Unconstrained vs. Constrained 3D Scene Manipulation

T. Salzman, S. Stachniak, W. Stuerzlinger

Dept. of Computer Science, York University, 4700 Keele Street, Toronto, ON, M3J 1P3, Canada
{salzman | szymon | wolfgang}@cs.yorku.ca

Abstract. Content creation for computer graphics applications is a very time-consuming process that requires skilled personnel. Many people find the manipulation of 3D object with 2D input devices non-intuitive and difficult. We present a system, which restricts the motion of objects in a 3D scene with constraints. In this publication we discuss an experiment that compares two different 3D manipulation interfaces via 2D input devices. The results show clearly that the new constraint-based interface performs significantly better than previous work.

1. Introduction

Computer Graphics applications, such as physical simulations, architectural walk-throughs, and animations, require realistic three-dimensional (3D) scenes. A scene usually consists of many different objects. Objects and scenes are usually created with a 3D modeling system. Many different commercial systems are available for this purpose. But in general these products are difficult to use and require significant amounts of training before than can be used productively. Realistic 3D scenes often contain thousands of objects, and complex interfaces can leave the user feeling lost. For example, products such as Maya and 3D Studio Max have dozens of menus, modes and widgets for scene creation and manipulation, which can be very intimidating for an untrained user. Our efforts address these difficulties.

The focus of this work is on the creation of complete 3D scenes, not on the generation of 3D geometric models. We rely on a library of predefined objects from which the user can choose while constructing a scene. The challenge is to provide the user with a technique to easily add objects to the scene and then position them relative to each other.

Interacting with objects in a 3D environment is difficult because six independent variables must be controlled, three for positioning and three for orientation. Even when the task of object manipulation is decomposed into the two separate tasks of positioning and orientation, the user interface is still not intuitive. This is due to the problem of mapping a device with 3 modes (buttons) with 2 degrees of freedom each to 2 different manipulation techniques with 3 degrees of freedom each.

3D input devices, such as a space-ball or tracking devices, make direct interaction with objects in a 3D scene possible, but such devices are uncommon and often expensive, or require training and experience to use. Similarly, 3D output devices

such as head mounted displays or shutter glasses are a possible solution, but again are uncomfortable, uncommon, or expensive.

Our observations of humans rearranging furniture and planning environments indicate that humans do not think about scene manipulation as a problem with 6 degrees of freedom. The rationale is that most real objects cannot be placed arbitrarily in space and are constrained in their placement by physics (e.g. gravity) and/or human conventions (ceiling lamps are almost never placed permanently onto the floor or onto chairs). This lets us to believe that an interface that exposes the full 6 degrees of freedom to the user makes it harder for average persons to interact with virtual environments. Many real objects have a maximum of 3 degrees of freedom in practice – e.g. all objects resting on a plane. Furthermore, many objects are often placed against walls or other objects, thus further reducing the available degrees of freedom. This in turn leads us to believe that a two-dimensional (2D) input device such as a mouse may be sufficient to manipulate objects in a virtual environment. Furthermore, in previous work Poupyrev et al. [13] suggested that all ray-casting techniques can be approximated as 2D techniques as long as objects are relatively close to the viewer. This further supports our argument that a 2D input device is sufficient to manipulate most real objects in a 3D environment.

In our system, information about how objects interact in the physical world is used to assist the user in placing and manipulate 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. For example, a photocopier must stand on the floor at all times. When a user interacts with the photocopier by translating or rotating it in the scene, it never leaves the floor. This concept of constraints makes manipulating objects in 3D with 2D devices much simpler.

1.1. Previous Work

Previous work can be classified into two categories: those that use 2D input devices and those that use 3D input devices.

For 2D applications Bier introduced Snap-Dragging [1] to simplify the creation of line drawings in a 2D interactive graphics program. The cursor snaps to points and curves using a gravity function. Bier emphasized the importance of predictability in the system; i.e. that the interface should behave as the user expects it to. Hudson extended this idea further to take non-geometric constraints into account and called his technique semantic snapping [9]. Bier subsequently generalized Snap-Dragging to 3D environments [2]. Relationships between scene components were exploited to reduce the size of the interface and the time required to use it Interactive transformations are mapped from the motion of the cursor, which snaps to alignment objects for precision. The main features of this system are a general purpose gravity function, 3D alignment objects, and smooth motion affine transformations of objects. Houde introduced another approach that uses different handles on a box surrounding each object to indicate how it can be manipulated [8].

Bukowski and Sequin [6] employ a combination of pseudo-physical and goal-oriented properties called Object Associations to position objects in a 3D scene using 2D devices (mouse and monitor). They use a two-phase approach. First, a relocation

procedure is used to map the 2D mouse motion into vertical or horizontal transformations of an object's position in the scene. Then association procedures align and position the object.

Although fairly intuitive, their approach has a few drawbacks. Firstly, associations apply only to the object that is currently being moved and are not maintained after the current manipulation. Also, 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. Furthermore, cyclical constraints are not supported.

The fundamental ideas of the Object Associations approach were incorporated into the work of Goesele and Stuerzlinger [10]. Each scene object is given predefined offer and binding areas. These areas are convex polygons, which are used to define constraining surfaces between objects. For example, a lamp might have a binding area at its base and a table might have an offer area defined on its top surface. In this way a lamp can be constrained to a tabletop. To better simulate the way real world objects behave, a constraint hierarchy is used to add semantics to the constraint process. This is a generalization of the Object Association approach. Each offer and binding area is given a label from the hierarchy. A binding area can constrain to an offer area whose label is equal to or is a descendant in the constraint hierarchy to the label of the binding area. In this way a monitor may be defined to constrain to a tabletop, but never to the wall or floor. Also, collision detection is used to add realism to the interface.

Drawbacks are: Once a constraint has been satisfied, there are no means to un-constrain or re-constrain the object to another surface. Further, the constraint satisfaction search is global, in that an object will be moved across the entire scene to satisfy a constraint, an often-undesirable effect for the user, especially because constraints cannot be undone.

A number of authors have investigated the performance of object manipulation with 3D input devices. One of the first was Bolt in 1980 [3]. Most recently Bowman [4], Mine [11], and Pierce [12] proposed different methods that can be applied in a variety of settings. Poupyrev recently also addressed the problem of 3D rotations [14]. For a more complete overview over previous work in this area we refer the reader to [5]. Most relevant to the research presented here is the work by Poupyrev et al [13]. There, different interaction methods for 3D input devices are compared. The item that is of most interest in this context is that the authors suggest that all ray-casting techniques can effectively be approximated as 2D techniques (see also: [5]). This nicely supports our observation that for most situations a user interface that utilizes only a 2D input device may be sufficient to effectively manipulate objects in 3D.

The SmartScene system [16] by Multigen is a Virtual Reality system that employs a 3D user interface based on Pinch-gloves. Object behaviors that define semantic properties can be defined to simplify object manipulation.

To our knowledge, there is no previous work that compares 3D unconstrained and constrained manipulation,

2. The MIVE System

The MIVE (Multi-user Intuitive Virtual Environment) system extends the work done in [10] by improving the way existing constraints behave, and adding new useful constraints. This work concerns only the interaction of a single user, therefore we disregard the multi-user aspects of the system here.

The constraint relationships are stored in a directed a-cyclic graph called the scene graph. Figure 1 depicts a simple scene, and it's associated scene graph. When an object is moved in the scene, all of its 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 re-constrained in with ease by simply clicking on the desired object, and pulling away from the existing constraint to break it. This allows us to dynamically change the structure of the scene graph. Figure 2 shows the same scene as Figure 1 after the chair has been pulled away from the large table, and dragged under the smaller table.

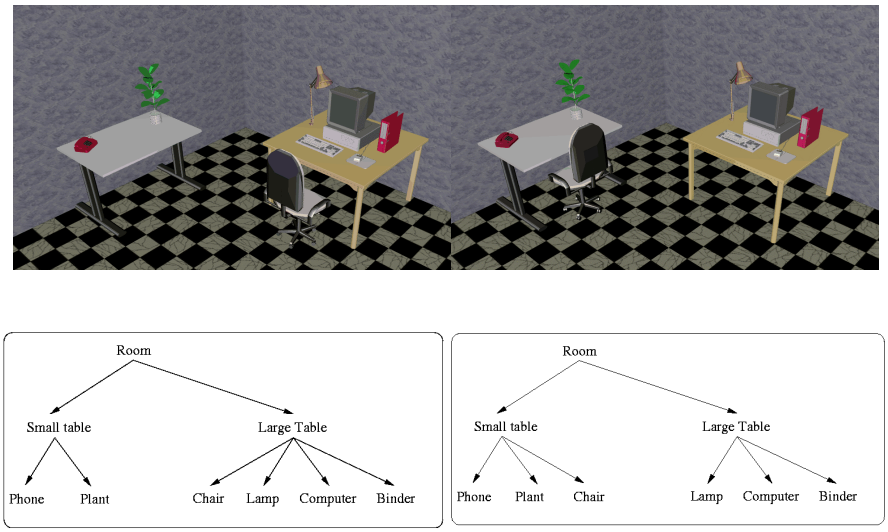

Fig. 1. A Scene and its associated Scene Graph. Links describe constraint relations. Both the initial state (left) as well the state after the chair has been moved (right) are shown.

2.1. MIVE Constraint Environments

Each object in the MIVE system has a set of constraints associated with it. For example, a table would have 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 phone) to lie on its surface. When the table is added to the scene, it will always lie on the floor. When

being moved or rotated the table remains on the floor, and any objects lying on its top surface will move/rotate with it. This is the default constraint mode in MIVE. We call this the Fully Constrained (FC) mode.

Other scene modelers, such as many 3D computer aided design (CAD) programs, use no constraints whatsoever. They provide only an interface to manipulate all six degrees of freedom of an object. We believe that interaction with objects in an unconstrained environment such as this, is much more difficult for the user. To enable an effective comparison we implemented a mode in MIVE, which does not exploit constraints. We call this the Unconstrained (UC) mode.

3. User Testing

The main objective of the work presented here is to evaluate if a constraint-based system provides a more effective 3D manipulation interface. We asked participants in our study to design scenes in both the FC as well as the UC mode.

3.1. Method

Two representative tasks were chosen to evaluate our objective.

T1) Creation of a scene: See Figure 4 for the target scene image.

T2) Modification of a scene: See Figure 5 and 6 for initial respective final scene.

We choose these two tasks because together they exploit a majority of the operations needed to create and manipulate 3D scenes.

Both scenes consist of 30 objects. For the creation task (T1) the participant can select the objects from a window that displays all 30 objects.

We chose not to attempt a comparison with a user interface that utilizes 3D input devices as these are usually used in a very different setting (standing user and large screen stereo projection or HMD). Although very interesting, this introduces several additional factors into the user tests that may not be easy to account for. Furthermore, even if good 3D output equipment was readily available, in our tests stereo output should not constitute an important factor as most objects are placed clearly in relation to other objects such as a phone on a desk. Also, the test scenes in our environment are limited in range, therefore we believe that stereo viewing will not provide important visual cues.

For task T1 no navigation was necessary to complete the task. For task T2 users had to navigate to move one of the objects that was hidden behind another.

3.2. Participants

Fifteen volunteers (3 female, 12 male) participated in this experiment. Half of the participants were computer science students, half had other backgrounds (including 4 persons with artistic talents). Most participants had at least average computer skills, but only 5 had any exposure to 3D scene construction. The ages of the participants ranged from between 20 to 43 years, the average age being 23.

3.3. Apparatus

The MIVE interface was designed to be very simple. Figure 2 shows the full user interface of the MIVE program running with its default object list.

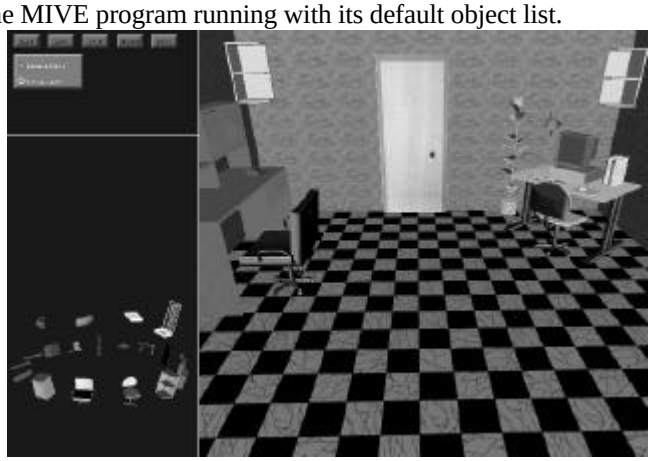

Fig. 2. User Interface of the MIVE system.

The MIVE interface 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 them.

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 performing tasks such as loading or saving the scene, deleting an object, undoing the previous operation, or quitting the program. There is also a radio button which can be used to switch between interaction and navigation mode. This functionality was disabled for the tests in this publication.

The MIVE system is implemented in C++ and runs on an SGI Onyx2 running IRIX 6.5. It is based on the Cosmo3D [7] scene graph API.

3.3.1. 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 mode uses only two of the three buttons. The left mouse button is used to move objects by clicking and dragging them to the desired new location. The middle mouse button is used to rotate the objects. The third mouse button is currently unused in this mode.

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 [15] on the object. Using

these three buttons it is possible to place any object in any orientation in 3D space. Although constraints are disabled in this mode, collision detection remains active so that objects do not interpenetrate.

3.4. 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 worked in each of the two modes. Participants were allowed to familiarize themselves with the interface for 5 minutes.

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. When the participant deemed that the scene resembled the target closely enough, the participant quitted the program after checking back with the experimenter.

Each participant was asked to perform each the two tasks in each of the two constraint systems for a total of 4 trials for each participant. The starting task for each participant was randomly selected. To counterbalance the design, subsequent tasks were ordered so that a UC task followed a FC task and vice versa. All tests were done in a single session, which took slightly more than one hour on average.

For each of the tests, we recorded the time taken by the participant, the number of actions and the accuracy of the final scene. Accuracy was measured by summing the distances in centimeters between each of the object centers in the user's final result and the target scene.

When the participants had completed all of their tests, they were given a questionnaire that posed questions relating to their preference among the interaction modes in the system.

4. Analysis

At the end of each experiment task completion time and the modified scene was stored. The Euclidean distance between the participant's solution and the reference solution was computed later on. Finally, basic statistics and a 2 by 2 repeated measures ANOVA was performed with statistical software.

4.1. Adjustments to Data

The data for two participants was excluded from the analysis. One participant tried to visually match the results to the target scene at the level of pixel accuracy by repeatedly navigating back and forth. The experiment was aborted after 1½ hours with 2 tasks completed. Another participant could not understand how Arcball rotations in the 3D unconstrained (UC) mode worked and got too frustrated.

No other adjustments were made to the collected data.

4.2. Computed Formulas

Errors are specified as the *sum* of Euclidean distances for all objects, as 3D distances are themselves quadratic measures. We ignore rotation because no ideal measure for rotation differences exists to our knowledge. Moreover it is hard to find a meaningful combination of translation and rotation errors into one number.

5. Results

Figure 3 summarizes the result of our user test. The center line of the boxes shows the mean, the box itself indicates the 25^{th} respective 75^{th} percentile, and the 'tails' of the boxes specify the 10^{th} respective 90^{th} percentile. All statistical values in this publication are reported at alpha = 0.05.

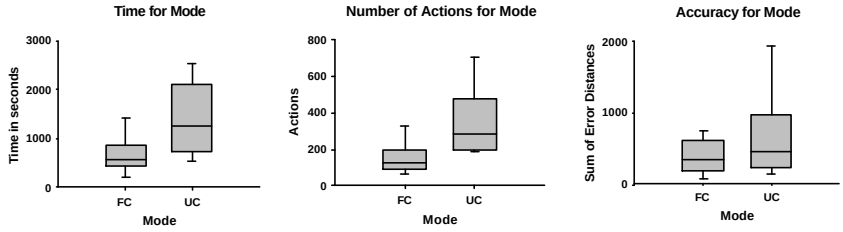

Fig. 3. Box-plots of times, actions and accuracy for UC and FC modes.

The analysis of variance showed clear main effects for task completion time ($F_{1,12} = 26.36$, $p < 0.0005$), number of actions ($F_{1,12} = 16.0$, $p < 0.005$) and accuracy ($F_{2,12} = 7.31$, $p < 0.05$) with respect to the two different interaction modes. The statistical power of the first two tests is larger than 0.95, the last test has only a power of 0.699.

The mean completion time (variances are shown in brackets) for FC was 719.7 seconds (543.9) and 1621.4 seconds (1233.1) for UC. This shows that FC is faster by a factor of almost 2.3.

The mean number of actions for FC was 157.96 (110.9) and 395.92 (318.2) for UC. In terms of user actions UC requires more than 2.5 times the number of actions.

The mean sum of error distances for FC was 381.3 cm (242.7) and 762.1 cm for UC (880). This seems to indicate that objects are positioned more accurately in the FC system, although the statistical significance is not as strong as for the other two criteria.

Task T1 is significantly different from T2 in terms of time. (T1 took 30% more time). T1 is not significantly different from T2 in terms of the number of actions (289 vs. 264). Accuracy shows also no significant difference, although we observe that T1 has a slightly larger error on average.

5.1. Questionnaire

Questions evaluated participant preferences between each combination of two modes on a 5 point Likert scale. The results show a clear preference for the FC mode. Users found it much easier to manipulate objects when they felt that those objects acted as would their counterparts in the real world. Imposing constraints on objects abstracted difficult transformations that, to the untrained person, were hard to visualize. In particular, most people found rotations hard to grasp in unconstrained mode, since it was much more difficult to orient objects in full 3-space than it was in the more limited yet more intuitive FC mode. Users were noted to find the spinning object selection menu to their liking for it gave quick access to all the objects without cluttering precious desktop space. The only complaint voiced about the menu was the selection of small objects, as users found them hard to select at times.

To aid in the positioning of objects in UC mode, users were quick to point out the benefits of shadows. A visual cue such as a shadow would be necessary to aid in object placement of a fully UC system, but imposes many new problems, i.e. positioning a light source. A FC system eliminates the need to judge height distances from one object to another, and therefore eliminates this problem. The absence of additional (potentially confusing) visual cues makes onscreen manipulation.

6. Discussion

The above analysis of the user test data supports the following observations:

First, unconstrained manipulation (UC) is definitely slower than the FC. For scene creation we observe a factor of roughly 2.3. Moreover, we can say that UC manipulation is at least 50% less accurate than the FC mode.

This current test involved complex tasks. In effect this prohibited us from determining precisely the relative performance of the two modes. Another complicating influence is that T2 necessitates navigation.

General observations of users during the tests led to a determination that the majority of inaccuracies and time delay in UC is due to problems in rotation. When constraints are imposed, rotations become much more intuitive, as the user interface restricts rotations to the free axis, which in turn increases speed and accuracy.

Participants seemed to have taken the most time in T1, constructing the scene. We believe that this may be because of the time it takes to ‘retrieve’ an object. When adding an object the user has to 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 30), but judging from our observations during the test identifying and selecting an object still takes roughly 3 seconds.

A constrained system, unlike an unconstrained system is much more intuitive for users who have little to no experience in 3-dimensional manipulation. Users were able to construct scenes with ease only several minutes after having been introduced to the FC system, yet in UC mode issues relating to 3-dimensional manipulations would often arise.

The main objective of the user test presented here was to determine if constrained scene manipulation is helpful to the user and if it makes a difference. The results indicate that this is true in practice. The work presented here did not look into a comparison with a system that is equivalent to the Object Associations [6] system. This system uses only horizontal and vertical constraints. It is unclear at the moment how powerful these two constraints are compared to the 'semantic' constraints in the MIVE system. An small experiment by the authors with an initial implementation showed that the difference in performance between the two systems is small. Future work will look further into this issue.

7. Conclusion

In this publication we presented a system that allows users to easily manipulate a 3D scene with traditional 2D devices. The MIVE system is based on constraints, which enable an intuitive mapping from 2D interactions to 3D manipulations. Phrased differently, the constraints and associated manipulation techniques encapsulate the user's expectations of how objects move in an environment. The user tests showed that the interaction techniques presented here provide clear benefits for 3D object manipulation in a 2D user interface. The unconstrained system was significantly slower and less accurate than the system that uses constraints. Users unanimously preferred the constrained system to the unconstrained one.

Interestingly enough, 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 are fairly confident that we can speculate that the techniques presented here are directly beneficial to scene manipulation with 3D input devices. To our knowledge, this work is the first to compare 3D unconstrained and constrained manipulation.

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 such as AutoCAD, Maya or 3DStudio Max that are also based on 2D input devices. We can only hypothesize at the outcome of a test comparing our system with e.g. Maya, but are confident that it is clearly easier to learn our user interface due to the reduced complexity. In fairness, we need to point out that these packages are also capable of object creation and the specification of animations, which our system does not currently address.

7.1. Future Work

Our vision is to create a system that makes adding objects to a scene as quick and easy as possible. The user test involved complex tasks that had to be accomplished by the user. In effect this prohibited us from determining the relative performance of individual elements of the user interface precisely. Therefore, future work will focus on determining the relative performance of the individual techniques. As mentioned in the discussion, we will test the developed techniques against an implementation that mimics the interaction methods in the Object Association system [6].

The current set of user tests was designed in such a way that users themselves determine when they have completed the desired tasks. This method was chosen when we realized that it is hard to automatically implement an adequate measure of scene similarity. Factors would need to be taken into account are differences in position, rotation, location in the scene graph and the fact if an object is correctly constrained or not. A viable alternative that we will pursue in the future is that the test supervisor determines when the user has accomplished the task.

Also we noticed that a considerable amount of time was spent selecting new objects in the object menu. This issue warrants further investigation. Furthermore, small objects in the rotating menu are hard to select, which needs to be addressed, too.

Also, we will study how constraint based scene construction performs in immersive virtual environments such as a CAVE.

8. Acknowledgements

Thanks to N. Toth, G. Smith and D. Zikovitz for their help. Furthermore, we would like to thank all persons who volunteered for our user tests.

References

1. Bier, E.A., and Stone, M.C. Snap-dragging. ACM SIGGRAPH 1986, pp. 233-240.
2. Bier, E.A. Snap Dragging in Three Dimentions, ACM SIGGRAPH 1990, pp. 193-204.
3. Bolt, R., Put-that-there. SIGGRAPH 1980 proceedings, ACM press, pp. 262-270.
4. Bowman, D., Hodges, L. An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. Proceedings of ACM Symposium on Interactive 3D Graphics, 1997, pp. 35-38.
5. Bowman, D., Kruijff, E., LaViola, J., Mine, M., Poupyrev, I., 3D User Interface Design, ACM SIGGRAPH 2000 course notes, 2000.
6. Bukowski, R.W., and Sequin, C.H. Object Associations. Symposium on Interactive 3D Graphics 1995, pp. 131-138.
7. Eckel, G., Cosmo 3D Programmers Guide. Silicon Graphics Inc. 1998.
8. Houde, S., Iterative Design of an Interface for Easy 3-D Direct Manipulation, ACM CHI ’92, pp. 135-142.
9. Hudson, S., Adaptive Semantic Snapping – A Technique for Semantic Feedback at the Lexical Level, ACM CHI ’90, pp. 65-70.
10. Goesele, M, Stuerzlinger, W. Semantic Constraints for Scene Manipulation. Proceeedings Spring Conference in Computer Graphics (Budmerice, 1999), pp. 140-146.
11. Mine, M., Brooks, F., Sequin, C. Moving Objects in Space: Exploiting Proprioception in Virtual-Environment Interaction. SIGRAPH 1997 proceedings. ACM press, pp. 19-26.
12. Pierce, J., Forsberg, A., Conway, M., Hong, S., Zeleznik, R., et al. Image Plane Interaction Techniques in 3D Immersive Environments. Proceedings of Symposium on Interactive 3D Graphics. 1997. ACM press. pp. 39-43.
13. Poupyrev, I., Weghorst, S., Billinghurst, M., Ichikawa, T., Egocentric object manipulation in virtual environments: empirical evaluation of interaction techniques. Computer Graphics Forum, EUROGRAPHICS '98 issue, 17(3), 1998, pp. 41-52.

14. Poupyrev, I., Weghorst, S., Fels, S. Non-isomorphic 3D Rotational Techniques. ACM CHI'2000, 2000, pp. 546-547.
15. Shoemake, K., ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse, Graphics Interface, 1992, pp. 151-156.
16. SmartScene promotional material, Multigen, 1999.

Appendix

As more of a joke than anything else we kept a profanity count during the test. The number of unprintable words uttered can be seen as a simple measure of user frustration. In FC mode users uttered 0.33 profanities on average during the test. UC mode featured an average of 3.53 profanities!

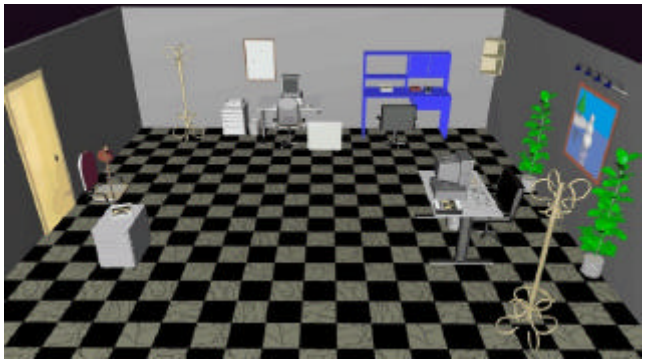

Fig. 4. Target Scene for T1) Scene Creation Task. Initial state is an empty room.

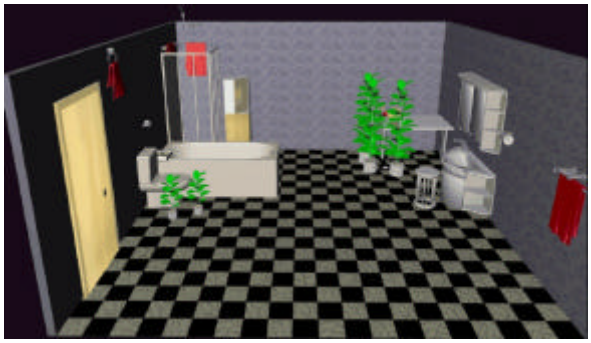

Fig. 5. Initial Scene for T2) Scene Modification Task

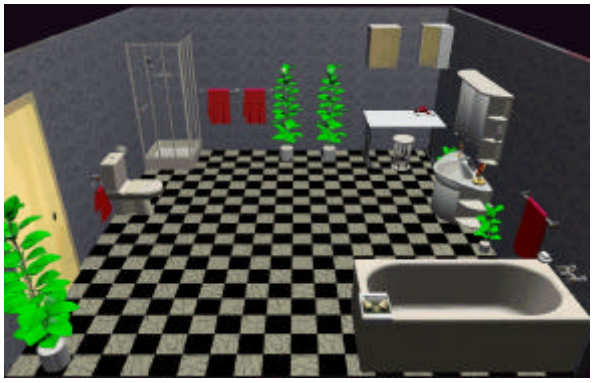

Fig. 6. Target Scene for T2) Scene Modification Task