabilities across them. We use synchronous and asynchronous bimanual input for view, object and menu manipulation.

We also share lessons learned from our efforts to build expressive, easy to use, and ergonomic input devices. Immersive modeling tools from the past have mostly relied on tracked wands and game controllers with only physical inputs such as buttons, triggers and joystick. This can both limit the expressiveness of user interaction due to the simplistic nature of these types of inputs while simultaneously complicating the interface due to the large number of inputs and the need for the user to remember complex functional mappings. We detail here our approach in building a hybrid controller that collocates a touch display, a casing with physical buttons, and 6 degrees of freedom (DoF) tracking to enable a dynamic and rich medium of interaction. The presence of two touch screens and floating virtual menus (Fig 1) provide a great avenue to perform much of the system control and menu navigation without dividing attention between the displays, and we share our design practices for the same. Furthermore, we have found that collocating spatial tracking and touch on two hands enables opportunities for novel tools and new forms of interaction. We describe several new widgets for object manipulation that were developed from this opportunity.

RELATED WORK

Immersive 3D modeling has been of interest for almost two decades now. Butterworth et al. [1] developed a 3D modeling system for head-mounted displays, with a 6-DoF 2-button mouse. Mine [11] explored scene composition within Virtual Environments (VE) with ISAAC. Hughes et al. [6] developed CaveCAD, an application to build architectural models with 3D interaction inside a CAVE environment. Ponto et al. [14] developed a system for freeform virtual sculpting of organic shapes with 3D interaction.

Multigen [15] pioneered the SmartScene modeling system that enabled fully immersive modeling using two tracked hands. Jerald et al. [7] also demonstrated a two-handed 3D modeling system with magnetically tracked controllers. There has been evidence of interest in using commercial-off-the-shelf applications within VEs. Takala et al. [22] adapted Blender, an open-source desktop modeling application by incorporating 3D interactions into it with the PlayStation Move controllers. The immersive desktop holographic system - ZSpace [24] has enabled support for Maya.

Menu selection and navigation are important components of an immersive modeling tool to facilitate quick and efficient workflows. Ni et al. [13] developed the rapMenu system that used tracked pinch gloves to interact with hierarchical pie menus. Menu selection with 6-DoF magnetically tracked controllers was examined by [16] with the rAirFlow menus. Many researchers have investigated the use of touch screens for menuing within VEs in the past. Medeiros et al. [10] explored using a tracked tablet for view manipulation, object selection and manipulation and for symbolic input in a virtual engineering environment. The application required having the tablet lifted up to perform object manipulation. Wang et al. [23] used a tablet strapped to the arm for menu and attribute selection. A 3D tracked controller on the other hand controlled navigation. Gebhardt et al. [5] studied the usage of smartphones for system control within VE. Their menus were represented only on the smartphone and not within the VE.

Researchers both within and outside of Virtual Reality have investigated combining spatial tracking and touch. Marquardt et al. [9] defined the continuous interaction space between 2D touch surfaces and the 3D tracked spaces above them. De Araujo et al. [4] expanded on [9] to create an intuitive bimanual application that combined on and above surface interactions along with gaze and posture to facilitate creation of 3D models. Additionally, they also segregated interactions between hands, with system control and menu selection performed on the non-dominant (ND) hand and sketching or spatial modeling gestures on the dominant (D) hand. Coffey et al. [3] examined the use of multi-touch surfaces as an input device for manipulating view and widgets in a VE on a distant display. Controlling virtual widgets with a tracked, unseen touch screen was explored by Steed et al. [21]. Seifert et al. [19] extensively investigated using smartphones as pointers to large displays. Song et al. [20] defined the concept of a handle bar metaphor to manipulate virtual objects at a distance. Rashid et al. [17] studied the effectiveness of mobile devices for proximal and distal selection tasks on a remote large display. They found touch interactions on mobile devices to be easier when many widgets needed to be modified, but introduced the issue of attention switching between the phone and the large display. In this paper, we expand on research reported in our previous work [12].

SYSTEM

Hardware

Our system has been developed and used in 3 environments:

- The DISH - a 360° CAVE environment with 5 stereo projected surfaces – 4 walls and a floor.
- The Wall – A stereo rear-projected curved display wall.
- A regular desktop without stereo rendering

Phasespace Inc's optical and sensor fusion tracking systems were used for 6-DoF head and hand tracking within the

large environments. A Polhemus Fastrak magnetic system was used on the desktop version. Hybrid input devices known as Hand Controllers (HC) were designed and developed in-house. They each consist of a Samsung Galaxy S3 smartphone, an Arduino Nano microcontroller, and 3 physical buttons, all connected with each other and housed in a custom casing that was 3D printed. The physical buttons are reserved for fundamental actions or events that remain consistent throughout the application such as object manipulation, view manipulation and menu reset. The controller design was iterated for ergonomic comfort and for prolonged usage.

Software

TechViz, an OpenGL intercept application and cluster renderer like [2] was used to adapt SketchUp for VR displays. SketchUp runs a custom plugin that manages the tracking information, communicates with the TechViz API to update graphical transformations, emulates the mouse and keyboard inputs, provides 3D representations of hands, generates floating menu interfaces, and manages the state of the system. A modified version of Mape's Two Handed Interface (THI) [8] was implemented to achieve bimanual view manipulation. The HCs run custom Android applications that manage touch and button presses. Thus SketchUp and HCs communicate with each other over Wi-Fi to update the state of the system and adapt dynamically.

VR SKETCHUP

There are several benefits that can be achieved from adapting SketchUp into VEs. Head tracking provides a quick and natural way of making minor viewpoint changes to look around a model to study and absorb it, resulting in a better understanding of the 3D space. With the combination of head and hand tracking, users can manipulate both the object and the viewpoint at the same time. Direct 6-DoF interactions enable intuitive 3D manipulation of objects in the scene, and a bimanual system allows the user to take great advantage of proprioception for the interactions.

These benefits, however, come at a cost. There are several challenges associated with immersive 3D modeling specifically, and all immersive applications in general, the primary of which is system control. The lack of the familiar WIMP interface within VR environments makes it difficult to navigate about menus and tool options. Additionally, the absence of the familiar mouse and keyboard only serves to deepen this issue. As a result, even operations that do not require 3D interaction such as precise numeric input, textural input and symbolic input are often achieved with complicated 3D widgets, resulting in a cumbersome and unintuitive user experience. Finally, even though spatial interaction offers an intuitive mechanism for design, their control is still restricted to coarse interactions. Thus precise alignment and positioning in a VE still remain as a largely unsolved problem.

In addition to the above, one of the main challenges in adapting COTS applications to VEs arises from the fact that

Figure 2. Hand Controller

these applications are not designed or meant to be used in VEs. Very quickly, VR developers start to face limitations with respect to rendering, interactions and the software architecture. For example, SketchUp does not have a way to classify 3D objects as either scene objects or 3D GUI elements. As a result, simple operations such as picking and ray collisions end up acting on these GUI elements unintentionally. Also, native tools for such applications are usually closed source and are not available for the developer to extend or customize for VR. As a result, mouse emulation is resorted to in native tools, where spatial interaction could have very well made a big difference.

Hand Controllers

As mentioned in the Introduction, the challenges associated with using traditional game controllers such as the Nintendo Wii and the Razor Hydra as input devices for immersive modeling applications motivated us to develop custom hand-held controllers that combine a smartphone, full 6-DoF tracking and a casing with physical buttons (Fig 2).

We incorporated touch screens because they provide a rich dynamic interface that allows for higher bandwidth of input and output. In addition, they support both unseen touch interactions and the more traditional visual widgets for 1D and 2D interaction, informational readouts, numeric input etc. They allow for dynamic affordances, enabling a wide option of widget choices that can be more or less complicated depending on the task at hand. The touch screens also provide tangible steadying surfaces for fine manipulation, which is often an issue in pure spatial interaction.

The smartphone form factor was chosen because users could easily hold one in each hand (allowing for bi-manual interaction) and carry them wherever they went. In the early stages of our project, we experimented with a larger touch tablet for this purpose, but the size of the device and the necessity to hold it with one hand for interacting with the other proved to be limiting. Similarly, we considered a larger pen-based input device mounted on a wheeled platform. This would enable rich forms of interaction, but

we found that the device was never where one wanted it to be when physically moving around in the VE.

We also tested skipping the physical buttons and just relying on the soft buttons for all of our interactions, but the loss of tactile feedback in addition to the attention required to visually locate buttons resulted in reduced focus and immersion. The physical buttons provide consistent access to system features that are tool agnostic and are constantly accessed throughout the application such as view and object manipulation, system control, and modifiers. By tools, we refer to the different class of actions within SketchUp such as the rectangle tool, circle tool etc. The number of buttons was deliberately kept low to minimize user confusion.

On extended use, ergonomics and shape of the device have proven to be critically important. The controllers were designed to be lightweight and to fit comfortably within a single hand. Additionally, since touch interactions have to happen with the thumb of the same hand that holds the HC, design changes were made to ensure that the thumb could reach as much of the touch screen as possible without much strain. Factors such as wrap around radius, angle of the physical button surface, angle at which the HC is being held at neutral position, size and resistance of the button, number of fingers dedicated for physical gripping, distribution of work load between fingers operating on physical buttons were all iterated upon through multiple revisions to arrive at a design that satisfies our needs. The HC still has issues related to unintentional touch screen presses when pushing physical buttons that we are continuing to work on. Other modalities that are available from a smartphone such as voice and gestural input were investigated and found to be more suitable for one-off operations; for more frequently used interactions, they were found to be either ambiguous or to slow down the interactions.

Interaction Design Philosophy

Considering that our system is made up of multiple input and output mechanisms with varying levels of controls, we found it important to build the user experience based on certain design guidelines.

Minimize energy

3D modeling by its nature calls for extended hours of design, iteration and modification. Therefore, conserving energy is important. Even though the user's head and hands are tracked, it is in our interest to design for minimum locomotion and physical exertion. Animated hand gestures could be used for virtual menu selection, but they will soon tire the user out. We restrict such frequent actions to the touch screen, where only small movements of the thumb are used to perform the same selection. Similarly, much of the user's energy is going to be spent in navigating from place to place if a system is designed such that objects can only be manipulated when within arm's reach. Therefore, we lean heavily towards action at a distance and image plane interaction techniques. Additionally, we provide quick

access to most frequently used system controls and tools in order to reduce time spent in menu navigation.

We designed our system to be used in a comfortable seating posture in all display environments. The THI allows users to grab and pull the world towards them, essentially moving their viewpoint forward. The same action when performed alternately with two hands results in a movement that is akin to walking with two hands within the virtual world. That combined with THI's ability to rotate, scale and translate about the user helps reduce the need for physical locomotion. Also, wherever possible 2D touch alternatives are provided for spatial interactions.

Maximize comfort

Issues of ergonomics and comfort play an important role in the success of any immersive modeling system. Due to the inherent limitations of the movement style and reach available for the thumb, the touch interfaces were correspondingly designed to work within these limitations. For example, widgets are typically placed within the area of the touch screen that is comfortably swept by the thumb; placements in regions that require extensive reach are avoided as much as possible. It was also observed that a simultaneous physical button press and touch screen interaction on the same hand resulted in some physical strain; as such these were avoided or used in a limited fashion. Instead, interaction patterns that required simultaneous use of physical and soft buttons were distributed across the hands such that one hand engages the physical button and the other engages the soft buttons.

3D spatial interaction for coarse input

3D spatial interactions are inherently intuitive for 3D tasks. Users do not need to have a mental map that translates 2D interactions to 3D in an awkward manner. On the contrary, spatial interactions are not a good fit for precision operations due to the lack of tangible support surfaces and the existence of superfluous degrees of freedom. We use 3D spatial input to achieve a coarse but intuitive starting step.

2D touch for precision input

2D touch interactions are very well suited for controls such as precision numeric input, textural input etc. We adapt the methodology of performing spatial work to achieve coarse results and then tweaking them with precise interactions on touch surfaces.

Combine modalities and use what works

Every modality has its strengths and weaknesses. Our system is designed to combine multiple modalities in a manner that each modality is only utilized for controls for which they are naturally suited. For example, touch screens are great for selecting colors and textures but are not independently suited for view manipulation due to the differences in the number of degrees of freedom. Overall, depending on the task at hand, it is important to eliminate and avoid superfluous DoFs to achieve consistent mapping between user's interactions and output.

Figure 3. Left: radial widget for setting the angle, Right: Precision modifier set to 15° interval between increments

Constrain yourself

Great control comes with good constraints. The rotary dial in Fig 3 for example has additional controls around the outer edge of the dial that can be used to set the angle of increment and hence the precision of the operation. On the move tool interface (Fig 4), users can depress the record button to record the current vector between the two hands and use either that vector or the closest primary world axis as a constraint along which the object should move. Alternatively, they can use the constraint hot spot (see below) to constrain movement along the X, Y or Z axis.

Preserve immersion (as much as you can)

Preserving immersion and retaining focus and context on the current task at hand is crucial for immersive modeling. Therefore, it is important that the system's interactions divert the attention of the user away from the VE as little as possible. This being one of our top requirements, our system is designed such that menu navigation is represented by floating GUIs in the VE that are invoked by interactions on the touch surface that do not require the user to have to look down at them. Users only need to look at the touch screen for tool specific context menus.

Figure 4. Constraint options menu for the move tool

Figure 5. Region surrounding menu dial broken into zones for quick access to certain menu levels. Corners of screens contain hot spots from where hot keys are evoked as marking menus

Capitalize on proprioception

Human beings are very good at sensing the position of their body and remembering body relative information; we have tried to take advantage of this factor in our application design. For example, the area around the system control dial menu is divided into four hot zones that can be used to invoke frequently used menus (Fig 5). The user's sense of his or her hand position is usually all the user needs to select any one of these hot zones without looking at the touch screens.

Use haptic feedback

Within moments of holding the hand controller, the tactile feedback associated with the sides and the corners help one to uniquely identify and access them without the need for visual cues. To make use of this, our touch surfaces are divided into static and dynamic regions. The static regions lie at the corners of the screen (Fig 5) and remain in place throughout the application. They house hot spots, which when touched invoke marking menus with hot keys and system controls, such that one can access them without having to look down at the touch surface. For example in Fig 5, the user reaches to the right-bottom corner of the touch screen to touch a hot spot and slides across to activate a modifier via the marking menu. The farthest corners are usually not quickly reachable by the thumb and hence hot spots are avoided there. Hot spots provide a clean and intuitive way to add up to 3 modifiers or hot keys per corner without crowding the interface, yet taking full advantage of proprioception and haptic feedback. Similarly, using large touch buttons as modifiers (Fig 4) provides the advantage that once the user learns how to use them they are easy to remember spatially and can often be used without having to look down at the screens.

System Control

As shown in Fig 1, the system can be broken down to a head-tracked VR Display, and two touch-screen-enabled Hand Controllers. The GUI floating in the VE is controlled by touch interactions performed on the Hand Controllers. System interactions can be divided into 2 categories:

Menu Navigation

An important part of any application is the ability to navigate menus and select appropriate tools. In place of the traditional WIMP interface, system menus in VR SketchUp are represented as pie menus floating in the VE.

Users can invoke the menu navigation radial dial (Fig 5) by touching a dedicated hot spot on the top left corner of the touch screen that remains there all through the application. Instead of one tap for invoking the menu navigation page and another to start interaction with the dial, the interactions are programmed such that the user can start the touch on the hot spot and drag their thumb to the center of the screen. By the time their thumb reaches the center, the widgets would have changed to that of menu navigation dial, allowing a tap-drag-circle style interaction.

For menu navigation, touch screens are used as primarily unseen touch surfaces and users can fully rely on the visual feedback from the GUI floating in the VE for selection. The system displays a dial at the center of the non-dominant Hand Controller (ND-HC) touch screen that invokes the pie menu when touched and a large button on the dominant Hand Controller (D-HC) that can be used to either navigate

Figure 6. Multi-level pie menu navigation; scroll to sub-menu option, preview menu level, and tap to enter

to a different level in the menu hierarchy or to confirm tool selection (Fig 6). We find this to be a fast way to navigate, preview and select hierarchical menus while still preserving immersion. Furthermore, the region immediately surrounding the dial is broken down into quick access zones (Fig 5) that have specific menu levels assigned to them previously. When a user starts touch on one of these zones and slides into the dial, they are immediately taken to the assigned menu irrespective of their current state.

While the DISH with its 360° view facilitated placement of visual menu elements in almost any view direction, this was not the case with the more limited displays such as the Wall and the desktop monitor. Therefore, placement of floating GUI was constrained and configured differently based on the display in use.

Context Menus

Upon selecting a particular tool, the touch screens update to display the corresponding context menus that are used to specify and control parameters particular to the active tool. The dynamic nature of touch devices makes it possible to use a wide variety of 2D widgets to perform many different

Figure 7. ND and D screens for texture tool context menus.

tasks such as: texture selection, numeric input, and color selection to name a few (Fig 7).

When a large number of 2D widgets are required to control a particular tool it becomes hard to fit them all in a single touch screen. In this case, the widgets are distributed across the touch screens on the two Hand Controllers. 2D widgets that require involved interaction are as much as possible retained on the D-HC, while the widgets that require coarse selection such as large modifier buttons are assigned to the ND-HC. The context menu for the texture tool (Fig 7) is a good example where the widgets can neatly be split across the screens, with the ND-HC focusing on texture selection and application, while the D-HC focuses on texture manipulation. The split across two screens also enables the simultaneous two-handed control of multiple parameters; the color selection context menu for example, has a hue widget on the ND-HC and a saturation and value widget on the D-HC. All three parameters can be adjusted at once.

In the case of complicated menus such as numerical input, splitting a tool's widgets across multiple screens would result in the user having to constantly shift their attention back and forth between the two Hand Controllers. In these situations, we split a tool's widgets across multiple layers on the D-HC instead (Fig 8). A modifying tap button on the ND-HC is used to switch between the two layers. Given

Figure 8. ND and D screens for multi-layer context menus.

Left: Input layer before pressing modifier button.

Right: Popup layer after pressing modifier button.

that the interactions with the touch screens are limited to the thumb, we find using these tap modifiers on the alternate hand to invoke a popup layer to be much faster and more convenient than swiping the screen with the thumb to go from one page to the other.

Object Manipulation

There are three important ways in which objects can be manipulated within VR SketchUp:

- *Direct 6-DoF Manipulation* Where the user grabs objects within arm's reach and positions and rotates them in a manner similar to the real world. Scaling of the object can be achieved using bi-manual interaction. DoF constraints, rotational axes, and special behavior such as position-only manipulation are specified using the touch-screen interface.
- *Image Plane Interaction* Where movement of the user's hand within their field of view is mapped to screen space interactions. Note that the screen space is defined to be a rectangular region perpendicular to the user's current gaze direction. It roughly corresponds to the SketchUp active modeling window.
- *Track Pad Interaction* Where the user manipulates objects via a touchpad widget on the touch screen to emulate mouse interactions within the user's screen space.

VR SketchUp and the HC are in constant communication with each other, synchronizing information such as hand collision with virtual objects, screen space picking information and touch activity on the track pad. Using these, the application can identify which object manipulation mode the user is interested in working in and transition to it automatically.

In large virtual worlds, scene objects are often dispersed at various distances and directions. Physically or virtually moving to within arm's reach of an object every time one needs to interact with it is a tiring experience and often not practical for extended hours of interaction. Also, it is sometimes important to tweak an object's position and orientation while still needing to have the rest of the scene or other objects in context, making physical or virtual proximity to the object not an option. Therefore, image plane and other action-at-a-distance interaction techniques are essential for effective object manipulation. It is to be noted that our system leverages the benefits of mouse style input with image plane interactions but adds spatial input on top of that for additional expressiveness and control. World In Miniature style input was considered and experimented with, but proved to be less elegant to implement due to SketchUp's architectural limitations.

SUI Tools

We developed several action-at-a-distance tools for the translation, rotation, and spatial arrangement of objects within the VE that take advantage of the colocation of 2D

touch with 3D spatial user interaction (SUI) afforded by the application. We categorize them as SUI tools and we detail them here.

Action Plane Widget

All of our SUI tools depend on a custom 3D widget called the Action Plane Widget (APW). The APW widget allows users to define the coordinate space in which the selected object is to be translated and/or rotated. Visually, the APW (Fig 9) consists of a 2D plane represented by a grid and a normal to the plane. The Center of Action (CoA) of the APW is defined to be the root of the plane's normal vector. The CoA as its name suggests, is the point about which all transformations are intended to happen. The user interacts with the APW using a combination of 6-DoF and image plane interactions with touch as a modifier.

To position the CoA the user pushes a dedicated physical button on the ND-HC and then uses 6-DoF or image plane interaction techniques to specify the exact location of the CoA within the scene. SketchUp's picking and inference mechanisms make it easy for users to snap the CoA to features of interest in the scene.

Figure 9. Top: APW normal definition via bimanual spatial input, Middle: SUI translation, Bottom: SUI Rotation

Figure 10. ND and D screens for SUI Translation tool

The orientation of the APW is set when the user presses the “Manipulate Orientation” button on the ND-HC touch screen (Fig 10, 11 Left) and is defined by the vector between the user’s two hands. (Fig 9 Top) The APW’s orientation can be set freely or can be aligned relative to scene features such as a primary world axis, a primary object axis or any other feature in the scene. Radio buttons on the touch interface (Fig 11 Left) are used to control which types of inferences are currently active.

Using these techniques, one can define the APW transformation space in a very detailed manner, for example: CoA at the corner of one object in the scene, normal aligned to the edge of a different object; alternately, CoA at the corner of one object, normal aligned to a primary world axis and so on.

The CoA can also be manipulated using track pad interaction on the ND-HC (Fig 10, 11 Left). Interaction on the track pad moves the CoA correspondingly in the user’s screen space. While using the track pad to place CoA within the user’s screen space, the jitter from head tracking can significantly reduce the effectiveness of placement. We deal with this problem by temporarily suspending the update of screen space pose the moment the user touches the track pad. Note that this only suspends update of the screen-space interaction plane pose but does not suspend normal head tracking. If the user does not significantly stray from the pose of the head at the start of interaction this doesn’t cause any problems. The tracking picks up again when the user’s finger releases from the track pad’s surface or when the user strays beyond a threshold.

SUI Manipulation general strategy

To use the SUI tools one must first indicate the object to be manipulated using image plane interaction and select it using the physical action button on the D-HC.

Then, if necessary, the pose of the APW can be modified using the techniques described above, or the user can continue with the APW in its existing pose.

Figure 11. ND and D screens for SUI Rotation tool

Once the pose of the APW is satisfactory, the user can employ either image plane interactions or the touch screen widgets (described below) to perform the desired manipulation. Typically, users begin with image plane interaction for coarse manipulation until the object is almost at the desired transformation and then switch to touch interaction to perform fine grain manipulations and adjustments until the precise effect is achieved

To ensure a consistent experience using the SUI tools we adopted the following strategy:

- Manipulation of the APW is performed by interactions initiated on the ND-HC
- Real object manipulation is performed by interaction initiated on the D-HC

SUI Translation tool

Translation of the object happens either on the APW’s plane or in another plane parallel to it. Translation can also happen along the normal to the APW plane. Image plane interaction for translation is relative as well, the distance between the start and end points of the image plane intersection point on the APW grid determines the distance by which the selected object is moved (Fig 9 Middle). The “Translate Along Normal” modifier (Fig 10 Left) can be depressed to make the object move along the normal direction to the APW plane. Projection of the vector from the user’s head to his or her dominant hand onto the APW’s normal defines the amount of vertical traversal of the object. Users can switch back and forth between translation along the plane and traversal in the direction of the normal seamlessly by simply depressing and releasing the Translate Along Normal modifier without breaking their interaction.

Additionally, users can also use the large 2D (Fig 10 Right) track pad widget on the D-HC to translate the object along the plane with respect to the user’s screen space. We have found that differences in head orientation and the orientation with which users hold the touch screen can

Figure 12. D screen for SUI spatial array tool

affect their perception of what is up-down-left-right and will on occasion provide less than satisfactory results.

SUI Rotation tool

Rotation of the object is relative to the APW normal. Vectors from the start and end points of the image plane intersection point on the APW grid with respect to the CoA define the angle of rotation (Fig 9 Bottom). The selected object is correspondingly rotated by the same amount with respect to the CoA. Similarly the radial menu on the D-HC touch screen (Fig 11 Right) can also be used to perform the same action.

SUI Spatial Array tool

Copying objects to form a 1D, 2D or a 3D array is a task that benefits greatly from a hybrid touch-SUI interface. As with the above tools, the user can first pose the APW about the point of interest. Then he or she can use the widgets on screen (Fig 12) to define the number of axes the array will have and the divisions on each of those axes. The user can then press the soft button that corresponds to the axis of interest and move his or her hand in space to redefine the direction of the axis while simultaneously pre-visualizing the arrangement of the copies (Fig 13). Once satisfied with the layout, pressing the confirm button actually creates the array.

Figure 13. SUI Spatial Array tool. Left: Set first axis and divisions. Middle: Set second axis and division. Right: Resulting array

CONCLUSIONS AND FUTURE WORK

In this paper we have presented some of the experiences we have had and the lessons we have learned in converting a commercial-of-the-shelf modeling application, SketchUp, to work in a virtual world. Though we have yet to achieve our goal of a real-world modeling application, based upon our preliminary results (Fig 14) we strongly feel that we are moving in the right direction. The combination of our philosophy of minimizing energy while maximizing comfort; our careful mapping of actions across 2D and 3D interactions; and the power, flexibility, and expressiveness of our bimanual touch-based Hand Controllers shows great promise for creating effective immersive modeling environments in the future.

Depending on the size of the user's hands, we observed that reaching the opposite corners of the touch screens proved difficult at times. We hope to alleviate this issue by using smaller smartphones. We also suffer from noticeable latency in button presses, since the communication is happening across Wi-Fi. We would like to experiment with other low latency wireless communication methods to resolve this issue.

We feel that we have just begun to explore the potential of the collocation of 2D touch with 3D spatial input. We continue to explore potential tools and interaction techniques that exploit their coexistence to the fullest. We continue to refine the design of our Hand Controllers, working to minimize user strain and maximize long-term use. We would like to further explore using tactile cues on the screen surface such as dots or thin transparent frames that can further our spatial understanding of our touch screens and help with unseen touch interactions.

REFERENCES

1. Butterworth, J., Davidson, A., Hench, S., & Olano, M. T. (1992, June). 3DM: A three dimensional modeler using a head-mounted display. In *Proceedings of the 1992 Symposium on Interactive 3D graphics* (pp. 135-138). ACM.
2. <http://chromium.sourceforge.net/>
3. Coffey, D., Malbraaten, N., Le, T. B., Borazjani, I.,

Figure 14. Models created with VR SketchUp

- Sotiropoulos, F., Erdman, A. G., & Keefe, D. F. (2012). Interactive slice wim: Navigating and interrogating volume data sets using a multisurface, multitouch VR interface. *Visualization and Computer Graphics, IEEE Transactions on*, 18(10), 1614-1626.
4. De Araújo, B. R., Casiez, G., & Jorge, J. A. (2012, May). Mockup builder: direct 3D modeling on and above the surface in a continuous interaction space. In *Proceedings of Graphics Interface 2012* (pp. 173-180). Canadian Information Processing Society.
5. Gebhardt, S., Pick, S., Oster, T., Hentschel, B., & Kuhlen, T. (2014, March). An evaluation of a smart-phone-based menu system for immersive virtual environments. In *3D User Interfaces (3DUI), 2014 IEEE Symposium on* (pp. 31-34). IEEE.
6. Hughes, C. E., Zhang, L., Schulze, J. P., Edelstein, E., & Macagno CAVE. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on* (pp. 193-194). IEEE. (2013, March). CaveCAD: Architectural design in the
7. Jerald, J., Mlyniec, P., Yoganandan, A., Rubin, A., Paullus, D., Solotko, S., MakeVR: A 3D World-Building Interface. 3DUI 2013
8. Mapes, D. P., & Moshell, J. M. (1995). A two-handed interface for object manipulation in virtual environments. *Presence: Teleoperators and Virtual Environments*, 4(4), 403-416.
9. Marquardt, N., Jota, R., Greenberg, S., & Jorge, J. A. (2011). The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. In *Human-Computer Interaction-INTERACT 2011* (pp. 461-476). Springer Berlin Heidelberg.
10. Medeiros, D., Teixeira, L., Carvalho, F., Santos, I., and Raposo, A. 2013. A tablet-based 3D interaction tool for virtual engineering environments. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI '13)*. ACM, New York, NY, USA, 211-218
11. Mine, M., "ISAAC: a meta-CAD system for virtual environments." *Computer-Aided Design* 29.8 (1997): 547-553.
12. Mine, M., and Yoganandan, A., "Building (Disney) Castles in the Air", *Proceedings of Interactive Surfaces for Interaction with Stereoscopic 3D (ISIS3D)*, 2013.
13. Ni, T., McMahan, R. P., & Bowman, D. A. (2008). Tech-note: rapMenu: Remote menu selection using freehand gestural input. In *3D User Interfaces, 2008. 3DUI 2008. IEEE Symposium on* (pp. 55-58). IEEE.
14. Ponto, K., Tredinnick, R., Bartholomew, A., Roy, C., Szafir, D., Greenheck, D., & Kohlmann, J. (2013, March). SculptUp: A rapid, immersive 3D modeling environment. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on* (pp. 199-200). IEEE.
15. <http://www.presagis.com/>
16. Rado, D., & Keefe, D. F. (2009, August). rAir flow menus: toward reliable 3D gestural input for radial marking menus. In *SIGGRAPH'09: Posters* (p. 26). ACM
17. Rashid, U., Kauko, J., Häkikilä, J., & Quigley, A. (2011, August). Proximal and distal selection of widgets: designing distributed UI for mobile interaction with large display. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (pp. 495-498). ACM.
18. Schultheis, U., Jerald, J., Toledo, F., Yoganandan, A., & Mlyniec, P. (2012, March). Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3D tasks. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on* (pp. 117-124). IEEE.
19. Seifert, J., Bayer, A., & Rukzio, E. (2013). PointerPhone: Using Mobile Phones for Direct Pointing Interactions with Remote Displays. In *Human-Computer Interaction-INTERACT 2013* (pp. 18-35). Springer Berlin Heidelberg.
20. Song, P., Goh, W. B., Hutama, W., Fu, C. W., & Liu, X. (2012, May). A handle bar metaphor for virtual object manipulation with mid-air interaction. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (pp. 1297-1306). ACM.
21. Steed, A., & Julier, S. (2013, March). Design and implementation of an immersive virtual reality system based on a smartphone platform. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on* (pp. 43-46). IEEE.
22. Takala, T. M., Makarainen, M., & Hamalainen, P. (2013, March). Immersive 3D modeling with Blender and off-the-shelf hardware. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on* (pp. 191-192). IEEE.
23. Wang, J., Leach, O., & Lindeman, R. W. (2013, March). DIY World Builder: An immersive level-editing system. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on* (pp. 195-196). IEEE.
24. <http://zspace.com>