Plane, Ray, and Point: Enabling Precise Spatial Manipulations with Shape Constraints

Devamardeep Hayatpur[1] **Seongkook Heo**[1] **Haijun Xia**[1] **Wolfgang Stuerzlinger**[2] **Daniel Wigdor**[1]

[1]University of Toronto
{hayatpur|seongkook|haijunxia|daniel}@dgp.toronto.edu

[2]SIAT, Simon Fraser University
w.s@sfu.ca

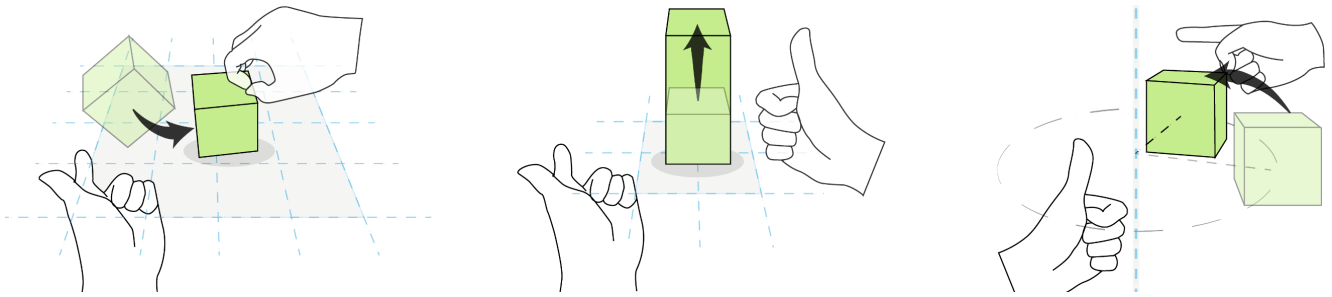

Figure 1. A user can create shape constraints such as Plane (left) or Ray (right) through gestures with their non-dominant hand. The same gestures can be used by the dominant hand to control the manipulation degrees of freedom (middle, right).

ABSTRACT

We present *Plane, Ray, and Point*, a set of interaction techniques that utilizes shape constraints to enable quick and precise object alignment and manipulation in virtual reality. Users create the three types of shape constraints, Plane, Ray, and Point, by using symbolic gestures. The shape constraints are used like scaffoldings and limit and guide the movement of virtual objects that collide or intersect with them. The same set of gestures can be performed with the other hand, which allow users to further control the degrees of freedom for precise and constrained manipulation. The combination of shape constraints and bimanual gestures yield a rich set of interaction techniques to support object transformation. An exploratory study conducted with 3D design experts and novice users found the techniques to be useful in 3D scene design workflows and easy to learn and use.

Author Keywords

3D object manipulation, precise object manipulation, constraints separation, shape gestures.

CSS Concepts

• **Human-centered computing~Gestural input;** *Virtual Reality*; User studies.

UIST'19, October 20–23, 2019, New Orleans, LA, USA.

ACM ISBN 978-1-4503-6816-2/19/10...$15.00.
DOI: http://dx.doi.org/10.1145/3332165.3347916

INTRODUCTION

Commercial implementations of Virtual Reality (VR) systems typically provide rich and immersive visual experiences in 3D environments controlled with 6 degrees of freedom (DOF) (3 position and 3 orientation) input devices, also known as controllers. Many content creation applications, such as those used for 3D modeling [5], game development [38], automotive design [39] and film production [37], are created to take advantage of the rich representation of the virtual world and 6-DOF control. 6-DOF input enables direct object manipulation to quickly and easily grab, move, and rotate objects [20, 32].

In typical HCI, precise manipulations tasks, such as object alignment, benefit from allowing users to manipulate a reduced set of the degrees of freedom, such as 1-D resize handles in generic window managers. The same is true in tasks suited to VR, where a 3D scene designer may wish to rotate a spotlight about a single axis or add a door to a scene by aligning it against a door frame and then rotating it 'open'. Similarly, a designer may wish to scale an object along a single axis, such as making a tree taller, while not moving or rotating it. For such tasks, isolating the available degrees of freedom using visual widgets can make object manipulation more precise [11, 20, 21]. However, visual widgets are slower at completing complex tasks [20].

To address this, we present a set of VR-based interaction techniques termed *Plane, Ray, and Point*, which use shape manipulation constraints to enable varying levels of DOF separation and object alignment (Figure 1). The shapes are created using expressive hand gestures. As an example, a user can constrain the translation or rotation of an object to 1-DOF using a Ray shape, which is created by stretching her index finger or thumb out, and then manipulating the

object using her other hand. By extending both the thumb and the index finger at the same time, the user creates a Plane. Just like with a physical plane, it prevents objects from passing through. If the user is not stretching their index or thumb out, i.e., their hand is closed, they can create a Point; which acts as a pivot, allowing them to change the distance between an object and the Point, as well as rotate the object around that Point. In addition, these shape constraints offer different manipulations for objects that are intersecting with them. For instance, if an object is intersecting with the Plane, then the user can stretch the object out of the Plane for 1D scaling.

This work describes the design and implementation of *Plane, Ray, and Point*. We report the results of a user evaluation where we interviewed six expert users as well as three novices. This work thus contributes the following: (1) a novel interface for DOF separation and object alignment and manipulation; (2) a qualitative exploratory study with six expert users and three novices which shows that the interactions are easy to activate and learn and are also applicable for real world use.

REALATED WORK

Many interaction techniques have been developed to enable more efficient object manipulation. In a recent survey, Mendes et. al. provided an extensive summary of 3D object manipulation techniques [19]. Here, we review the body of literature that has studied direct object manipulation and precise object manipulation using DOF separation.

Direct Object Manipulation

Direct manipulation using the hands enables one to interact with virtual objects similar to the way one interacts with real objects in the world. This affords benefits such as an increased ease of learning and use, and the ability to simultaneously perform complex modifications in a single operation [20, 32].

Extensive work has explored the benefits and novel interaction techniques afforded by direct bimanual manipulation. Buxton and Myers found that the use of bimanual interaction enable parallel execution of sub-tasks, which significantly outperforms one-handed interaction in navigation as well as content selection and manipulation [6]. Song et al. proposed an interaction technique for 3D object manipulation that uses a handlebar metaphor [28]. With this technique, the user can manipulate objects by holding a handlebar that pierces the objects between the two hands and perform 7-DOF operations (i.e., translation + rotation + 1D scale) in a fast and precise manner.

Built upon Guiard's finding of the asymmetric roles of the hands [15], Stoakley et al. proposed the World in Miniature object, which can be held by the non-dominant hand as reference [29] and manipulated with the dominant hand to quickly navigate a virtual environment. Similarly, WritLarge, designed for wall-size touch display, allows users to frame a portion of canvas as selection with the non-dominant hand and simultaneously invoke actions on the selection with the dominant hand [35]. Hinckley et al. further divide the labor based on the input modalities of different hands – pen writes and touch manipulates [16].

While these techniques demonstrate the promise of direct physical manipulation for intuitive interactions, as well as rapid and simultaneous adjustments of multiple degrees of freedom, they do not afford the precise object manipulation required for any design tasks, due to the inability to separate degrees of freedom [20] and constant C/D gain [12]. A common approach, therefore, is to fall back on traditional graphical widgets for object translation, rotation, and scaling. However, using such widgets serializes the users' interaction, resulting in tedious operations. We seek to preserve bimanual physical manipulation for its directness and high throughput and provide precise object manipulation by allowing users to easily separate degrees of freedom and adjust the C/D gain.

Alignment and Constraint Support Tools

Much research has explored the use of snapping, alignment, distribution, and constraint support tools in 2D environments. While snapping, alignment, and distribution are typically implemented as transient commands, constraints typically persist within an environment. For example, snap-dragging automatically created transient alignment objects from drawing elements [3]. Briar integrated snap-dragging with constraints in 2D drawings [13]. Wybrow et al. evaluated the usability of multi-way constraints and found that they were beneficial for object alignment tasks during diagram editing [34]. Xu et al.'s beautification approach [36] inferred object relationships and used them for alignment, which enabled objects to be aligned in a balanced way. The Beyond Snapping approach [9] explored persistent alignment and distribution guides that could be directly manipulated and tweaked. All this work focused on 2D drawing contexts.

Little work, however, has generalized such methods to 3D environments. One of the challenges of applying such tools to 3D is that there are many more options for alignment and/or constraints in 3D and the additional visual complexity of a perspective view makes graphics-heavy mechanisms inadvisable.

Bier, for example, generalized snap-dragging into 3D [2], while others have explored the benefits of pre-defined constraints for 3D manipulation [4, 26, 27]. A related approach by Oh et. al. derived plane constraints from an object's contact with the environment [23], which implicitly reduced the degrees of freedom available for manipulation and aligns the position of objects with the environment. Recent work extended this approach to enable users to perform context-sensitive 3D positioning tasks even if they are not in contact [30]. Yet, all this work focused on desktop environments, i.e., applications that are used with a mouse and keyboard, and does not immediately generalize to VR scenarios.

Interaction Separating Degree of Freedom

Previous literature has shown that separating the degrees of freedom for desktop and touch based interactions is beneficial for precise object manipulation [20, 31], and many techniques have been developed to support DOF separation via virtual widgets and multi-touch gestures. Using a virtual widget with orthogonally placed controls is the most common way to provide isolated DOF manipulation and is widely available in 3D design and modeling software, such as Autodesk Maya and Blender. The implementation is often a variant of Brookshire et al.'s 3D widgets [11] that uses visual controls aligned to the local x, y, z axes of the object to translate, rotate, or scale. Although 3D widgets [11] was originally designed for a mouse, Mendes et al. found that the method is also useful for precise object manipulations in VR [20].

Other forms of degree-separating virtual widgets have also been studied. Houde developed Handlebox [18], a bounding box with handle-shaped controls used for isolated 1-DOF rotation or 2-DOF translation. Its user study showed that the handle-shaped controls could inform users what operations they could perform with the controls. Several studies investigated the use of high-DOF single hand widgets that support DOF separation, finding that such techniques outperformed bimanual manipulation techniques without DOF separation [7, 8]. Instead of using a widget with orthogonally placed handles, 7-Handle [22] uses seven point handles attached to an object. Each point can be locked in space to constrain the rotation as well as position. Work by Au et al. [1], and tBox [10] utilized the expanded gesture vocabulary afforded by multi-touch touch screens and enabled faster object manipulation along isolated axes. Schmidt et al. [24] presented a gesture interface that can create a transient widget for manipulating objects, which is aligned with the stroke the user made to create said widget.

While the use of these virtual widgets enables precise object manipulation, using these DOF-separation widgets often requires multiple steps to activate and manipulate them, resulting in a slower performance compared to direct manipulation methods [20].

Grossman et al. demonstrated DOF-separation via hand gestures, but their set was relatively small, and manipulations could *only* be performed in isolation [14]. Though also in 3D, our techniques are more similar to Rock & Rails, which employed shape gestures performed with the non-dominant hand to constraint free-form object transformations controlled by the dominant hand [33]. While designed only for 2D manipulation, this allowed users to quickly change the C/D gain, create constraints, isolate manipulation DOF, and preserve direct manipulation with the dominant hand. Our new *Plane, Ray, and Point* techniques employ the concept of shape gestures to enable the user to create constraints for DOF separation, allows variable C/D gain, and extends the idea into 3D environments, which required a new set of different gestures, graphical controls, and interaction methods. As our gesture-invoked constraints are lightweight and allow for rapid adaption to different contexts, they help users to quickly align and place objects in the virtual environment.

PLANE, RAY, & POINT INTERACTION

The goal of our interaction is to enable users to quickly and easily align and manipulate objects. To achieve this goal, the user must be able to access varying levels of DOF isolation and, for alignment, be able to perform such manipulations relative to other objects in the scene. We propose three shape constraints, *Plane*, *Ray*, and *Point*, each of which characterizes a canonical feature of a 3D object (face, edge, and vertex) and which further provides a corresponding level of DOF control. As such, we can afford a flexible range of techniques that isolate the DOF with respect to relevant features in the scene. Similar to Rock & Rails [33], we use a series of gesture-invoked, physically-mimicking constraints, enabling transform isolation and easy alignment.

Shape Gesture Vocabulary

A user can create a Plane by simultaneously pointing forward with her index finger and up with her thumb (Figure 2d). Alternatively, she can point forward with her index finger to create a forward Ray (Figure 2c) or point up with her thumb to create an upward Ray (Figure 2b). When her index finger and thumb are not extended, the user is able to create a Point (Figure 2a). Note that a Point will only be created if user is not grabbing an object. Planes, Rays, and Points disappear whenever the user is not performing the associated gestures, i.e., they are kinesthetically held quasimodes [25].

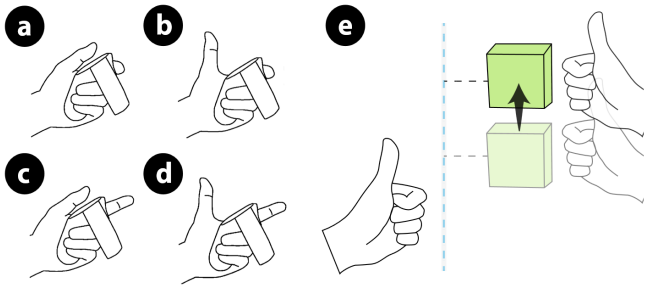

Figure 2. Hand gestures for creating a Point (a), creating an upward Ray (b), creating a forward Ray (c), and creating a Plane (d). The same hand gestures are also used on the dominant hand to further specify the DOF (e).

After creating a Plane, Ray, or Point, the user can squeeze her hand to activate that constraint. For our prototype, we use the *grip trigger* to detect if the user is squeezing her hand. When active, the constraint is locked in space and cannot move; users then do not need to maintain the shape gesture while the constraint is activated.

From our exploration of different gesture sets during the initial phases of the work reported here, we picked the ones above since they (1) adequately symbolize the respective shapes and (2) provide directionality, which is used to further specify the manipulation DOF (Figure 2e).

Grasping Objects

To facilitate the accurate grasping of objects, each hand has a cursor (Figure 3a) located at the place where the fingers would naturally come together to grasp an object (Figure 3b). To grasp an object, the user simply positions the cursor inside the object, and then squeezes her hand.

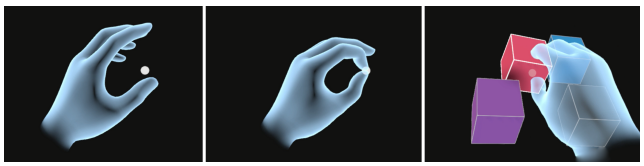

Figure 3. A white cursor located near the virtual hand (left) indicates where the hand will grasp an object (middle). If the cursor is inside the object, then the user can grab it (right).

Tethering Objects

To specify that an object should be actively constrained, the user must tether the object to a constraint. This can be done by clicking (i.e., temporarily grabbing) the object (Figure 4). Note, an object that is intersecting with the constraint is automatically tethered to it. To un-tether an object, the user must click the tethered object again.

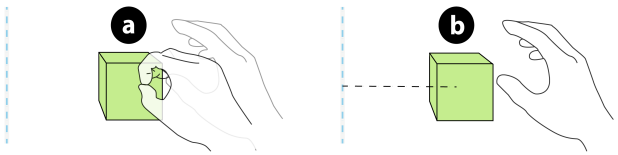

Figure 4. Tethering an object to a Ray by clicking on it.

Manipulation Hand Gesture Vocabulary

To constrain an object, the user activates the desired shape constraint (Plane, Ray, or Point) with one hand, and grasps the object with her other hand (which we refer to as the manipulation hand). Each shape constraint offers several different DOFs it can isolate. For example, a Ray can isolate to 1-DOF translation but also to 1-DOF rotation separately. To distinguish which DOFs to isolate, we use two techniques.

Hand Gestures with Manipulation Hand

With the manipulation hand, a user can point with either her index finger or her thumb in the direction of the DOF to isolate. For example, pointing parallel to the Ray will translate, while pointing "around" the Ray will lock it to rotation. Using this method, a user can switch the direction she is pointing to change the type of DOF isolation on the fly. Moreover, by explicitly pointing in multiple directions or only a single direction, a user can lock or unlock DOFs to offer a high number of possible constrained movements. In our implementation, a user begins a constrained manipulation of an object after she has pointed, with either index or thumb, in a direction for at least 0.5s (to prevent accidental manipulations when adjusting hand posture). When she changes her hand posture by opening or closing a finger, then the system recomputes and updates the direction. As such, a user does not need to always keep pointing in the exact direction with her finger, and at the same time, can rapidly switch between different types manipulations by 'opening' or 'closing' DOFs with her fingers.

Initial Movement of Object

While hand gestures offer flexible changes in a sequence of actions, we observed that for simple one-off actions, user tend to grab the object and move it in the desired direction. To minimize the overhead of performing the shape gesture for such quick actions, we also support the use of initial movement of the object to specify the direction to isolate. For instance, if the user starts moving an object *along* an activated Ray, then the object movement is locked to 1-DOF translation *along* that Ray. Similarly, if she initially moves her hand *around* the activated Ray, then the object is locked to 1-DOF rotation *around* the Ray. Should the user want to switch manipulation directions without re-grabbing the object, she can employ the hand gestures described earlier.

Plane Constraint

A Plane, when activated, mimics the behavior of a physical plane: it passively constraints all objects to allow for collision against it (Figure 5a).

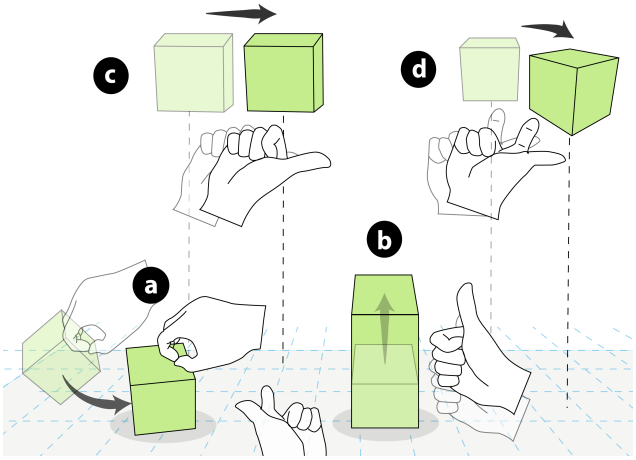

Figure 5. Interactions with an active Plane. (a) Objects collide against the Plane and move on it. (b) When an object was left on the Plane, re-grasping and moving the object scales it relative to the Plane. Tethered objects can (c) move along one of Plane's directional axes or (d) move parallel to the Plane.

Interactions with Tethered Objects

For tethered objects, the Plane generally offers the following constrained manipulations: (i) move the object parallel to the Plane (Figure 5d) and (ii) move the object along the Plane's normal. Using hand gestures for manipulation, as opposed to initial movement, one can also isolate to 1D translation along the Plane's directional vectors (Figure 5c).

Interactions with Intersecting Objects

As mentioned earlier, all intersecting objects are considered tethered. Beyond this, intersecting objects offer an additional constrained manipulation method. If an object is intersecting with the Plane, then instead of moving closer or further from the Plane, the object will 1D scale relative to the Plane (Figure 5b), with the metaphor being that when it is stuck to the Plane, the object stretches when pulled away.

Ray Constraint

An activated Ray is like a physical rail. When objects start intersecting with the Ray, they align against it and can be slid along it.

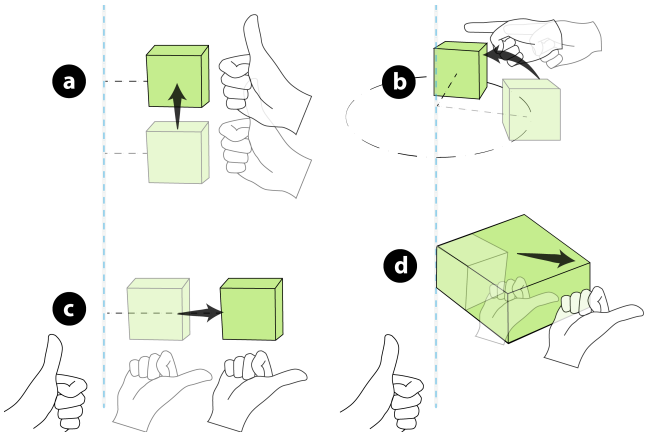

Figure 6. Tethered objects can (a) move along the Ray, (b) rotate around the Ray, (c) move away from the Ray. Intersecting objects can also (d) 2D uniform scale out from the Ray.

Interactions with Tethered Objects

The Ray offers three general manipulations: (i) move objects along the Ray (Figure 6a), (ii) rotate objects around the Ray (Figure 6b), and (iii) move objects closer or further away from the Ray (Figure 6c). Like before, hand gestures can be used to combine different manipulations, for example, one can point up to move an object along the Ray, and at the same time point forward to move it along the Ray as well.

Interactions with Intersecting Objects

Analogous to the Plane, if an object is intersecting a Ray, it 2D scales when motioned away from the Ray (Figure 6e).

Point Constraint

An active Point does not impose any restrictions on an object unless the object is explicitly tethered to it. It behaves like a pivot, with objects being able to move closer or further to it, as well as rotate around the Point.

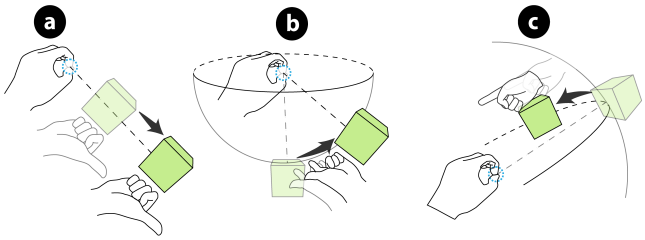

Figure 7. An object that is (a) changing its distance from the Point, (b) rotating around the Point, and (c) rotating around a concentric circle using hand gestures

Interactions with Tethered Objects

Tethered objects can be constrained in the following ways: they can (1) move objects closer or further away from the Point (Figure 7a) or (2) rotate objects around the Point (Figure 7b). A user can also use hand gestures to rotate the object on a concentric circle around the point (Figure 7c).

A Point does not offer any additional manipulations to objects that intersect with it.

Constraint Snapping

Snapping is used to enable constraints to be placed faster and more accurately. All thresholds were empirically determined based on observations during pilot tests.

A Plane can snap to the faces of an object's bounding box to help align objects along the surface of an object and enable the Plane to stretch along the local axis of the object. The Plane snaps to a face if the angle between the Plane and face is less than 10 degrees, the orthogonal distance from the Plane to the face is less than 1 cm, and the Euclidian distance between the hand and the closest point on the face is less than 30 cm. Analogously, the Ray can snap to the edges of an object's bounding box. It snaps to an edge if the angle between them is less than 10 degrees and the Euclidean distance from the Ray to the edge is less than 1 cm. We found that the above-mentioned values work best for the scale of the objects used in the study.

Multi-Object Manipulation

As mentioned previously, when a constraint is activated, the user can tether multiple objects to it. Normally, the user would grasp a tethered or intersecting object to begin her manipulations, however, if she grasps at the air instead, she can manipulate all tethered objects simultaneously.

Persistent Constraints and Grouping

A user can also create permanent constraints in the world. When a constraint is activated, elastic bands connect the constraint and the hand. As the hand moves away from the constraint, the elastic begins to stretch. After the hand has moved 15 cm away from the constraint, the elastic breaks and the constraint persists in the environment (Figure 8). These persistent constraints are like normal objects except they can be re-activated to become a constraint again.

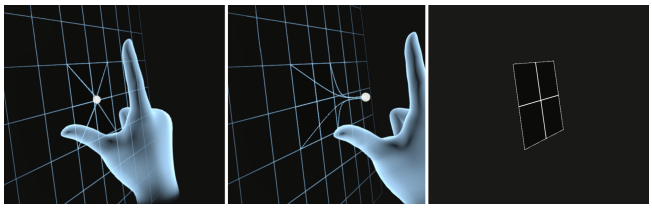

Figure 8. Elastic bands stretch as the hand moves further away from the constraint, until eventually breaking and leaving the constraint permanently in the world.

A persistent constraint also maintains its tethers; however, it is inactive by default and as such, does not constrain any objects. To re-activate a constraint, the user must move her hand over the constraint and then squeeze the controller. Once re-activated, the constraint behaves as it did before the elastic was broken.

Being able to leave constraints in the environment and come back to them at a later time removes the (tedious) repetition in re-creating the same constraint. For example, a designer may want to create a Plane as a floor and have

objects that are constrained to it. Instead of creating this constraint and repeatedly tethering the objects, the designer can make the constraint persistent and leave it in the environment. Then, to re-activate it, she can squeeze the controller as needed.

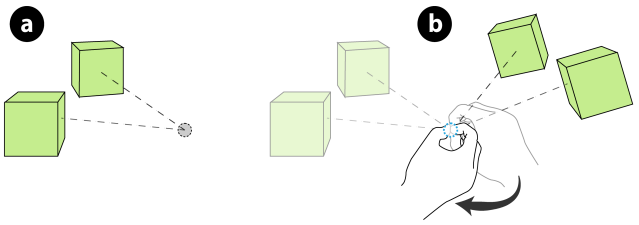

Figure 9. A Point acts as a natural metaphor for grouping objects, with a persistent Point that has objects tethered to it (a), the user can grasp the Point with her index (b) to move all tethered objects around like a group.

Alternatively, if the user moves her hand over a constraint and holds down on the index trigger, then she can move the constraint around and treat it as an object. As she manipulates the constraint, all objects tethered to it are also manipulated. As such, one can group objects together by connecting them to the same constraint and then can manipulate the constraint as a whole. A Point serves as a natural shape constraint for grouping objects together (Figure 9).

Multi-Constraint Manipulation

There are many instances where a user may need to activate multiple constraints at the same time. For instance, if a designer needed to place a bookcase against a wall, she would need both a ground Plane constraint as well as a wall Plane constraint to be activated simultaneously. In such cases, the user can first create and activate a Point. Then, she tethers the constraints she wishes to activate to that Point. While the Point is active, the tethered constraints also remain active. As a proof of concept, multiple Plane constraints can be activated simultaneously to define more complex collision surfaces such as corners (Figure 10).

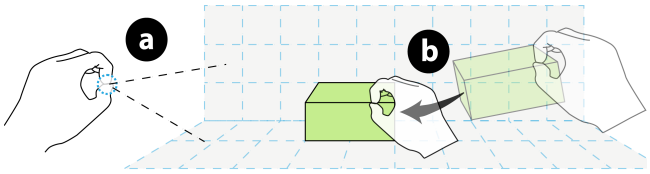

Figure 10. To create and use a corner, we first activate each of the associated walls using a Point (a), and then can grab any object to place it against the corner (b).

IMPLEMENTATION

To evaluate the *Plane, Ray, and Point* interactions, we developed a system using an Oculus Rift HMD and two Oculus Touch controllers. The HMD and the controllers were tracked by four Oculus sensors. The software was implemented in C# using the Unity game engine and Oculus SDK and ran on a desktop computer with Intel i7-4770 processor and a NVidia GTX1050Ti graphics card.

As Oculus Touch controllers have capacitive sensors in the index finger trigger and thumb buttons, they can detect if the finger or the thumb is resting on the controller. During initial testing, the finger posture detection that was built into the controller was not reliable enough to detect the finger stretch gestures used for the *Plane, Ray, and Point*. To overcome this limitation, two gesture detection gloves were designed (Figure 11).

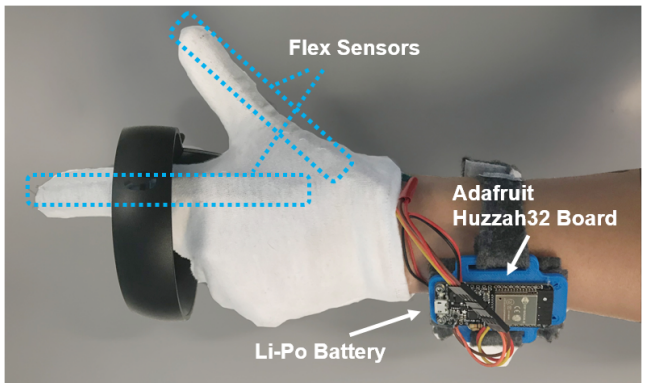

Figure 11. Gesture glove for detecting the gestures, while holding the controller in hand.

Each glove had two 114 mm long flex sensors running along the index finger and thumb. An Adafruit Huzzah32 board with an ESP32 microprocessor sampled the sensor values at 60 Hz and relayed the values to a PC via Bluetooth connection. Each glove was powered by a 1000mAh Lithium Polymer battery. To accommodate different hand sizes and finger angles, the system initially requires the user to comfortably rest her index finger and thumb on the controller and extend them while holding the controller in hand. The system then measures the flex sensor value range for the index finger and thumb and also monitors it when the users are closing their fingers during calibration. To prevent jitter, the threshold to detect an extended finger was set at 80% of the range (100% being the maximum stretch) and to cancel the stretch at 60%, but the thresholds were adjusted by the experimenter to tailor them to individual differences.

USER STUDY

We seek to understand whether the proposed techniques can work as a whole to support existing workflows for composing 3D scenes. We conducted an expert evaluation to gain feedback on the utility and usability of *Plane, Ray, and Point*, where we encouraged them to compare it with the user interfaces of software they regularly use. We also received feedback from novice users to evaluate the ease of learning and using our new user interface.

Participants

We recruited nine participants: six (E1-E6) were expert users (1 female; μ = 32 years, range = 27 to 41 years) with 3D modelling tools (average 12 years of experience). The other three were novices (N1-N3) with minimal experience using 3D modelling tools (2 females; μ = 20 years, range =

19 to 22 years). Out of the six experts, two (E1, E6) were also experts in VR development (average 2.5 years of experience). No participant had notable experience with modeling within VR; indeed, we believe that the population of such users is de minimis. It is our hope that by supporting expert users from these populations we can provide useful and usable tools which can in part help to enable such a user group to form.

Apparatus

The study was conducted in a room with a working area of 180 x 180 cm. Participants were encouraged to move around so that they could see various objects from a viewpoint where they could understand the 3D locations of the objects.

Procedure

The study consisted of the following phases.

Greeting and Introduction (5 minutes)

The participant filled out the consent form as well as a demographic questionnaire. Then, the purpose of the study and the basic concepts of *Plane, Ray and Point* were explained.

Tutorial (20 minutes)

After being given brief instructions on the use of *Plane, Ray, and Point*, participants wore the Oculus headset and proceeded through a guided tutorial using the techniques. The tutorial covered creating each of the shape constraints, grabbing objects, tethering objects, performing different types of manipulations with the constraints using both the initial movement and the hand gesture, and performing multi-object manipulations with the constraints. As the participants were learning and being introduced to these techniques, they were encouraged to speak out loud and provide thoughts on the interactions.

Guided Scenarios (20 minutes)

Once the participant completed the tutorial, they were guided through four scenarios which covered the main features – creating various appropriate shape constraints and using them to precisely manipulate objects. They included the creation and manipulation of different persistent constraints, and interaction with multiple constraints and objects simultaneously. In all four scenarios, the scene was populated with a variety of objects and participants were encouraged to explore and deviate from the script after finishing the required task, as well as to provide us their thoughts on the interaction.

Interview (20 minutes)

After the guided scenarios, participants were asked to evaluate each of the features in *Plane, Ray, and Point* with regards to their (i) ease of use and learning and (ii) advantages and disadvantages of the techniques over previous techniques they had used in desktop software and VR tools (if they were expert users). They were also asked to evaluate the gestures used in each interaction.

Results

The results from the novice and expert users revealed many interesting recommendations and opportunities for *Plane, Ray, and Point.* We start with the overall usefulness of our techniques in comparison to traditional workflows, and then detail specific usability features.

Comparison to Traditional Workflows

All experts agreed that the *Plane, Ray, and Point* interactions, along with the affordances of VR, could speed up their design workflows. E2, an architect, believed that these tools would be most helpful in a schematic design phase involving operations to easily define and use the Ray as an axis rotation for direction lights (like the Sun) as well as to enable the quick alignment of objects against one another (using Planes). E4 also commented that while these interactions and others in VR would work on a small scale to design rooms, for example, in the current state, none would be usable for modelling on a large scale, where exact distances, quick grouping, and multiple viewpoints are required.

When asked about the usefulness of the interactions compared to traditional tools, E3 also noted that they are *"interested in creating it [Plane] at a weird angle because that is actually tough to do in current software"*, indicating novel use-cases that are not easily supported by traditional tools.

Shape Gesture Language

Most participants (i.e., E1,3-5, N1-3) were able to comfortably create all different shape gestures. Only E6 noted that extending their thumb was uncomfortable. E2 also had difficulty maintaining the gesture for the Plane when pressing down on the grip trigger to activate it. They described that it was hard to keep the index and thumb open while closing their middle finger. E4, E2, and E6 also had concerns that the gestures could get tiring over time, mainly because they were trying to balance the controller in their hand while performing them.

Plane Constraint Interactions

Being able to place objects against the Plane was quickly picked up by all participants and was unanimously cited as easy to use and helpful when quickly aligning objects.

As for the 1-DOF scaling of objects using the Plane, all participants were able to easily learn and perform the interaction but were divided on whether this was a useful feature. E1, E3, and E5 stated that 1-DOF scaling will be useful in their workflows but E2 did not see a significant number of use cases for 1-DOF scaling. E2 argued that one would usually want uniform scaling for non-primitive objects. E4 was neutral. They could see it being useful in certain stages of their design process, for example, when they may need to quickly scaffold walls or floors out of primitive objects. However, when populating a scene, they usually used objects that were already correctly scaled, so 1-DOF scaling was undesirable.

Ray Constraint Interactions

Generally, participants did not experience issues with Ray; most participants noted that it had fewer use-cases than the Plane but could be very helpful in the right situation. E2 raised similar concerns for 2D scaling as for the 1D scaling with the Plane, in that, uniform scaling was usually the most desirable type of scaling and 2D would not be used often. E5 would *"find the Ray more useful if it snapped to normal of Planes or surfaces".*

Point Constraints Gesture

All participants found Point to be easy to make and activate. Only E6 found it hard to use. They thought of Point as more of a marker or reference rather than a constraint, and as such expected different interactions. E2 mentioned that it does not constrain the objects enough for it be useful in scene or architecture design. They suggested it would be better suited to more freeform scenarios such as character modelling. All participants except E6 agreed that the point provides a natural metaphor for grouping objects together. E6 thought of the Point as only applicable to apply constraints to objects so using it for grouping would not be intuitive. They suggested being able to click or double click on objects to select them, independent of creating any constraints. E3 was also concerned that with multiple groups, it would be hard to differentiate between each group and which point one should grab.

Multi-object Manipulation

All participants agreed that the gesture used to grab the air to move all objects was easy to learn and use. E3 *"would definitely use this for 1D or 2D scaling objects all at the same time",* for scenarios such as making all the walls in a room taller, where each wall would need to be scaled by the same amount.

Persistent vs Temporary Constraints

The gesture for making a constraint persistent was easy to perform and participants easily recognized it from the visual feedback provided. Many participants, however, accidently triggered it and inadvertently broke the elastic when trying to rest their hand. Two experts, E2 and E5, would have liked to always create a persistent constraint by default, e.g., when they activated the Plane and then released the grip trigger, they would have liked the Plane to be left in the world and an additional gesture be used to temporarily create a Plane. E5 mentioned that the constraint left in the world should always be activated by default, because otherwise if they *"parent a bunch of objects to this constraint, the fact that I have to go back and activate the constraint to manipulate those objects will take some getting used to".*

Multiple Active Constraints

All participants found the gesture for activating multiple constraints to be intuitive, but E2 and E5 did not find it ideal. E5 commented that the gesture serialized selection and wished that by default constraints would remain active when left in world. They wanted to place a bunch of Planes in the environment and start placing objects against them.

Initial Movement and Manipulation Hand Gestures

After creating a constraint, users have the option to either use their initial movement (less powerful, but potentially easier to use), or a hand gesture (more robust, but potentially harder to use) in order to manipulate objects. We found that all participants were able to quickly use initial movements to choose the constraint type and found it intuitive to use. All participants excluding E6 were able to effectively use hand gestures with their non-dominant hand. However, initial movement was used much more frequently than the hand gestures. All experts except E6 agreed that they would have to get used to the hand gestures, but after learning them, agreed they would use them a lot more due to the benefits over initial movements. E6 did not find it easy to make the gestures, and because one would need to maintain them when using the manipulation hand gestures, it would be straining for them.

Miscellaneous

Most participants experienced difficulties with grabbing persistent constraints because they were sometimes too small (Point) or too thin (Ray, Plane). E2, E5, E6 would have liked there to be a way to remove constraints.

The two experts (E1,E6) who were also experienced in VR development learned our techniques much faster than others, but all experts were equally capable of using our techniques after the tutorial.

DISCUSSION

The study revealed that our techniques were easy to learn for novices and can be very useful in scene design workflows. The gestures were found to be fast to perform, easy to remember and use, and afforded direct, efficient manipulation of objects in VR. The asymmetric use of the two hands and the adoption of naïve physics [13] might be two of the main reasons that make the techniques so easy to use. The gestures are also similar to gestures already used in the real world to describe common physical constraints. The fact that constraints can be both transient and persistent in our method made it also easy for users to adapt their usage to the current context.

The biggest benefit of *Plane, Ray, and Point* is that the techniques make it very easy to avoid one of the biggest drawbacks of current VR systems: namely that objects cannot be (easily) aligned with other objects, which leads to objects being posed at slightly odd angles, interpenetrating slightly, or floating a bit in the air. With *Plane, Ray, and Point* objects can be placed more accurately, which enables users to convey the intent of their virtual designs better.

The presented techniques are (mostly) orthogonal to current interaction schemes in VR systems. This makes it easy to add them to other applications that currently do not yet support constraints.

While the results are promising, potential usability issues and limitations of the techniques do need to be addressed. From an ergonomic viewpoint, the main usability concern

would be fatigue, potentially caused by performing the required gesture(s) while simultaneously holding and balancing the controller(s). An alternative would be to use a controller that is fixated to the hand [17] so it does need to be balanced; this should reduce strain and ensure that the gestures are even more comfortable to perform.

Alternatively, the design of *Plane, Ray and Point* is such that it could be easily adapted to freehand gestural input, removing the need for controllers. Of course, this would introduce its own set of learnability and usability concerns that would need to be considered and designed for.

There are some limits to the types of scenes supported by our technique. First, our techniques may not work well if there are very dense clutters of snapping targets (such as multiple surfaces stacked close to each other), since the constraints could snap to any one of a set of unintended objects. Second, as pointed out by E4, our techniques can encounter limits when manipulating large-scale scenes with deeply nested hierarchical structures.

Our study also demonstrated that there were a few usability issues with the persistent constraints. First, the need to maintain the hand position in space to keep them active did not permit the user to drop their hand to waist-level, as dropping the hand implies moving one's hand away from the constraint, thus breaking the elastic bands. A more explicit action may be better suited for keeping a constraint active, e.g., a simple press of a button on the controller. Having said that, typical manipulation episodes do not last long enough for this to be a major concern. Another issue was that using a Point to activate multiple persistent constraints simultaneously required the user to first create the Point and then select each constraint the user wanted to activate. This interaction can get tedious, and for some users, the Point may not be a natural metaphor for such an interaction. Toggling constraint activation as needed through a button on the controller would again fix this. Lastly, we did not support deletion of persistent constraints, but this could again be mapped to pressing or holding a button. As this was not a contribution of *Plane, Ray, and Point* as presented above, it was not included in the original system.

CONCLUSION AND FUTURE WORK

In this work, we presented new interactions to quickly and easily align and constrain the manipulation of objects in VR. Our new interaction technique, *Plane, Ray and Point*, uses shape gestures to constrain different DOFs. Due to its use of physical metaphors and direct manipulation, the technique was easily learned and used by novices and experts alike.

There are several interesting directions that future work should explore. First, this work only considered straight Rays and flat Planes. A natural extension may be to extract more information from the shape of a virtual object and create curved constraints that lie along the surfaces of objects, such as a sphere or rolling hills. Such curved constraints could enable quick and easy object alignment with round or irregular-shaped objects. Future work could also investigate adapting our new interactions to hands-free interfaces. This would, however, require new methods to switch modes to activate constraints, because trigger or controller buttons cannot be used in such a system.

REFERENCES

[1] Oscar Kin-Chung Au, Chiew-Lan Tai, and Hongbo Fu. 2012. Multitouch Gestures for Constrained Transformation of 3D Objects. *Computer Graphics Forum* 31, 2pt3 (2012), 651–660. http://dx.doi.org/10.1111/j.1467-8659.2012.03044.x

[2] Eric A. Bier. 1990. Snap-dragging in Three Dimensions. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics (I3D '90)*. ACM, New York, NY, USA, 193–204. http://dx.doi.org/10.1145/91385.91446

[3] Eric A. Bier and Maureen C. Stone. 1986. Snap-dragging. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. ACM, New York, NY, USA, 233–240. http://dx.doi.org/10.1145/15922.15912

[4] Richard W. Bukowski and Carlo H. Séquin. 1995. Object Associations: A Simple and Practical Approach to Virtual 3D Manipulation. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics (I3D '95)*. ACM, New York, NY, USA, 131–ff. http://dx.doi.org/10.1145/199404.199427

[5] Jeff Butterworth, Andrew Davidson, Stephen Hench, and Marc. T. Olano. 1992. 3DM: A Three Dimensional Modeler Using a Head-mounted Display. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics (I3D '92)*. ACM, New York, NY, USA, 135–138. http://dx.doi.org/10.1145/147156.147182

[6] W. Buxton and B. Myers. 1986. A Study in Two-handed Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '86)*. ACM, New York, NY, USA, 321–326. http://dx.doi.org/10.1145/22627.22390

[7] Fabio M. Caputo, Marco Emporio, and Andrea Giachetti. 2018. The Smart Pin: An effective tool for object manipulation in immersive virtual reality environments. *Computers & Graphics* 74: 225–233. https://doi.org/10.1016/j.cag.2018.05.019

[8] Fabio Marco Caputo, Marco Emporio, and Andrea Giachetti. 2017. Single-Handed vs. Two Handed Manipulation in Virtual Reality: A Novel Metaphor and Experimental Comparisons. *The Eurographics Association*. https://doi.org/10.2312/stag.20171225

[9] Marianela Ciolfi Felice, Nolwenn Maudet, Wendy E. Mackay, and Michel Beaudouin-Lafon. 2016. Beyond Snapping: Persistent, Tweakable Alignment and Distribution with StickyLines. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 133–144. http://dx.doi.org/10.1145/2984511.2984577

[10] Aurélie Cohé, Fabrice Dècle, and Martin Hachet. 2011. tBox: A 3D Transformation Widget Designed for Touch-screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 3005–3008. http://dx.doi.org/10.1145/1978942.1979387

[11] Brookshire D. Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. 1992. Three-dimensional Widgets. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics (I3D '92).* ACM, New York, NY, USA, 183–188. http://dx.doi.org/10.1145/147156.147199

[12] Scott Frees, G. Drew Kessler, and Edwin Kay. 2007. PRISM Interaction for Enhancing Control in Immersive Virtual Environments. *ACM Trans. Comput.-Hum. Interact.* 14, 1, Article 2 (May 2007). http://dx.doi.org/10.1145/1229855.1229857

[13] Michael Gleicher. 1992. Briar: A Constraint-based Drawing Program. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 661–662. http://dx.doi.org/10.1145/142750.143074

[14] Tovi Grossman, Daniel Wigdor, and Ravin Balakrishnan. 2004. Multi-finger Gestural Interaction with 3D Volumetric Displays. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST '04)*. ACM, New York, NY, USA, 61–70. http://dx.doi.org/10.1145/1029632.1029644

[15] Yves Guiard. 1987. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of motor behavior* 19, 4 (1987), 486–517.

[16] Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and Bill Buxton. 2010. Pen + Touch = New Tools. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST '10).* ACM, New York, NY, USA, 27–36. http://dx.doi.org/10.1145/1866029.1866036

[17] Sean Hollister. 2018a. Valve's Knuckles EV2 controller will let you squeeze objects in games. (Jun 2018). https://www.cnet.com/news/valves-new-knuckles-vr-controller-knows-where-your-fingers-are/

[18] Stephanie Houde. 1992. Iterative Design of an Interface for Easy 3-D Direct Manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 135–142. http://dx.doi.org/10.1145/142750.142772

[19] D. Mendes, F. M. Caputo, A. Giachetti, A. Ferreira, and J. Jorge. 2019. A Survey on 3D Virtual Object Manipulation: From the Desktop to Immersive Virtual Environments. *Computer Graphics Forum* 38, 1 (2019), 21–45. http://dx.doi.org/10.1111/cgf.13390

[20] Daniel Mendes, Filipe Relvas, Alfredo Ferreira, and Joaquim Jorge. 2016. The Benefits of DOF Separation in Mid-air 3D Object Manipulation. *In Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology (VRST '16)*. ACM, New York, NY, USA, 261–268. http://dx.doi.org/10.1145/2993369.2993396

[21] Daniel Mendes, Maurício Sousa, Rodrigo Lorena, Alfredo Ferreira, and Joaquim Jorge. 2017. Using Custom Transformation Axes for Mid-air Manipulation of 3D Virtual Objects. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology (VRST '17)*. ACM, New York, NY, USA, Article 27, 8 pages. http://dx.doi.org/10.1145/3139131.3139157

[22] Thi-Thuong Huyen Nguyen, Thierry Duval, and Charles Pontonnier. 2014. A New Direct Manipulation Technique for Immersive 3D Virtual Environments. In *Proceedings of the 24th International Conference on Artificial Reality and Telexistence and the 19th Eurographics Symposium on Virtual Environments (ICAT - EGVE '14).* Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 67–74. http://dx.doi.org/10.2312/ve.20141367

[23] Ji-Young Oh and Wolfgang Stuerzlinger. 2005. Moving Objects with 2D Input Devices in CAD Systems and Desktop Virtual Environments. In *Proceedings of Graphics Interface 2005 (GI '05)*. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 195–202. http://dl.acm.org/citation.cfm?id=1089508.1089541

[24] Ryan Schmidt, Karan Singh, and Ravin Balakrishnan. 2008. Sketching and Composing Widgets for 3D Manipulation. Computer Graphics Forum 27, 2 (2008), 301–310. http://dx.doi.org/10.1111/j.1467-8659.2008.01127.x

[25] Abigail J. Sellen, Gordon P. Kurtenbach, and William A. S. Buxton. 1992. The Prevention of Mode Errors Through Sensory Feedback. *Hum.-Comput. Interact*. 7, 2 (1992), 141–164. https://doi.org/10.1207/s15327051hci0702_1

[26] Graham Smith, Tim Salzman, and Wolfgang Stuerzlinger. 2001. 3D Scene Manipulation with 2D Devices and Constraints. In *Proceedings of Graphics Interface 2001 (GI '01)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 135–142. http://dl.acm.org/citation.cfm?id=780986.781003

[27] Graham Smith and Wolfgang Stuerzlinger. 2001. Integration of Constraints into a VR Environment.

[28] Peng Song, Wooi Boon Goh, William Hutama, Chi-Wing Fu, and Xiaopei Liu. 2012. A Handle Bar Metaphor for Virtual Object Manipulation with Mid-air Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1297–1306. http://dx.doi.org/10.1145/2207676.2208585

[29] Richard Stoakley, Matthew J. Conway, and Randy Pausch. 1995. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 265–272. http://dx.doi.org/10.1145/223904.223938

[30] Junwei Sun, Wolfgang Stuerzlinger, and Dmitri Shuralyov. 2016. Shift-Sliding and Depth-Pop for 3D Positioning. In *Proceedings of the 2016 Symposium on Spatial User Interaction (SUI '16)*. ACM, New York, NY, USA, 165–165. http://dx.doi.org/10.1145/2983310.2991067

[31] Manuel Veit, Antonio Capobianco, and Dominique Bechmann. 2009. Influence of Degrees of Freedom's Manipulation on Performances During Orientation Tasks in Virtual Reality Environments. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology (VRST '09)*. ACM, New York, NY, USA, 51–58. http://dx.doi.org/10.1145/1643928.1643942

[32] Yanqing Wang, Christine L. MacKenzie, Valerie A. Summers, and Kellogg S. Booth. 1998. The Structure of Object Transportation and Orientation in Human-computer Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '98)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 312–319. http://dx.doi.org/10.1145/274644.274688

[33] Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo, and Sarah Williams. 2011. Rock & Rails: Extending Multi-touch Interactions with Shape Gestures to Enable Precise Spatial Manipulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1581–1590. http://dx.doi.org/10.1145/1978942.1979173

[34] Michael Wybrow, Kim Marriott, Linda Mciver, and Peter J. Stuckey. 2008. Comparing Usability of One-way and Multi-way Constraints for Diagram Editing. *ACM Trans. Comput.-Hum. Interact*. 14, 4, Article 19 (Jan. 2008), 38 pages. http://dx.doi.org/10.1145/1314683.1314687

[35] Haijun Xia, Ken Hinckley, Michel Pahud, Xiao Tu, and Bill Buxton. 2017. WritLarge: Ink Unleashed by Unified Scope, Action, & Zoom. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3227–3240. http://dx.doi.org/10.1145/3025453.3025664

[36] Pengfei Xu, Hongbo Fu, Takeo Igarashi, and Chiew-Lan Tai. 2014. Global Beautification of Layouts with Interactive Ambiguity Resolution. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 243–252. http://dx.doi.org/10.1145/2642918.2647398

[37] 2018. See How Steven Spielberg Used Real VR Headsets to Make "Ready Player One." Film. Retrieved August 16, 2018 from https://www.slashfilm.com/ready-player-one-vr-featurette/

[38] Unreal Engine VR Mode. Retrieved August 16, 2018 from https://docs.unrealengine.com/en-us/Engine/Editor/VR

[39] 3D Visualization Software | VRED | Autodesk. Retrieved August 17, 2018 from https://www.autodesk.com/products/vred/overview