3D Scene Manipulation with Constraints

Graham Smith, Tim Salzman, Wolfgang Stuerzlinger
York University
Toronto, Canada

Abstract

Content creation for computer graphics applications is a laborious process that requires skilled personnel. One fundamental problem is that manipulation of 3D objects with 2D user interfaces is very difficult for non-experienced users.
In this paper, we introduce a new system that uses constraints to restrict object motion in a 3D scene, making interaction much simpler and more intuitive. We compare three different 3D scene manipulation techniques based on a 2D user interface. We show that the presented techniques are significantly more efficient than commonly used solutions. To our knowledge, this is the first evaluation of 3D manipulation techniques with 2D devices and constraints.

1 Introduction

The task of creating a 3D scene from scratch is very complex. To simplify the problem, we choose to focus on the creation of complete 3D scenes based on a library of existing objects. Here the challenge is to enable the user to easily add objects and to quickly position them in the environment. In general, positioning an object in a 3D scene is difficult as six independent variables must be controlled, three for positioning and three for orientation.
Our observations of humans rearranging furniture and planning environments indicate that humans do not think about scene manipulation as a problem with six degrees of freedom. The rationale is that most real objects are not placed arbitrarily in space, but are constrained by physics (e.g. gravity) and/or human conventions (ceiling lamps are almost never placed permanently onto the floor or onto chairs). This leads us to believe that an interface that exposes the full six degrees of freedom to the user makes it harder for average persons to interact with virtual environments. Many real objects have a maximum of three degrees of freedom in practice – e.g. all objects resting on a plane. In addition, many objects are often placed against walls or other objects, thus further reducing the available degrees of freedom. This implies that a two-dimensional (2D) input device such as a mouse is sufficient to manipulate objects in a virtual environment.
In our system, information about how an object interacts with the physical world assists the user in placing and manipulating objects in virtual environments. Each object in a scene is given a set of rules, called constraints, which must be followed when the object is being manipulated. This concept of constraints makes manipulating objects in 3D with 2D devices much simpler.

1.1 Previous Work

For 2D object manipulation various forms of constraint systems have been introduced. For recent work on interactive constraint satisfaction and references to previous work see [3][11]. We will concentrate here on 3D object manipulation with 2D devices. For work with 3D devices, we refer the reader to [10].

The simplest solution for a 2D input device is to decompose the manipulation task into positioning and orientation. Unfortunately, there is no intuitive mapping of these tasks with three degrees of freedom each to a mouse with three buttons.

Bier introduced 'Snap-Dragging' [1] to simplify the creation of line drawings in a 2D interactive graphics program. The mouse cursor snaps to points and curves using a gravity function. Bier subsequently applied these ideas to placing and orienting objects in a 3D environment [2]. The main features of this system are a general-purpose gravity function, 3D alignment objects, and smooth motion affine transformations of objects. Gleicher [6] built on this work and introduced a method that can deal even with non-linear constraints.

For 3D scene construction Bukowski and Sequin [3] employ a combination of pseudo-physical and goal-oriented properties called 'Object Associations' to position objects in a 3D scene with 2D devices (mouse and monitor). A two-phase approach is used. First, a relocation procedure maps the 2D mouse motion into vertical or horizontal transformations of an object's position. Then association procedures align and position the object. Although intuitive, their approach has a few drawbacks. First, associations apply only to the object currently being moved and are not maintained after the current manipulation. In addition, when an object is selected for relocation, a local search for associated objects is performed. This can result in lag between the motion of the selected object and the motion of its associated objects. Cyclical constraints are not supported.

Goesele and Stuerzlinger [7] built upon the ideas of Object Associations. Each scene object is given predefined offer and binding areas. These areas are used to define constraining surfaces between objects. For example, a lamp has a binding area at its base and a table has an offer area on its top. Consequently, a lamp can be constrained to a tabletop. To better simulate the way real world objects behave, a labeled constraint hierarchy adds semantics to the constraint process. Each constraint area is associated with a label from the hierarchy. A binding area constrains then only to offer areas whose label is equal to or is a descendant in the constraint hierarchy. In this way, the legs of a chair can be constrained to the floor, or in front of a desk, but never to the wall. Collision detection is used to prevent objects from passing though each other.

Drawbacks of this approach include the following: Once a constraint has been satisfied, there are no means to re-constrain an object to another surface or to unconstrain it. Furthermore, the constraint satisfaction search is global, in that an object will be moved across the entire scene to satisfy a constraint, This has often-undesirable effects for the user, especially because constraints cannot be undone.

While it may seem obvious that the introduction of constraints makes interaction in 3D easier, it is unclear how strong this effect is. This is an issue that appears to have been previously neglected in the literature.

1.2 Motivation

Most relevant to the research presented here is the work by Poupyrev et al. [8]. There, different interaction methods for 3D input devices are compared. All techniques that perform well in practice are based on the notion of ray casting. Ray casting identifies the first object that is visible along an infinite ray from the manipulation device into the scene (much like a laser pointer). As a 2D image describes all visible objects, the authors hypothesize in [8] that all ray casting techniques can be approximated as 2D techniques (see also [4]). This supports our observation that for most situations a user interface that utilizes only a 2D input device is sufficient to effectively manipulate objects in 3D. The fact that most successful commercial products (e.g., *Maya, 3D Studio MAX*) use almost exclusively 2D input can also be seen as further support.
Based on the mentioned observations we decided to investigate the performance of different object manipulation techniques for 3D environments with 2D devices. Most people assume correctly that constraints will provide benefits, but it is unclear how much faster constraints systems are for this task as no formal evaluation of constraint systems for 3D object manipulation has been published to our knowledge.

2 The MIVE System

The MIVE (Multi-user Intuitive Virtual Environment) system extends the work done in [7] by improving the way existing constraints behave, and adding new types of constraints. This work concerns only the interaction of a single user with the system. Therefore, we disregard the multi-user aspects of the system here.

2.1 Constraints

Each object can have any number of user-defined constraints. Constraint definition is done on a per model basis, and is done in a separate program. A constraint can be one of three types: offer, binding, or dual. The binding areas of one object constrain to offer areas of another. Dual constraints are explained in detail below. When a user is interacting with a scene, feedback is given by highlighting in green any offer area where the manipulated object can be placed.

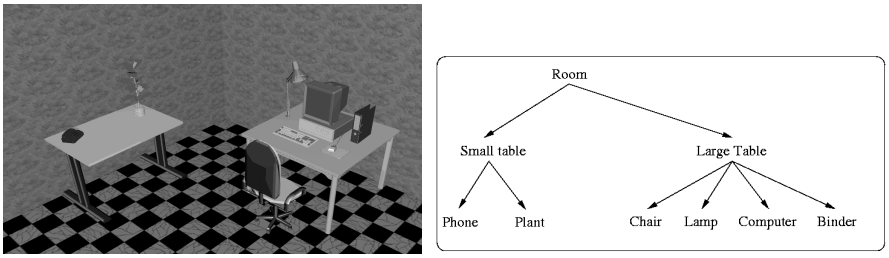


Figure 1: Scene and associated scene graph.

The constraint relationships are stored in a directed a-cyclic graph called the scene graph. Figure 1 depicts a simple scene, and its associated scene graph. When an object is moved in the scene, all descendants in the scene graph move with it.
Notice that edges in the scene graph of figure 1 correspond directly to satisfied constraints in the scene. The user can modify the scene graph structure by interacting with objects in the scene. Constraints can be broken and objects can be re-constrained by simply clicking on the desired object, and pulling away.

2.2 Virtual Constraints

In MIVE, geometry of objects and the geometric definition of the constraints of objects are not the same. In fact, constraints for an object can float in mid air. We call these 'virtual constraints'. For example, the table has a floating offer area underneath it for the front of a chair to constrain to.
Notice that another big advantage of virtual constraints is that the geometry and constraints are effectively de-coupled. Consequently this works even with geometry that features slight inaccuracies such as small cracks in the bottom of a table leg, a flat surface that is composed of many triangles, or intersecting polygons.

2.3 Dual Constraints

Since the scene graph is a-cyclic, only parent child relationships are allowed between constrained objects. In order to facilitate the use of cyclical constraints, we created a new type of constraint, which we call 'dual constraint'. The dual constraint is very useful in situations where a sibling relationship between constraints makes more sense than a parent child relationship. For example, we can use a dual constraint to constrain two cabinets side by side on a wall.

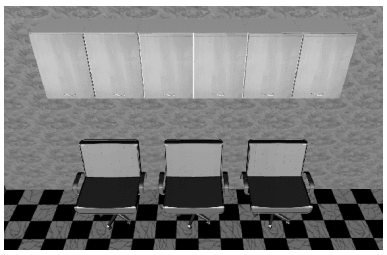


Figure 2: Two dual groups

By dual constraining objects together, we create 'dual groups'. Figure 2 shows two dual groups: one group of cabinets on a wall, and one group of chairs on the floor. Objects in the dual group are translated, rotated, and regrouped together. To break a dual group, the user simply selects an element and moves the cursor in the desired direction. The motion simulates a pushing action, and all objects in contact in the direction of motion are grouped with the selected object and moved with it. The remaining objects stay in place, and form their own new group(s). This has the desired effect that groups are made and broken with extreme ease. Figure 3 shows the behavior of the manipulation techniques.

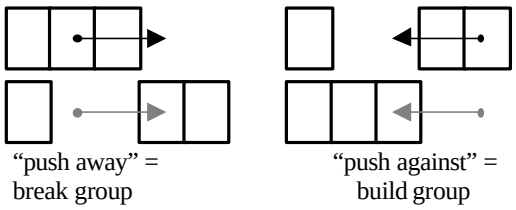


Figure 3: Dual group manipulation techniques (before & after mouse drag)

3 Constraint Satisfaction

For virtual constraints, binding and offer areas both have a polygon and vector, which represent their effective areas and orientation. A binding area is satisfied by an offer area by aligning their orientation vectors and by translating the binding polygon so that it lies within the offer polygon. If after rotation and translation the binding polygon is not completely enclosed by the offer polygon, then the binding area is not bound to the offer area.

Dual constraints are defined as points instead of areas (like binding and offer areas). To satisfy a dual constraint between two objects D1 and D2, we match up their dual points and rotate their orientation vectors in opposing directions. Since the dual constraint is defined as a point, we permit any translation of D1 that will bring the two constraint points together. There is no special technique required to deal with dual-constraint cycles. All dual group members are treated as siblings in the scene DAG, with a special dual-group parent node.

To constrain an object, we attempt to satisfy all of its binding areas and dual constraints. For each binding area and dual constraint of an object, we search through the scene to find potential satisfying offer areas and dual constraints. To prevent objects from jumping large distances to satisfy constraints, we only consider constraining an object to offer areas and dual constraints that are close to the object being constrained. Closeness is relative to object size, therefore we consider only objects that are within a sphere with a radius that is twice the radius of the sphere bound of the object.

Using this heuristic, constraints remain unsatisfied until an object is moved close to a valid offer area. For each binding area, if there are multiple satisfying offer areas, the closest satisfying offer area found is chosen. The object is moved to connect the binding and offer areas. The bound object then becomes a child of the offering object in the scene graph, and the search is repeated for the next binding area.

Once an object is constrained, its motion is restricted such that the binding areas of the object always remain in contact with the associated offer areas. This essentially removes degrees of freedom from object manipulations. Constraints can be broken with ease by simply pulling an object away from its associated offer area.

4 MIVE Constraint Environments

MIVE offers three different levels of constrained working environments: unconstrained, partially constrained, and fully constrained. Each of the environments is described in more detail below.

Many scene modelers, such as standard 3D computer aided design (CAD) programs, use no constraints. They provide only an interface to manipulate all six degrees of freedom of an object. To enable an effective comparison with this class of systems we implemented a mode in MIVE, which does not exploit constraints. We call this the Unconstrained (UC) mode. In this mode, the user can place an object anywhere in 3D space, with any orientation. When working in UC mode, we keep collision detection enabled to facilitate placing an object against another, and to prevent interpenetration.

Previous systems have used a more general constraint environment, where objects only know that they must lie on a horizontal and/or vertical surface, such as the Object Association system [5]. We hypothesize that this makes interaction less intuitive because it gives the user less control over how objects behave in the scene. A chair, for example, can be placed on a table, bed, refrigerator, or any other horizontal surface. We have implemented such a constraint system in MIVE and call it the Partially Constrained (PC) mode. This mode realizes a slightly improved version of Object Associations as it performs collision detection and has some support for virtual constraints (e.g. for chair constraint under a table). Also, unlike in Object Associations, satisfied constraints are explicitly remembered in PC mode, and no searching is necessary to maintain them.

Each object in the MIVE system has a set of constraints associated with it. For example, a table has a constraint on its base that causes it to stand on the floor, and a constraint on its top that allows other objects (i.e., a telephone) to lie on its surface. When the table is added to the scene, it will always be positioned on the floor – even if the user clicks on a wall! When moved or rotated, the table remains on the floor and objects lying on its top surface move/rotate with it. This is the default constraint mode in MIVE. We call this the Fully Constrained (FC) mode. Dual constraints are also enabled in this mode.

4.1 Interaction

The interface for MIVE was designed to be as simple and uncluttered as possible. All interactions between the participant and the program are done using a 3-button mouse.

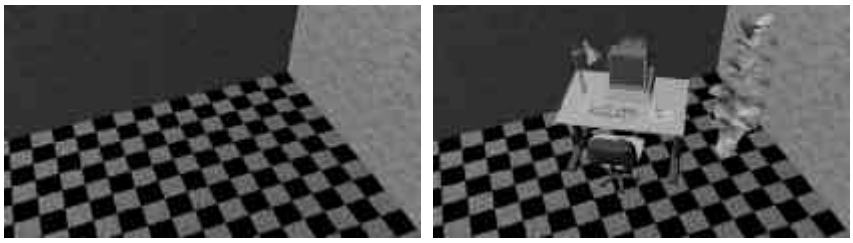
The FC and PC modes use only two of the three buttons. The left mouse button is used to move objects by clicking and dragging to the desired new location. The middle mouse button is used to rotate the objects. The third mouse button is currently unused in these modes.

The UC mode uses the right and left mouse buttons to move objects in 3D space, and the middle mouse button to perform an Arcball rotation [9] on the object. With this button assignment, it is possible to place any object in any orientation in 3D space.

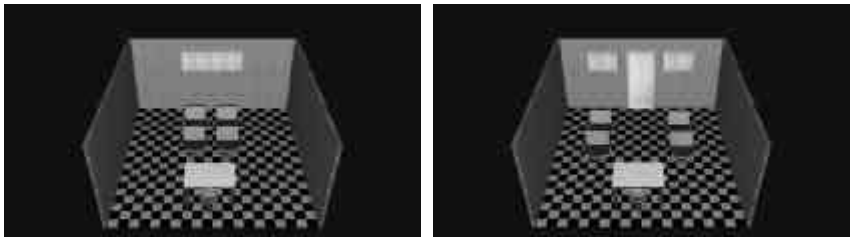
5 User Testing

We designed experiments to evaluate the differences between the three different constraint environments with a standard 2D user interface (mouse & screen). Five simple tasks were chosen to test performance in different contexts:

T1) Moving a telephone from a table to a desk of a different height.
T2) Pulling a chair out from under one table, and placing it under a different table on the other side of the room.
T3) Creating a simple scene consisting of a table, lamp, computer, chair and plant.
T4) Moving three cabinets from one wall to a second wall.
T5) Splitting a group of cabinets and a group of chairs into two, and adding a door between the cabinets.



Task 3: Create a scene



Task 5: Modify a scene

We choose these tasks because together they represent a majority of the operations to create and manipulate 3D scenes. Both T1 and T2 test the manipulation of a single object. T1 necessitates both a 3D translation and rotation. T2 requires only a 2D translation and a rotation with the additional difficulty of obstacles. T3 investigates the performance of scene construction. Dual Constraints were tested in T4 and T5. T4 involves a simple re-constrain of a group of cabinets, while T5 analyzes how groups are split.

5.1 Participants

Fifteen volunteers (thirteen males, two females) participated in this experiment. Participants were computer science students with different experience and back-

grounds, different computer skills and different degrees of exposure to 3D computer graphics. The average age was twenty-three.

5.2 Interface

The MIVE interface (figure 4) consists of three windows: the scene window, the object selection window, and the button window. The scene window sits on the right hand side of the screen. The participant directly interacts with objects in the scene window by clicking and dragging.

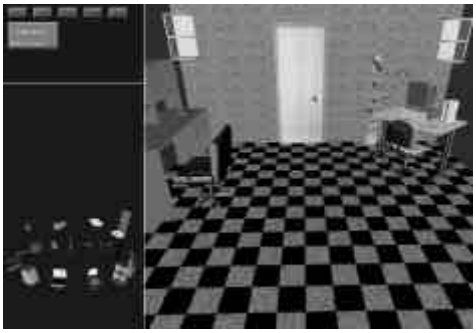


Figure 4: The user interface for MIVE

The lower left-hand corner shows the object selection window. Objects are positioned on an invisible cylinder, which is rotated by clicking any mouse button within the window and dragging left and right. Objects are added to the scene window by simply clicking on the desired object in the object selection window, and clicking on the desired location to add it in the scene window. The upper left-hand corner of the window contains buttons for various tasks, such as undoing the previous operation, or quitting the program.

5.3 Procedure

A five-minute tutorial was given prior to the testing, at which time the experimenter gave the participant instructions on how to use the system and how the mouse works in each of the three constraint modes. Each participant was then allowed to experiment about ten minutes with the system before testing started.
Each test began with the participant sitting in front of a computer monitor with a scene displayed. A target scene was displayed on an adjacent monitor, and the participant was instructed to make the scene on their screen look like that in the target scene. The experimenter supervised the participant, and when the task was complete, the supervisor instructed the participant to continue to the next task.
Each participant performed each of the five tasks in each of the three constraint systems. The order that the participant performed the tasks, and the used constraint system, was chosen using a Latin square method. Each participant completed five tasks in one constraint system, and then completed the same tasks (in the same order) in a second constraint system, then a final time in the third constraint system. All tests were done in a single session, which took thirty-five minutes on average.

When the participants completed all of their tasks, they were given a questionnaire on their preference among the different interaction modes in the system.

6 Results

At the end of each experiment task, completion time and the modified scene were stored. Basic statistics and a three by five repeated measures ANOVA was performed with statistical software.

Figure 5 summarizes the results of our user test. The (thick) center line of a box shows the median, the (thin) dotted line is the mean, the box itself indicates the 25th and 75th percentile and the 'tails' specify the 10th and 90th percentile. All statistical values in this publication are reported at alpha = 0.05.

The analysis of variance showed clear main effects for task completion time ($F_{2,112}$ = 42.2, p < .0001) and accuracy ($F_{2,112}$ = 20.55, p < .0001). The statistical power of both tests is larger than 0.9999. Post-hoc comparisons show a clear difference between UC mode and the other two modes in both time and accuracy. The results are detailed in the following subsections.

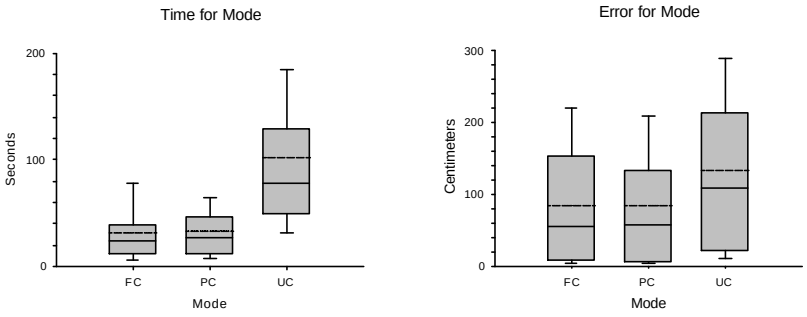


Figure 5: Box-plots for different modes.

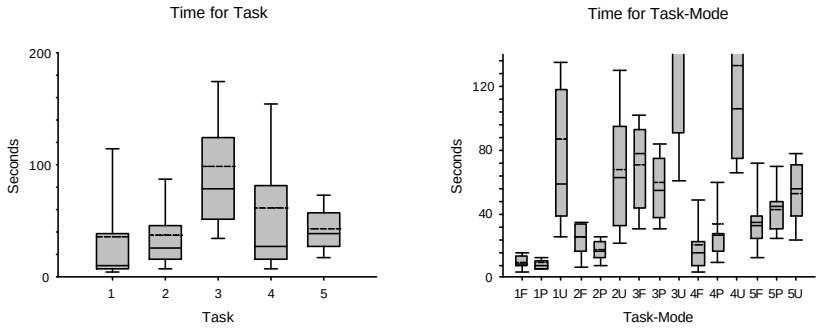


Figure 6: Box-plots for different tasks.

6.1 Performance and Accuracy

The mean completion time (variances are shown in brackets) for FC was 32.15 seconds (27.75) and 32.65 seconds (29.13) for PC. The mean time for the UC mode is 101.62 seconds (77.47). There is no significant difference between the FC and PC mode, while the difference for the UC mode is significant with $p < .0001$.
Accuracy was measured by summing the Euclidean distances in centimeters between each of the object centers in the participant's result and the target scene. The mean sum of distances for FC was 84.13 cm (88.02) and 83.09 cm for PC (88.58). The mean distance sum for the UC mode is 131.79 cm (126.2). There is no significant difference between the FC and PC mode, while the difference for the UC mode is significant ($p < .0005$).

6.2 Task

Figure 6 shows that T3 (scene creation) took the most time. The difference between the UC mode and the other modes is quite noticeable for the first four tasks. For T5 the difference between UC and FC is significant with $p < .002$, while there is no significant difference between the two other pairs. The ratio between the FC and UC modes is as follows: Test T1 ranks the highest with 8.7, followed by T4, which has a ratio of 6.65. Test T2 and T3 have slightly lower ratios (2.7 and 2.3 respectively), and finally T5 has a ratio of 1.5.

6.3 Questionnaire

Questions evaluated participant preferences between each combination of two modes on a five point Likert scale. The results show a clear preference for the FC mode. The mean value of a UC (1) vs. FC (5) comparison is 4.8, between UC (1) and PC (5) 4.6 and between PC (1) and FC (5) 3.93.

7 Discussion

The above analysis of the user test data supports the following observations:
First, unconstrained manipulation (UC) is definitely slower than the other two modes. For scene creation we observe a factor of roughly 2.3, and for most manipulation techniques, the factor is higher still. Moreover, UC manipulation is at least 50% less accurate than the other two modes. We believe that the difficulty in UC manipulation lies in the fact that it is hard for the user to visualize the exact positioning of objects in 3D with a 2D output device, hence positioning takes much more time.
Most users took the most time in T3, constructing the scene. We believe that this is due to the time it takes to 'retrieve' an object. When adding an object the user must first locate it within the object selection window. For this test we tried to keep the number of objects in the window to a minimum (around fifteen), but judging from our observation logs, identifying and selecting an object still takes almost half the time.
Many users did not use dual constraint grouping, and preferred to move objects individually instead of as groups. This occurred despite the fact that the users were

introduced to dual constraints during the tutorial. We believe either the users did not feel comfortable using the dual constraints, or they simply forgot about them. The mean times for T5 in each of the three modes are relatively close to each other compared to the other tasks. The reason is that in T5 all of the objects only need to be moved in a horizontal plane, and no rotations are needed. In the UC mode, translations along a horizontal plane are simple, so the steps needed to complete the task in the unconstrained system were the same as those in the other two systems. The significant difference between PC and FC is explained by the fact that all participants, who actually used the dual constraints, performed much better in FC mode. From these results we conclude that more investigation was needed.

An informal study, where the introduction to the test placed more emphasis on dual constraints, was performed after the user test reported above. The preliminary results indicate that dual constraints are at least a factor of two faster for the manipulation of object groups.

The partially constrained system was statistically identical to the fully constrained system in time and accuracy. Initially, we found this surprising, as the participants clearly preferred the fully constrained system in the questionnaire. Further analysis revealed that in the tested context PC and FC are not different, because the semantic information in FC mode does not affect performance. The visual feedback is different – in one case, the telephone moves over the floor, in the other it appears to jump over the intervening gap. However, the manipulation time is the same in both cases. Nevertheless, users felt the objects in the scene behaved more intuitively when using the fully constrained system.

8 Conclusion

In this publication, we presented the first evaluation of 3D constraint interaction techniques with a 2D user interface. We showed that in most cases constraints provide more than a factor of two speed-up combined with a significant increase in accuracy. The unconstrained system was significantly slower and less accurate than the other two systems, and was the least preferred of the three systems.

Furthermore, we presented a system that allows users to easily manipulate a 3D scene with traditional 2D devices. The constraints encapsulate the user's expectations of how objects move in an environment. Thus the system introduced here enables an intuitive mapping from 2D interactions to 3D manipulations.

We introduced and extended version of Object Associations, which utilized constraint maintenance, virtual constraints, and collision detection.

The benefits of our interaction techniques become very apparent when one compares the simple MIVE user interface with the complex 3D user interface in commercial packages. We can only hypothesize of comparison of our system with e.g., *Maya*, but are confident that it is clearly easier to learn our user interface due to the reduced complexity.

Most systems that use 3D input devices support only unconstrained manipulation. Based on the observation of Poupyrev that many 3D manipulation techniques can be approximated with 2D techniques we hypothesize that the techniques presented here are directly beneficial to scene manipulation with 3D input devices.

8.1 Future Work

Our vision is to create a system that makes adding objects to a scene as quick and easy as possible.
In the future, we will perform a user study that compares the relative performance of 2D vs. 3D devices in a constraint based 3D manipulation system. As this study establishes a benchmark for 2D devices we can use the results presented here as a basis for a fair comparison between 2D and 3D devices.
As a significant amount of time is spent selecting objects, we are currently investigating different methods to speed up object selection. Possibilities include speech recognition, the use of hierarchies, and different presentation techniques.
Easy constraint creation is another topic for future research. Many published constraint systems suffer from the fact that the definition of constraints is a complex task and often requires intimate knowledge of the associated algorithms. This prevents users from easily integrating new objects. We are currently investigating ways to automatically define constraints for new objects.

References

1. Bier, E.A., and Stone, M.C. Snap-dragging. *SIGGRAPH 1986 proceedings*, ACM Press, pp. 233-240.
2. Bier, E.A. Snap dragging in three dimensions, *SIGGRAPH 1990*, pp. 193-204.
3. Borning, A., Freeman, B., Ultraviolet: A Constraint Satisfaction Algorithm for Interactive Graphics, *Constraints: An International Journal*, 3, 1-26, 1998.
4. Bowman, D., Kruijff, E., LaViola, J., Mine, M., Poupyrev, I., 3D user interface design, *ACM SIGGRAPH 2000, Course notes # 36, 2*000.
5. Bukowski, R., and Sequin, C. Object associations. ACM *Symp. Interactive 3D Graphics 1995*, 131-138.
6. Gleicher, M., Integrating Constraints and Direct Manipulation. *Symp. on Interactive 3D Graphics*, 1992, pp. 171-174.
7. Goesele, M, Stuerzlinger, W. Semantic constraints for scene manipulation. *Proc. Spring Conference in Computer Graphics* 1999, pp. 140-146.
8. Pierce, J., Forsberg, A., Conway, M., Hong, S., Zeleznik, R. *et al.*, Image plane interaction techniques in 3D immersive environments. *Proceedings of ACM Symp. on Interactive 3D Graphics*. 1997. pp. 39-43.
9. Shoemake, K., ARCBALL: A user interface for specifying three-dimensional orientation using a mouse, *Graphics Interface*, 1992, pp. 151-156.
10. Smith, G., Stuerzlinger, W., On the Utility of Semantic Constraints. To appear in *Eurographics workshop on Virtual Environments*, 2001.
11. Zanden, B., Myers, B., Giuse, D., Szekely, Integrating Pointer Variables into One-Way Constraint Models, *ACM Transactions on Computer-Human Interaction*, 1(2), 161-213, 1994.