Human-Computer Interaction -- INTERACT'03
M. Rauterberg et al. (Eds.)
Published by IOS Press, (c) IFIP, 2003, pp.319-326

Intelligent Manipulation Techniques for Conceptual 3D Design

Ji-Young Oh & Wolfgang Stuerzlinger

Computer Science, York University

Toronto, Ontario, Canada

http://www.cs.yorku.ca/~{jyoh,wolfgang}

Abstract: One of the main operations in conceptual 3D design is the rearrangement of single and composite objects. This paper presents a new conceptual 3D design system that affords easy manipulation of composite objects. We discuss several alternative manipulation techniques to separate complex parts off an existing model. Then, we present a new way to move such parts in a 3D scene, as well as an algorithm to place parts at arbitrary locations. Finally, we present and discuss the results of a user study of these manipulation techniques.

Keywords: Conceptual 3D Design, Interactive 3D Environment, 3D Manipulation.

1. Introduction

Easy creation of 3D designs is crucial for many application areas, such as architecture, industrial and mechanical design, creation of simulations and training environments, animation, and entertainment.

In general, the goal of any design activity is to produce a description of an artifact that fulfills a set of design problems. The main concern of designers is that design problems are often ill-defined, in a sense that they are abstract, poorly described, and require subjective interpretation.

To come up with new designs, designers usually go through the following stages: analysis of problems, conceptual design, embodiment of schemes, and the detailing stage (Cross, 2000). While the designers are going through this process, they progressively reduce the uncertainty of the problems by evaluating different solutions.

It is important to note that the conceptual design session has different characteristics from the later detailing session. While the detailing session requires high quality of a drawing without errors, the conceptual session requires a quick visualization to allow a designer to evaluate a concept. Designers repeatedly generate solutions and in turn identify problems in these solutions. As a consequence, designers want to visualize a solution with the least commitment to detail, and want to be able to quickly modify it based on their evaluation.

Sketching is one way to create conceptual designs. Recent research into the use of sketching in the design session argues that recognition of composite objects is one of the *main* mental operations involved in the design process (e.g. Purcell, 1996; Verstijnen, 1998). However, it is important to note that while sketching facilitates the recognition of composite objects, it does not afford the *manipulation* of such composites. Phrased differently, it is extremely important that conceptual design systems provide simple ways to modify and rearrange arbitrary parts of a scene.

This paper is about a new kind of conceptual design system targeted at the easy creation and modification of 3D scenes. A good real-life example for a conceptual 3D design system that affords easy creation and modification of scenes is Lego™, and it has already been used to prototype 3D models (Neale, 2002). Lego has the following properties:

- It allows to quickly create *approximate* 3D models. However, although approximate, practically everybody understands the design intent immediately.
- Only solid blocks exist, and they slide along each other, allowing for precise placement.
- A large variety of block sizes and shapes exist.
- Blocks connect to each other and form a stable 3D model.
- The ability to easily separate any part, modify it, and then re-assemble the model is an inherent feature.
- It is easy to learn and fun to use!

All of the above properties are clearly desirable in any conceptual 3D design system.

2. Previous Work

In previous work on 3D scene manipulation, the topic of composite object manipulation has rarely been considered. Several research projects used the paradigm of sketching to facilitate the easy creation of a scene. For recent examples see (Igarashi, 1999; Zeleznik, 1996; Pereira, 2000). Incremental modification is supported using Constructive Solid Geometry (CSG) operations. The only way to manipulate groups is via strictly hierarchical grouping or lassoing. A hierarchical relationship is usually established when one object is placed on another, e.g. a cup on a tabletop. Lassoing allows for more flexibility and can select groups of arbitrary objects. However, the accuracy of a lassoing gesture, a quick circling motion, is usually very limited.

Another approach is to create objects by filling space with blocks (similar to the idea of "painting in 3D"). One of the best examples is DDDoolz (Vries, 2002), an architectural conceptual design tool. While providing for easy scene creation, the authors also recognized easy scene modification as an important issue. In this system, a stroke creates a sequence of blocks along the stroke, which allows the user to create architectural elements by filling the space with blocks. Blocks are grouped together if the user assigns them the same color. The authors motivate their choice with the fact that different colors are used to visualize different architectural elements. However, in other application domains this may not be appropriate and is unnecessarily restrictive.

Several researchers also investigated how one can quickly assemble and rearrange a 3D scene if a library of predefined objects is available. Real-world behaviors such as gravity or collision are usually simulated to facilitate the task. One of the first was the object association system (Bukowski, 1995). This system allows populating a building with furniture, books, etc. Each object has pre-defined constraints that specify how an object can attach and move on horizontal or vertical surfaces. The constraints are also used to define a hierarchical scene graph, which affords composite object manipulation. Kitamura *et al.* (Kitamura, 1998) presented a scheme that dynamically generates constraints based on collisions with existing surfaces. This allows for natural object movement, except when many other objects are present in a region. The MIVE system (Stuerzlinger, 2002) extends this work with dual constraints, which afford bi-directional grouping (e.g. for cabinets on a wall). Such objects are then automatically grouped and can be manipulated as a group. However, dual constraints need to be predefined.

Last, but not least, several researchers investigated tangible user interfaces to 3D scene construction. Closest to our work is the work on geometric construction kits (e.g. Aish 2001). While these construction kits naturally allow for composite object manipulation their main disadvantage is the extremely limited set of primitive objects as most implementations offer only a single size and shape of block. This clearly limits the expressivity of these systems. The other downside of this approach is that certain operations, such as copying a model or rotating a part of the model by 90 degrees, are very time-consuming.

2.1 Motivation & Contributions

The goal of our work is to provide a conceptual 3D design system with a user interface that shares the main properties of Lego. In particular we want to support easy modification of composite objects. As Lego has many desirable properties we chose to emulate Lego insofar as we provide several kinds of basic blocks to construct 3D scenes. However, we aim to go beyond real Lego and support operations that are only possible with a virtual representation, such as easy object duplication and object resizing. Furthermore, our system allows blocks to attach side-by-side, which is also not possible in real Lego. No previous approach fulfils all these criteria, as the systems are either limited by simple group manipulation techniques or by physical constraints.

The technical contributions of our work are:

- *Group Separation*
  We present three new techniques that allow the user to separate (almost) arbitrary parts from a composite object.
- *Movement*
  We present a new technique for the movement of arbitrary objects, which provides predictable and visually smooth results.
- *Group Placement*
  To support the re-attachment of a complex composite part to the existing scene we present an adaptation of previous work to our system.
- *User Study*
  We present the results of a user study, which investigates the new group separation, as well as the group movement and placement techniques.

In the following sections, we discuss each of these contributions in turn.

3. Group Separation

In this section, we present two new techniques that support separation of objects into complex parts: *intelligent separation* and *separation with anchoring*. The second technique optionally allows for two-handed operation and we investigate this separately.

3.1 Intelligent Separation

In intelligent separation, a user clicks with the left mouse button on a block and drags it *away* to separate a part. The initial direction of the mouse drag is used to determine which part of the objects is being manipulated. Whenever the direction of the movement "pushes" a block away, that block is considered to be a part of a new group. Blocks connected along the moving direction are also added to the new group and this allows the formation and separation of arbitrary parts.

To make the separation visually predictable, the group of objects that is going to be separated is highlighted according to the mouse direction. In Figure 1(a), the user starts the selection by clicking on a block, in (b)-(e) the mouse is moved in a small circle to visualize all possible combinations of connected components. Finally, in (f) the user separates the component chosen in (e) by moving the mouse further than a certain threshold.

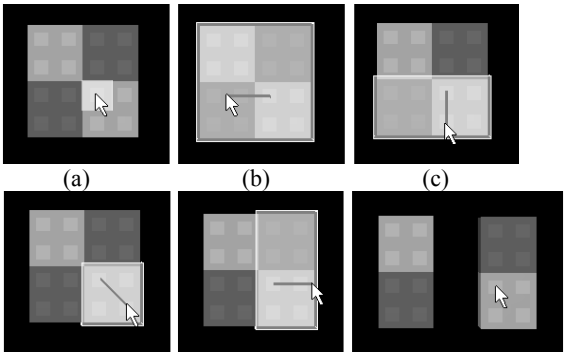


Figure 1: Intelligent separation technique. (a) Start of separation by clicking on a block, (b)-(e) different drag directions form different groups, (f) by dragging further away a group of blocks separates.

To address the problem of manipulating 3D with a 2D input device, the manipulations are view dependant. For this we assume that mouse movements take place on the axis-aligned plane that is most orthogonal to the view direction. Consequently, different movement directions (and even more alternatives for group separation) can be obtained by changing the view direction.

3.2 Separation with Anchoring

In initial tests of the intelligent separation technique, we found that many first-time users did not find this technique easy to learn, as they frequently activated it unintentionally. Consequently, we implemented a new separation technique that makes the process more explicit. Here the user must first specify the part that has to remain in place. This step is called anchoring (see Figure 2a). Then, in a second step, the user selects and drags a part that is to be broken off. If no anchor is placed, all objects connected to the one under the cursor simply move together. While this technique requires an additional manipulation step compared to intelligent separation, it makes the separation process more explicit for the user, which still maintaining the benefit of a directionally dependent method.

In our system the anchoring is activated via the middle mouse button, and the separation is activated with the left button (Figure 2). As this two-step approach can be easily mapped to a two-handed user interface we also implemented another version where the user holds two mice. An anchor is placed with the left button on either device and separation is done with the other mouse (Figure 3).

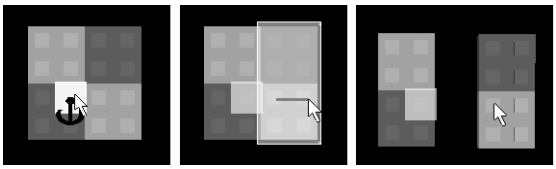


Figure 2: Anchoring with one mouse. (a) A user places an anchor visualized by the white rectangle, (b) the users clicks and drags rightwards, and once a certain threshold is reached a group is separated (c).

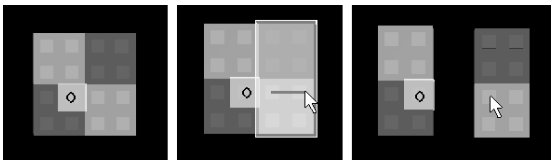


Figure 3: Anchoring with two mice. The second mouse cursor is visualized with a black circle. See text for description.

Here, we first test if there is a plane between the two mouse positions that allows for a simple cut through the model. If it exists, it is used to separate the new part. Otherwise, the dragging direction from the second click is used with the algorithm presented with the intelligent separation technique.

4. Movement

Since we are using 2D input devices to manipulate object in 3D, we have to map 2D inputs to 3D positions. Initially, we used Bukowski's (1995) work, which uses the first visible surface behind the moving object to determine in which plane an object moves. If a collision occurs the colliding surface is used for further movement.

The fundamental problem with this approach is that a composite part can be "grabbed" at many different places with the mouse cursor. Depending on the exact selection position, different results will occur if a composite part is moved across another part of the model. This makes the system unpredictable in situations with detailed object geometry. While this did not occur often in our pilot tests it was problematic enough that we decided to investigate alternative methods.

According to our observations, users seem to consider the entire area of the visual overlap of a foreground object and a complex background. The users seem to expect that the object moves on the foremost surface behind the moving object. Consequently, we select the movement plane from the object surface that is closest to the viewpoint in the region that is *occluded* by the moving object. Figure 4 demonstrates the concept. When the mouse cursor is in position (1), surface A is the surfaces closest to the viewer among all hidden surfaces. Therefore, the object slides on surface A. When moving to position (2), the closest surface is B and consequently, the object slides on the top of it. In our experiments, this results in a visually smooth and predictable object motion. For efficiency, we perform the computation of the foremost occluded surface with the aid of graphics hardware.

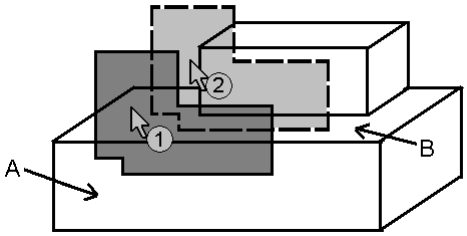


Figure 4: Objects slide on the surface that is both closest to the viewer and occluded by the object.

5. Placement

Once an object is released, it is snapped to the nearest surface. If an object is released above the background, it continues to float in free space, which provides a convenient temporary storage space.

Placing an arbitrary composite object onto an arbitrary surface requires that many possibilities should be considered as any face may potentially snap to any face. To solve this we adapted the algorithm from Kitamura (1998). In his work, several criteria such as angles between faces, movement direction, overlap ratio and face distances are used to reduce the number of candidate faces that can match each other. Scores are calculated for each pair of candidates, and the face pair with the highest score is selected as a constraining pair.

In our implementation, we use only the overlap ratio and the face distance. The overlap ratio computes the relative overlap of two surfaces that face each other. For simplicity, we use the distance along the normal of the current movement plane. We then construct a candidate set of all blocks whose overlap ratio is greater than some threshold. The set is sorted by descending order of overlap ratio and ascending order of distance as secondary criterion. The first entry of this set is chosen as it minimizes object movement and this conforms best to user expectations.

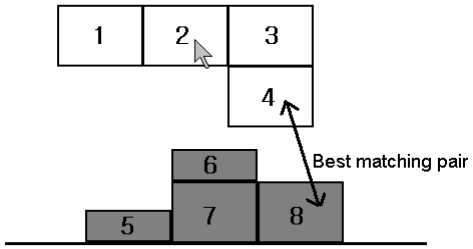


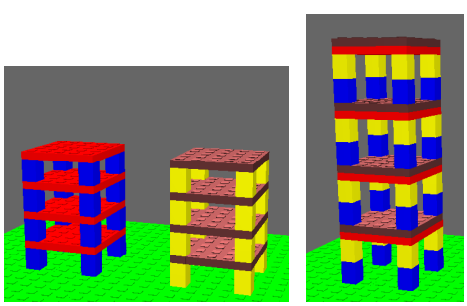
Figure 5: Object placement is determined by finding the pair with minimum distance among the ones with maximum overlap ratio (see text).

Figure 5 illustrates this technique with a side view. The moving part consists of parts 1-4 and the static scene consists of blocks 5-8. Each one of the bottom surfaces of 1, 2, and 4 can attach to each of top surfaces of 5, 6, and 8. The candidate array will store only blocks with maximal overlapping ratio (such as the pairs 1-5, 2-6, and 4-8). Among the candidates, the pair 4-8 will result in minimal movement and consequently it will be selected to snap the two parts together.

6. User Tests

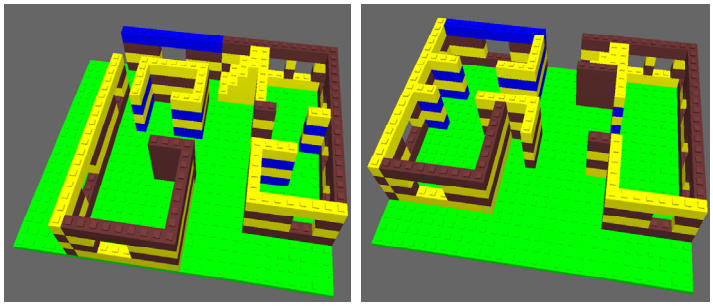
6.1 Test Procedure

Twelve paid participants (6 females, 6 males, age range 18-39, avg. 25.08) out of a pool of graduate and undergraduate university students were recruited. All of the subjects had experience with either 2D authoring tools or 3D games, or both.



(a) Initial scene with two towers (b) Target scene

Figure 7: Merging two 4-layer towers



(a) Initial 3D floor plan (b) Target 3D floor plan

Figure 8: Changing arrangement of 3D floor plan

None of the participants had previous experience with our system. We consider this population to be a reasonable set, given that the many designers use computers on an everyday basis.

Practice and evaluation sessions were conducted in order. In the first part, subjects learned the general operations of the system and practiced simple 3D object movement under the instruction of the experimenter. After this initial session participants were asked to perform two different experimental tasks. The first task was merging two 4-layer towers (Figure 7), and the second a rearrangement of a 3D floor plan (Figure 8). Each participant was asked to perform the each task twice under all three conditions, intelligent separation, separation with anchoring with one mouse, and with two mice. As the two tasks are non-trivial and require some reflection, we considered the first set of three results to be practice, and analyzed only the last three trials for each task. To combat learning effects the order of conditions was counterbalanced across subjects. For all trials, the mouse movement, and actions that the subjects performed were logged along with time.

6.2 Tasks

In the first task, subjects had to merge two 4-layer towers into one (Figure 7). The individual parts of the tower are stacked on top of each other in a repeated pattern. We hypothesized that a first-time user could easily do this task, since there was only a simple relationship between objects (one on top of the other) and the task was repetitive. The minimum number of movements to complete the task was fifteen.

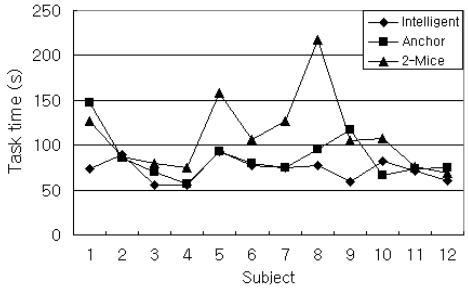
In the second task, participants had to change the arrangement of a floor plan by moving walls and wall assemblies around (Figure 8). The task is close to how a conceptual 3D design system would be used in the real world. This task is relatively difficult because a user must reflect on the connectivity of walls in order to select the correct wall fragments. The minimum number of movements was ten.

6.3 Test Results

For brevity, we refer to intelligent separation as *intelligent*, separation with anchoring with one mouse as *anchor*, and the two mice variant as *2-mice*. The task of merging 4-layer towers is *4-layer*, and the rearrangement of the 3D floor plan is *floor plan*.

6.3.1 Task completion times

In the 4-layer task, 2-mice is significantly slower than intelligent and anchor ($F_{11,2}$=7.3, p<0.01). In the floor plan task, there is no significant difference between the techniques ($F_{11,2}$=1.86, p>0.18). The high variation in completion time of intelligent and 2-mice seems to be the main cause of this. The reasons for this are investigated later in this section. Nevertheless, it should be noted that with a few exceptions, most participants could complete the tasks in a reasonable amount of time.



(a) Task time for merging 4-layer towers

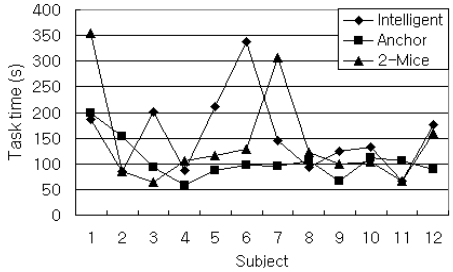


Figure 9: (b) Task time for rearranging 3D floor plan

6.3.2 Decomposition of actions

To gain an insight into how participants spent their time on each task, we decomposed time into the

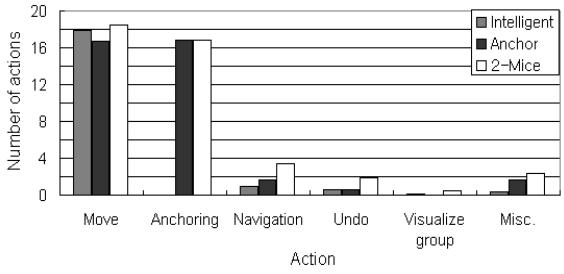
different action categories. The actions are move, anchor, navigate, undo, and *visualize group*. Visualize group means that the user highlights a group via directional dragging, but cancels the operations before actual separation occurs.

One noticeable result for the 2-mice condition is that users assigned practically exclusive roles to their hands. Usually, the left hand is used only for anchoring while the right performs all other actions (see Table 2). To simplify further analysis, we merged the data for left-hand and right-hand actions.

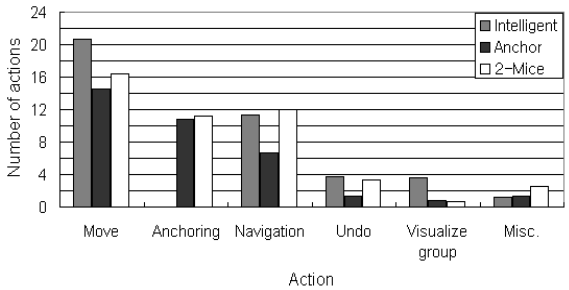
	4-Layer		Floor plan	
	Left	Right	Left	Right
Move	0	18.5	0.17	16.25
Anchor	16.83	0	11.25	0
Navigate	0.17	3.58	0.42	11.58
All other	2.83	3.42	1.58	5

Table 2: Average number of actions by each hand with 2-mice condition.

Most of actions performed are move, anchoring, and navigation (Figure 10). If we compare intelligent and anchor, users made significantly more errors with intelligent and also utilized the visualize group action significantly more often. In the floor plan task, there is a significant correlation of completion time and the number of navigation actions in both the intelligent (0.96) and 2-mice condition (0.73), but not in the anchor condition (0.2).



(a) Average number of actions in 4-layer task



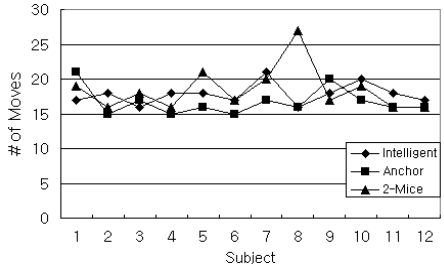
(b) Average number of actions in floor plan task

Figure 10: Average number of actions for each task

6.3.3 Number of move operations

In the 4-layer task, the number of move operations is not different ($F_{11,2} = 1.75$, $p > 0.1$). However, in the floor plan task, intelligent required significantly more operations than anchor ($F_{11,2} = 3.56$, $p<0.05$).

In many instances, participants took longer with intelligent than with anchor. Usually, this is due to an erroneous activation of the intelligent technique, which then results in a sequence of corrective actions. This did not occur with anchor. However, in the floor plan task under the intelligent condition subject #2 and #11 were able to finish the task with an (almost) optimal number of move operations, which shows that some participants were able to utilize this technique fully, as their task times were also minimal (see Figure 9(b)).



(a) Number of move operations in 4-layer task

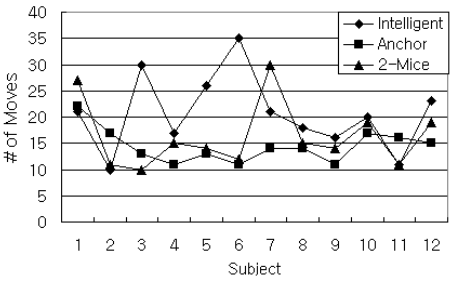


Figure 11: (b) Number of move operations by subjects in floor plan task.

6.3.4 Select time before a movement

To investigate how long it took users to select a group to separate, we decomposed the sequence of actions for a move operation (Table 4). We call the period that is shown in italic in Table 4, *select time*. We hypothesize that the select time would allow us to gain an insight into the complexity of selecting the right object, which is a mixture of cognitive processing as well as motor action.

We computed the average select time by dividing the select time with the number select actions for each subject. For both tasks, the effects of the average select time are strongly significant (4-layer: $F_{11,2}=28.69$, $p<<0.01$; floor plan: $F_{11,2}=28.31$, $p<<0.01$). Also, all three techniques are different for both tasks. For the 4 layer task, the average select time for intelligent is 0.8 s, for anchor 1.21 s, and for 2-mice 1.55 s. In the floor plan task, the average select time for intelligent is 1.52 s, for anchor 1.86 s, for 2-mice 2.55 s.

The select time for intelligent is significantly lower than that of anchor and 2-mice. This is surprising, given that the motor action for these is practically identical (moving the cursor over an object). Furthermore, it seems that participants took less time to reflect on their choice of drag direction for intelligent. The higher select time for 2-mice is most probably due to the fact that not many people are trained to use a mouse with the left hand.

Intelligent	*Move right hand onto block to move* – drag in a direction.
Anchor	*Move right hand onto the anchoring block* – click to anchor – move right hand onto the block to move – drag.
2-mice	*Move left hand onto the anchoring block* – click to anchor – move right hand onto the block to move – drag.

Table 4: Mouse and keyboard commands.

6.3.5 Qualitative results and observations

After finishing all the tasks, participants rated each technique according to a Likert scale (1:worst, 7:excellent). The average preference for intelligent is 5.33, for anchor 5.5, and for 2-mice 4.08. The difference is not significant ($F_{11,2}$=2.92, p >0.05).

Many subjects commented that anchor is more explicit and predictable for the first-time user, but intelligent is more natural to use. There was also a noticeably change of opinion after the tasks. Most of the users showed strong preference for intelligent after finishing the 4-layer task, but after the more challenging floor plan task, many withdrew their preference towards the technique. In the 4-layer task there is only one object relationship, one on top of another. Consequently, the task did not require much attention on how to separate a particular group. However, in the floor plan task, the connectivity relationship between objects is more complex and participants had to experiment with different dragging directions. This is reflected in the larger number of mistakes with this condition.

Some of the participants said explicitly that they liked that objects moved on the closest visible surface. None of the participants commented on the placement, so we can assume that this method works well. However, some participants found it non-intuitive that the viewpoint has to be changed to place an object onto an invisible surface. Some participants used the space in the air as a temporary place when the working plane was too crowded, or when they couldn't immediately find the right place for an object.

6.4 Discussion

One finding of this user study is that even first-time users can successfully complete some conceptual 3D design tasks, such as rearrangement, with the presented techniques. The intelligent separation technique performs well when the task is easy. For more difficult tasks (such as the floor plan scenario) separation with anchoring seem to be a better alternative.

The select time of for intelligent separation is the lowest regardless of the task. This contradicts our expectations, as we assumed that the select time of intelligent to be longer than that of the other techniques, since a user had to reflect on the dragging direction before the action. Participants seemed to use this technique with a trial and error approach. Conversely, with separation with anchoring participants spent more time to reflect on how to separate. The explicit two steps, selecting the object to remain, and then the 'moving' group of objects seem to force the users to think more.

Also, intelligent separation clearly is more susceptible to mistakes. In the floor plan task, only two users finished task without real errors. Some participants needed lengthy corrections to fix a small error. This can happen, as intelligent separation will break off parts with every selection.

For the 2-mice condition all participants used their left hand for anchoring exclusively, and had difficulty in moving the cursor with the left hand. This is clearly not a good alternative for first-time users.

Last, but not least, several participants asked if they could get this system to perform creative work! This is very encouraging, as this is only a prototype that is limited in several aspects.

7. Conclusion & Future Work

In this work, we presented a novel approach for easy modification of a conceptual 3D design. We presented new techniques to select and separate arbitrary object parts, to move them around and to place them. Phrased differently, this work provides some initial solutions to fill some important gaps in the functionality of a conceptual 3D design system.

Furthermore, we presented the results of our user study, which shows that the proposed techniques are useable for first-time users. The anchoring technique seems to be most promising for easy-to-use systems, although the intelligent separation technique may be a better choice for higher-end systems.

For future work we will extend the set of blocks available in the system. The real problem here is not the addition of new geometries, but that we have to

implement a simple, yet efficient way to select from a large palette of object shapes. We are currently investigating this. Also, we intend to incorporate at least a rudimentary sketching interface into the system.

References

Aish, R., Frankel, J., Frazer, J., Patera, A. & Marks, J., (2001), Panel: Computational construction kits for geometric modeling and design, *SI3D'01*, 125-128.

Bukowski, R. & Sequin, C. (1995), Object associations: a simple and practical approach to virtual 3D manipulation, *SI3D'95*, 131-138.

Cross, N. (2000), Engineering design methods: Strategies for product design, *Wiley, 3rd ed.*

Igarashi, T., Matsuoka, S. & Tanaka, H. (1999), Teddy: A sketching interface for 3D freeform design, *SIGGRAPH'99*, 409-416.

Kitamura, Y., Yee, A. & Kishino, F. (1998), A sophisticated manipulation aid in a virtual environment using dynamic constraints among object faces. *PRESENCE,* 7 (5), 460-477.

Neale, H., Cobb, S. & Wilson, J. (2002), A front-ended approach to the user-centered design of VEs, *IEEE VR'02*, 191-198.

Pereira J., Jorge J., Branco V. & Nunes F. (2000), Towards calligraphic interfaces: Sketching 3D scenes with gestures and context icons, *WSCG2000.*

Purcell, A. T. & Gero, J. S. (1998), Drawings and the design process: A review of protocol studies in design and other disciplines and related research in cognitive psychology, *Design Studies*, 19 (4), 389-430.

Stuerzlinger, W. & Smith, G. (2002), Efficient manipulation of object groups in virtual environments, *IEEE VR 2002*, 251-258.

Verstijnen, I., van Leeuwen, C., Goldschmidt, G., Hamel, R. & Hennessey, J. (1998), Sketching and creative discovery, *Design Studies*, 19 (4), 519-546.

de Vries, B. & Achten, H.H. (2002), DDDoolz: Designing with modular masses, *Design Studies*, 23 (6), 515-531

Zeleznik, R.C., Herndon, K. & Hughes, J.F. (1996), SKETCH: An interface for sketching 3D scenes, *SIGGRAPH'96*.

Appendix A: User Interface

The user interface of the Virtual Lego system consists of the main 3D scene view and a menu at the right side. The menu offers a color and object selection palette, an undo button, and a recycle bin (see Figure A1). Users can select a shape from the palette and place it by clicking into the 3D view, or by dragging the shape from the palette to the view.

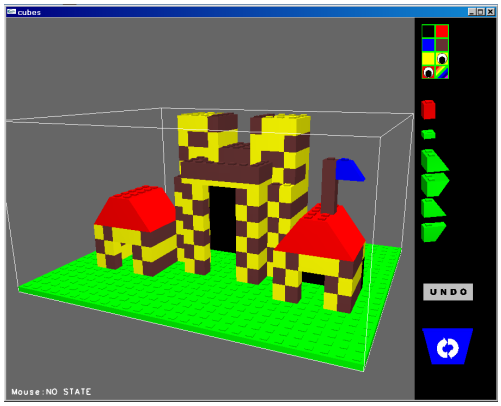


Figure A1: The Virtual Lego interface

Table A1 shows an overview of the commands in the system. Depending on the particular sequence of mouse movements and button presses different actions are taken. A threshold on the movement distance is used to distinguish between clicks and drag operations.

Input command	Function	Realization
Left button click	Add, recolor	On release
Left button drag	Move	Move on drag, Snap on release
Left button drag with shift key	Cloning	Object cloned on shift-press
Any drag started in background	Navigation	During drag
Right button drag	Resize	During drag
Drag and drop to recycle bin	Delete	On release
Middle button click (only in one mouse anchor mode)	Anchor	Realized only if move follows

Table A1: Mouse and keyboard commands.