Title: A System for Desktop Conceptual 3D Design

Authors: Ji-Young Oh, Wolfgang Stuerzlinger

Affiliation:

Department of Computer Science, York University

Toronto, ON, CANADA

Contact Information: Ji-Young Oh.

E-mail: jyoh@cs.yorku.ca

Address:

Dept. of Computer Science, York University

4700 Keele St.

Toronto Ontario, M3J 1P3

CANADA

A System for Desktop Conceptual 3D Design

Abstract: In the traditional design process for a 3D environment, people usually depict a rough prototype to verify their ideas, and iteratively modify its configuration until they are satisfied with the general layout. In this activity, one of the main operations is the rearrangement of single and composite parts of a scene. With current desktop Virtual Reality (VR) systems, selection and manipulation of arbitrary objects in 3D is still difficult. In this work, we present new and efficient techniques that allow even novice users to perform meaningful rearrangement tasks with traditional input devices. The results of our work show that the presented techniques can be mastered quickly and enable users to perform complex tasks on composite objects. Moreover, the system is easy to learn, supports creativity and fun to use.

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

1. Introduction

We present a system that targets the support of 3D scene construction particularly during the conceptual design session, an early stage in the design process. In such a session, a designer explores the problem/solution space to produce an optimum solution that satisfies design constraints. This is normally done by externalizing various solutions (*e.g.* via drawing), evaluating them (*e.g.* visually), and then modifying the solution according to the evaluation. In this stage, many parameters of the model are still undecided and/or frequently changed. Consequently, modifications are frequently drastic and structural and, the visualization of the model is done in a rapid way without much emphasis on precision. In addition, design researchers (Purcell, 1998; Verstijnen,

1998) argue that the main operations in this phase of the process are recognition of the parts of a model by function or structure, and manipulation of these recognized parts.

Therefore, one of the requirements for Computer Aided Design (CAD) systems for the conceptual session is the ability to rapidly build a model without much commitment to detail, and to be able to modify even the overall structure efficiently. Another goal is to provide a scene construction system that supports an iterative (and progressive) modeling process. In this sense, desktop environments are a good solution, as they are the normal work platform for designers, and many other applications are available to aid the users' decision process.

However, mapping from a 2D mouse input to a 3D position is a difficult problem in desktop VR (Issacs, 2002). This poses a challenge in that the system needs to provide for 3D motion of the objects as well as afford object group selection and manipulation. The system presented in this paper, Virtual Lego, was designed to with new techniques to solve these problems.

Our system is based on the idea of Lego™. It uses blocks as primitives that behave similar to real Lego, but there is additional functionality. From a user's standpoint, Lego is very simple and versatile, in that various models can be rapidly built using just simple blocks. From a system developers' standpoint, it is easy to test ideas for interaction techniques by narrowing the application domain, while still keeping relatively close to the context of real situations.

Judging from the way designers work, the (photo-) realistic display of early solution is not desirable as it narrows the solution space too quickly (Lawson, 1996; Schumann, 1996). Consequently, our work focuses on the user interface and not on the realistic display of geometric shapes.

In section 2 of this work, the motivation and our technical contributions are presented. Furthermore, we discuss some characteristics of the conceptual design session as well as the limitations of current CAD systems. In section 3, we review related work. Section 4 introduces our new conceptual design system. Sections 5 to 7 discuss the new techniques in detail, and present a comparison with other existing solutions. In section 8, the results of our user studies are presented. Section 9 discusses the generalization of our new techniques to systems that support arbitrary geometries. Finally, section 10 summarizes our works and provides an outlook to future work.

2. Motivation and Contribution

In general, the goal of any design activity is to produce a description of an artifact that satisfies a set of design problems. The main concern of designers is that design problems are typically *ill-defined* (Cross, 2000; Lawson, 1990), as opposed to well-defined problems (*e.g.* the Towers of Hanoi problem). Ill-defined problems poorly describe the starting state, the constraints that must be fulfilled in the solution, and the form of the final solution. Therefore, they require extensive time to structure the problem itself, and new constraints can arise as preliminary solutions are created. Furthermore, because design constraints are not well defined, solutions require subjective judgment to decide if one is better than the other.

Therefore, designers generally go iteratively through several distinctive stages, to clarify the problem and eventually to come up with a new design. These stages are the analysis of problems, the conceptual design, the embodiment of schemes, and the detailing stage (Cross, 2000). In particular, there are frequent iterations between the analysis of problems and the conceptual design session. In these stages, designers try out different solutions and clarify design constraints via the visualization of solutions.

Sketching is the conventional way to conduct the visual evaluation during the conceptual session. 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 this stage (Purcell, 1998; 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 arbitrarily complex parts of a scene. In addition, the above-cited research showed that current CAD systems do not support these operations very well, and this is the one of bottlenecks for CAD systems to support conceptual design.

There are three main technical contributions in our work. Each one of them addresses a particular sub-problem of the task of manipulating arbitrary parts of a model.

- *Group Separation*: We present three new techniques that allow the user to separate (almost) arbitrary parts from a composite object.
- *Group Movement*: We present a new technique for the movement of arbitrary objects. It provides predictable results and visually smooth motion.
- *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 context.

3. Related Work

In this section we first review previous work on techniques that are targeted at group selection, movement, as well as placement. Then we review several classes of systems that are targeted at the early design process, namely tangible user interfaces and electronic 3D construction kits, sketching systems, as well as Lego CAD systems.

3.1 Group Selection

The selection of arbitrary groups of primitive objects has been largely neglected in the VR/CAD research community, since few people realized this to be a significant problem. However, as it is evident from the research into the design process, group manipulation is very important early on in the 3D design process.

Group selection is fairly easy in 2D drawing applications and is usually implemented by a rectangle selection process or via shift-click selection. However, in 3D, these solutions are not applicable due to the perspective distortion as well as the occlusion between objects. A few systems (Bukowski, 1995; Zeleznik, 1996) support hierarchical grouping or lassoing. A hierarchical relationship is usually established when one object is placed on another, like a cup on a tabletop. Lassoing allows for more flexibility and can select groups of arbitrary objects, but the accuracy of a lassoing gesture, a quick circling motion, is usually very limited. Furthermore, the problem of perspective and occlusion still exists.

Another approach for grouping can be found in DDDoolz (Vries, 2002), an architectural conceptual design tool. In this system, a user fills space with blocks, and these blocks are grouped together by assigning 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.

MIVE (Stuerzlinger, 2002) investigates the use of a set of constraints to rearrange predefined objects efficiently. This system supports a form of bi-directional grouping, named dual constraints, to align objects of the same kind (*e.g.* cabinets on a wall, or chairs in a classroom). Objects satisfying dual constraint snap together, and can be

manipulated as a group. However, dual constraints must be pre-defined for each object and the grouping process still supports only 2D relationships.

3.2 Object Motion and Placement

Providing an intuitive mapping from 2D mouse motion to 3D object motion is a common problem in desktop 3D environments. In a perspective view, there are infinite number of possible positions along each ray defined by the eye and the mouse position in the image plane. The simplest solution is not to attempt to interpret this, but to provide handles/widgets for 3-axis manipulation, as many conventional CAD systems do. While this solution allows no room for the system to fail or to present unexpected results, the task of movement becomes tedious and especially difficult when a user wants to place an object in relation to other objects.

Another way to solve this is to provide a snap-to grid in space or to snap to surfaces of objects. For example, Bier (1990) presented snap-dragging, a technique where the cursor snaps to interesting features in the scene such as intersection of grids, edges or vertex of other objects. This technique works well only when the scene is not cluttered with many objects. As the complexity of the scene increases, it will interrupt the user more frequently with unnecessary snapping and the usability may suffer.

Apart from snapping to any object or grid in a scene, another solution is to snap to certain types of surfaces, usually defined by the real-world behavior of an object. Such systems (e.g. Bukowski, 1995; Smith, 2001) use pre-defined constraints for each object in a scene. There, these constraints specify how an object can move on horizontal or vertical surfaces and to which constraining surfaces it can attach. By doing this, a user can quickly populate a building with furniture, books, etc. The constraints are also used to define a hierarchical scene graph, which affords composite object manipulation. Last, but not least, several systems that use 6 degree-of-freedom (DOF) trackers as input

devices also use constraints to identify surfaces for object movement. Kitamura (1998a) presented a scheme that uses a collision with existing surface to constrain movement of an object in 3D. This allows for natural object movement, except when many other objects are present in a region.

3.3 Scene Construction in Tangible Interfaces

Ishii and Ullmer (1997) were among the first to present the idea of tangible interfaces. The motivation for tangible interfaces is to provide rich affordances and haptic feedback by enabling users to *grasp and manipulate* a physical representation of a virtual object. Most tangible interfaces support only 2D (or at most 2½D) by using a large table with image projected on top (Fjeld, 2002; Ishii, 1997; Piper, 2002). One fundamental limitation of such systems is that it is practically impossible to support full 3D scene construction, as everything is bound to the tabletop. This includes the input devices as well as the focal plane of the projection. Furthermore, these systems lack an undo/redo capability and replay option. Also, it is not clear how tangible objects (*e.g.* "phicons") can attach together to support grouping operations in such systems.

One way to support 3D modeling with tangible user interfaces is to use Lego-like construction kits or clay models (see *e.g.* Aish, 2001; Anderson, 2000; Kitamura, 2001b). A computer acquires the composition of the physical shapes, and then converts them into graphical representations. Typically, blocks are equipped with computational chips and communication ports so that the connectivity between blocks can be detected, and transmitted to the computer. This can also be thought as a specialized input device to enter arbitrary compositions of such blocks. In these systems, users can build scenes with almost no need of training. However, as the tight coupling between physical and graphical representation is important, the set of possible scenes is limited by the available shapes and the number of physical construction units. To work around this

limitation, Anderson (2000) proposed to implement semantic rules based on the composition of the blocks. Their system uses semantic rules to recognize architectural elements such as a door or a roof and *automatically* decorates corresponding parts. This is not a general approach, however, as the semantic rules constrain the type of structures one can build. Furthermore, there is again no undo/redo capability and no replay option.

3.4 Sketching Systems

Several research projects use sketching as the user interface for computer-aided modelling. Most of the systems interpret each sketching stroke as a gesture to create a primitive shape (Eggli, 1997; Qin, 2000; Zeleznik, 1996). For example, in Sketch (Zeleznik, 1996), a user enters a gesture that represents a salient feature of a primitive shape to produce a corresponding shape. Such gesture interfaces improve the efficiency of shape creations. However, the systems constrain users to a limited set of strokes to create a scene, thereby limiting the expressivity of the system. This contravenes a desirable property of the early sketching process, where designers repeatedly and unconsciously doodle before they produce a definite shape. In addition, the gestures require the users to memorize and to recall them for each primitive shape, which can interfere with the designers' thought process during the design sessions.

In contrast to gesture based interfaces, the research group led by Gross (Schweikardt, 1998) proposed to reconstruct a 3D model from sketching only when on the basis of an explicit request by the user. However, their focus is on 2D sketching applications (Gross, 2000), and not directly on a 3D system. In particular, they assumed all the creation activities occur in 2D and the 3D scene must be reconstructed from the 2D drawing. However, there is no complete algorithm that can successfully perform this, especially using only one sketch from one (unknown) viewpoint. To do this, reconstruction algorithms use a set of constraints. However, depending on the

constraints, there is a limit to the variety of shapes that can be handled by such a system (Clowes, 1971; Lipson, 1996).

3.5 Lego CAD Systems

Lego CAD systems (*e.g.* Ldraw, Lego CAD) are usually developed for people who want to build computer Lego models. For this reason, they stick to the rules of Lego blocks, and the user can only build models that are possible in real Lego™. Such systems provide usually virtual replica of many real Lego parts, so that the visualization of the computer model looks exactly like the real one.

4. A New Conceptual Design Tool

The new system presented in this work implements a new solution for 3D conceptual design. At first glance, our system resembles a virtual Lego system (or "Lego CAD" system). However, our system does not strictly follow the rules for Lego blocks. For example, blocks attach side by side, as well as on top of each other, and blocks can be resized without limits. Consequently, the resulting models are sometimes not feasible with real Lego. Given that our goal is the conceptual design session, we explicitly do not want to create a virtual Lego system. However, in order to create an easy-to-use conceptual design tool, we strive to leverage skills that a user has learned from previous experience with real Lego.

The fundamental primitives in our system are basic Lego block shapes, rectangular blocks with two different heights and two different wedge shapes. When an object, be it a single block or a composite, is selected, it follows the mouse cursor and slides over the (horizontal and vertical) surfaces of other blocks. Details of this process are described in section 6 and 7. In many virtual reality systems objects can interpenetrate. This is something that almost all naïve users find hard to understand.

Consequently, our system uses collision detection to prevent objects from interpenetrating each other. The main innovative feature of the system is the operations that affect more than a single primitive block. These are described in sections 5 to 7 in detail.

If the object is moved over the background, it remains floating in the air. This is frequently used as a temporary storage facility during complex operations. In this case, the floating object does not have a surface of movement. We choose to let the object slide on the axis-aligned plane that is most orthogonal to the viewing direction.

4.1 The User Interface

The user interface of the Virtual Lego system consists of the main 3D scene view and a menu on the right side. The menu offers a color and object selection palette, an undo button, and a recycle bin (see Figure 1). A user can place a shape into the 3D view by selecting one object from the palette and then clicking on any surface in the scene, or by dragging the shape from the palette to the view.

Also, the user can select a group of objects in the scene using techniques that are described in the section 5. After selecting a group, he/she can perform several manipulation actions on that selected group, such as resizing, recolor, rotation, and cloning.

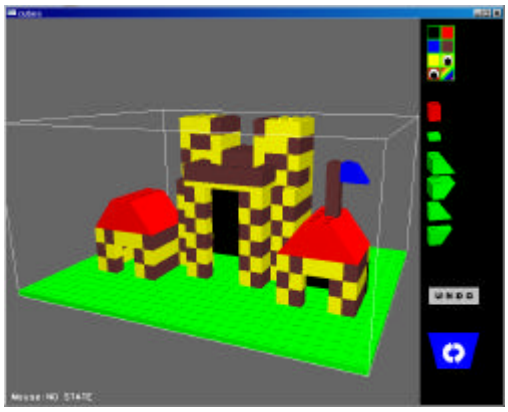


Figure 1. The Virtual Lego interface

Table 1 shows an overview of the commands in the system. Different actions are assigned to the different mouse buttons click and mouse drag actions, which makes (time-consuming) menu selection unnecessary. A threshold on the movement distance is used to distinguish between click and drag operations.

Discrete actions, such as adding a new shape, coloring, and 90° rotation are assigned to button clicks, whereas continuous actions, such as movement and resizing, are assigned to mouse drag actions. The new term “anchor” in Table 1 will be explained in the later section. For now, it is important to mention that the action is discrete and performed by a middle button click. In contrast, a middle button *drag* results in camera rotation. Also, to make this frequently used operation even more easily accessible, any mouse button drag started on the background also results in camera rotation.

In addition, cloning is considered a continuous action, rather than a discrete one. If an object is cloned at a mouse click position, then the cloned objects will overlap with the source objects. This violates the assumption that all objects are solid. Therefore, the system waits with the instantiation of the objects, until the mouse is dragged far enough away from the original position so that the cloned objects do not overlap with the source object. Also, different auditory feedback is provided when an operation succeeds or fails, such as snapping to a surface or deletion to the recycle bin.

Table 1. Mouse and keyboard commands.

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	Realized once no collision occurs with scene objects.
Middle button drag, or Any drag started on 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 operation follows.

5. Group Separation

To afford structural modification of 3D models, efficient group manipulation is important. As an analogy, to draw a large 2D diagram, users usually group shapes to designate them as a component, and then rearrange these components by copying, pasting, resizing, etc. To select a group of objects, users utilize either a rectangular drag gesture, or accumulate objects via shift-click operations. In 2D, this is a simple task, as objects are aligned in the plane. However, in 3D, selecting a rectangular region on the screen is not an appropriate mechanism for object selection due to the perspective distortion and the occlusion between objects.

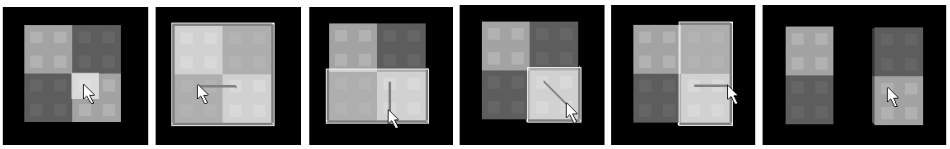
In this section, we present two new techniques for grouping of objects: *intelligent separation* and *separation with anchoring*. Separation with anchoring optionally allows for two-handed operation and this will be described as well.

5.1 Intelligent Grouping

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 object is being manipulated. Whenever the direction of the movement "pushes" another block away, that block is also considered to be a part of the new group. This relatively simple mechanism affords 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 current mouse direction. For simplicity, we show first a 2D example and then a 3D example. In Figure 2(a), the user starts the selection by clicking on a block and keeping the mouse button pressed, 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. Now the user can release the mouse button to place the selected component at a new location. However, to select a part that is interlocked with other parts, a user has to repeatedly decompose the model until he/she obtains the desired part.

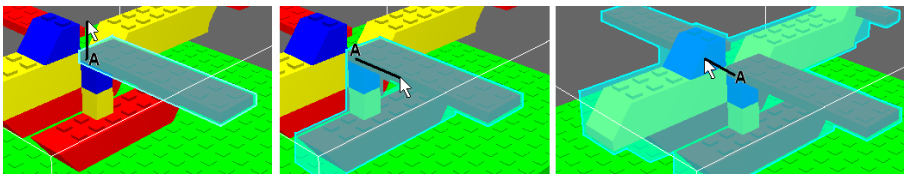


(a) (b) (c) (d) (e) (f)

Figure 2. Intelligent separation technique. (a) Start of separation by clicking on a block, (b)-(e) different drag directions form different groups, (f) dragging further separates a group of blocks.

Please note that, if the user does not move further than a certain threshold, he/she has the option to change the selection. The visual highlighting helps to prevent the user from making mistakes. In addition, if the user stops dragging before separating the highlighted group (*i.e.* before Figure 2f), the group remains selected. This group can then be manipulated with other actions, for example, 90° rotation, resizing, or recoloring.

Figure 3 visualizes the behavior of this technique in a 3D example, where different parts of a plane model are selected. Starting from position A, different connected parts of the plane model are selected according to the drag direction. In Figure 3c, the whole plane is selected, as all the blocks are on the left side of the right wing (the starting position A).

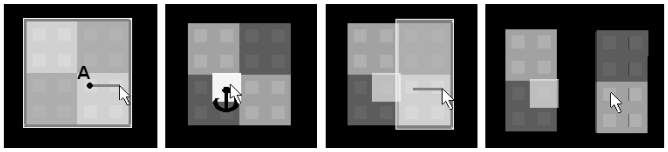


(a) (b) (c)

Figure 3. 3D example for intelligent separation technique. Starting from position A, different drag directions select different parts. See text for detail.

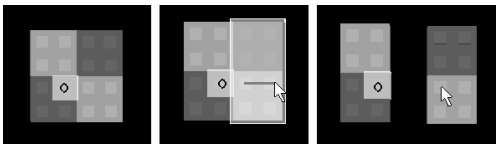
5.2 Separation with Anchoring

In initial evaluations 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 4a). Then, in a second step, the user selects and drags the part that is to be broken off. If no anchor is placed, all objects connected to the selected one simply move together, regardless of dragging direction. In other words, if no anchor is placed, then the whole object moves. While this technique requires an additional manipulation step compared to the intelligent separation technique, it makes the separation process more explicit for the user, but still maintains the benefits of a directionally dependent method.



(a) (b) (c) (d)

Figure 4. Anchoring with one mouse. (a) Dragging into any direction without placing an anchor selects the whole (connected) object. (b) The user places an anchor visualized by the white rectangle, (c) clicks and drags rightwards, and (d) once a certain threshold is reached the group is separated.



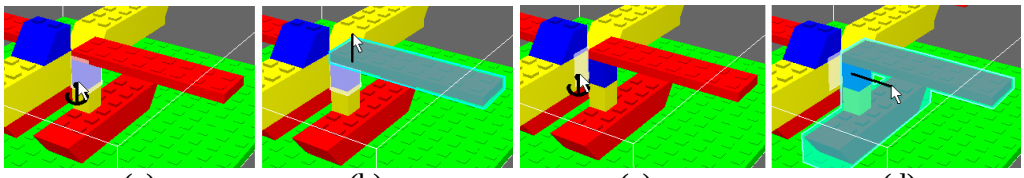
(a) (b) (c)

Figure 5. Anchoring with two mice. The second mouse cursor is visualized with a black circle. See text for description.

In our system, anchoring is activated via the middle mouse button, and the separation is activated with the left button (Figure 4). As this two-step approach can be easily mapped to a two-handed user interface we also implemented a second 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 5).

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

Figure 6 shows a 3D example of this technique. For simplicity, only the one mouse option is visualized, the two mice option works analogous. Again, as with intelligent grouping, group manipulation is possible after highlighting a group (*e.g.* at the step depicted in Figure 6b), and then starting other actions on that group, such as recoloring or resizing.



(a) (b) (c) (d)

Figure 6. 3D example for anchoring with one mouse. To select the wing, a user places an anchor (a), then select the upper part (b). The relative position between the anchor and the selected object position is used for group selection. The same rule applies to step (c) and (d). See text for details.

5.3 2D to 3D Mapping of Selection Gesture

In the techniques described above, we need to identify the drag direction, to map the 2D input to a 3D position. 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 can be obtained by changing the view direction.

However, it is important to point out that this view-dependency affects mostly the results of the intelligent separation technique. In the separation with anchoring technique, the objects to be separated are primarily determined by the spatial relationship between the anchored object and the dragged object. The drag direction is only used when the relationship is ambiguous.

6. Group Movement

Moving objects in 3D with a 2D input device is a typical problem in desktop CAD/VR systems. Clearly, the goal for this is to provide movement of objects that conforms best to the intuition of the user. The SIGGRAPH 2002 course notes on this topic (Issacs, 2002) summarize typical solutions for this problem as follows:

1. Let a user select a movement axis by providing handles for all three axes. In this method, the movement task becomes tedious and sometimes non-intuitive, especially when a user needs to place an object relative to others. For example, IronCAD provides a plane handle and an axis handle for movement. A user has to repeatedly click different handles to select the desired manipulation and then move the object until he/she can reach the target position.
2. Move an object on a plane parallel to the viewing plane. Although this is simple to implement, this technique does not work for users. The resulting movement is not intuitive and frequently misleading.
3. Use obvious structures in the scene to determine the plane of motion. When there is a drawing plane, then move the object parallel to it. When a user positions something on a surface, then snap to that surface. This is generally achieved by defining behavior of an object, for example, whether it moves on a horizontal or vertical surface (*e.g.* Bukowski, 1995; Stuerzlinger, 2002). The definition is generally hard-wired into the internal description of the object, and thus, changing its behavior is difficult for the user.
4. Finally, the system can try to guess the movement direction based on the initial cursor movement. Usually, heuristics are employed for this. Although the heuristics may work correctly most of the time, one wrong guess can frustrate the user and deteriorate the overall usability of the system significantly. Furthermore, it is

important that appropriate thresholds are employed; otherwise involuntary activation is a real problem.

In addition to the issues mentioned above, all these techniques strongly rely on the exact mouse cursor position. For example, in Cosmo3D (discussed in the SIGGRAPH 2002 course note on 3D manipulation; Issacs, 2002), the *follow-the-cursor* technique is used to find and to snap to the closest geometric feature (*e.g.* vertex, edge, or surface) behind the cursor. This approach works reasonably well when small single objects are manipulated. However, when moving bigger composite objects, this method becomes problematic. In our initial evaluations, we observed that users seem to concentrate more on the position of the moving object in relation to the scene, rather than the position of the cursor. When only the cursor position is taken into account, the same kind of mouse motions may then also result in different movements of the object depending on where the cursor position is on the visible surface of the moving object. This is clearly undesirable. Also due to the lack of visual depth information, the movement task thus becomes strenuous since users cannot predict how the mouse movement will be mapped to object movement.

6.1 Movement on a Surface

In our technique, we try to provide visually smooth and predictable object motion, without handles or predefined object behavior. Instead, we use occlusion, which is known to be the most powerful depth cue (Wickins and Hollands, 1999).

In preliminary experiments during our development process, we noticed that users 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 (hidden) surface behind the moving object. Consequently, we select the movement plane

according to the object surface that is closest to the viewpoint in the region that is *occluded* by the moving object. Figure 7 demonstrates the concept. When the mouse cursor is in position (1), surface A is the surface 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 top of the block. For efficiency, we perform the computation of the foremost occluded surface with the aid of graphics hardware.

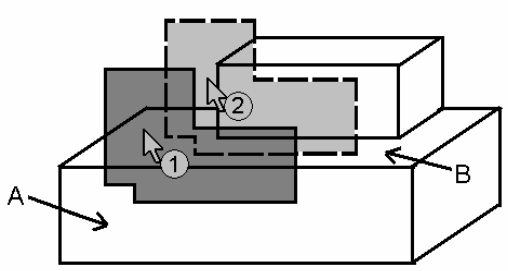


Figure 7. Objects slide on the surface that is both closest to the viewer and occluded by the object.

6.2 Movement in Free Space

In general, objects slide on the foremost surface, except when there is no surface behind the moving object. In this case, the object moves in free space on an axis-aligned plane. The choice of this plane is dictated by viewing direction. That is, the system chooses the axis-aligned plane that is most orthogonal to the view-direction, as this provides the most predictable mapping between 2D input and object motion.

7. Group 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 are considered as any face may potentially snap to any face. To solve this

we adapted an algorithm described first by Kitamura (1998a). In this 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 the constraining pair.

In our implementation, we use only the overlap ratio and the face distance. The reason for this simplifying choice is that we deal basically only with axis-aligned, rectangular blocks and consequently all other criteria are not appropriate. 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. Figure 8 illustrates this concept. According to the measure of face distance block 4 is closest to block 5, 6, 7, and 8. Among these pairs, the faces of block 8 overlap most with the faces of block 4, and thus block 4 and 8 snap together. The first entry of this set is chosen as it minimizes object movement and thus conforms best to user expectations. The snap is accomplished by translating the moving object so that the pair of best matching faces coincides.

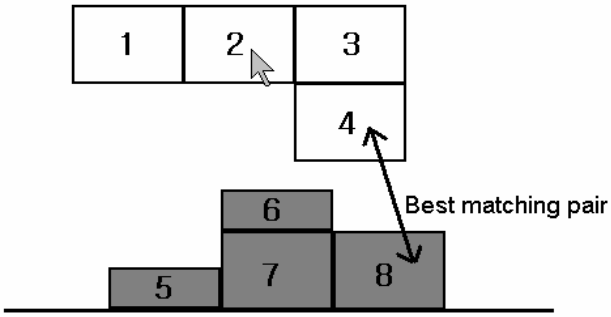


Figure 8. Object placement is determined by finding the pair with minimum distance among the ones with maximum overlap ratio (see text).

8. User Studies

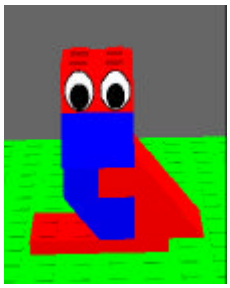
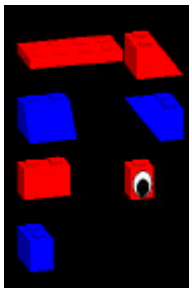
To verify the usability of the system, we conducted a pilot test and a formal test. The pilot test was mainly used to verify the movement algorithm, by asking the users to perform a simple task (moving one block at a time). In this test, we also compared the task completion time with real Lego.

In the formal test, we evaluated mainly the different group manipulation techniques, as these are the most innovative part of the system. We used three different tasks for the user test, which require different 3D manipulations and consequently exercise different parts of the functionality of the new techniques.

8.1 Pilot Test

We performed a pilot test with a preliminary version of the system. Nine subjects (3 females and 6 males, age range 21-30) from the pool of graduate students in our computer science department volunteered. We asked subjects to build a simple duck model composed of ten Lego blocks (Figure 9). In this pilot test, we compared the performance of a 6 DOF Polhemus tracker and a 2D mouse as input device for our system, as well as user performance with real Lego. To provide a fair comparison between our system and real Lego, we disabled the option to add a block in place via a simple mouse click in the Virtual Lego system. Users had to *drag* the shape from the menu and place it in the target position in the working area.

In the 6 DOF tracker condition, collision detection and snapping to the closest surface were used to make the comparison fair. We also disregarded tracker rotations, which made the tracking system effectively a 3 DOF input device. We emphasize that both of these choices were made to increase user performance with the 6 DOF tracker.

(a) Screen capture of the target duck model (b) Menu window for the pilot test.

Figure 9. Screen capture of the system for the pilot test.

The results of the pilot test show that, on average, the construction time with the 2D mouse was approximately 1.5 times slower than the performance with real Lego. However, the performance of the Polhemus tracker was 3.6 times slower compared to real Lego. Apparently, this solution did not work well. We hypothesize that the reasons for this are hand fatigue due to the lack of a resting surface, insufficient depth cues (non-stereo display), and most importantly, no haptic feedback. As we currently do not have access to a 6 DOF haptics system, we did not follow this line of research.

Overall, we were content with the results for the Virtual Lego system, given that it is hard to provide the same visual perception and the same tactile feedback as real Lego does, especially as the Virtual Lego system uses a desktop environment with a 2D mouse. Therefore, we selected more complex tasks to investigate the grouping features of the system. The experiment and the results are presented in the next section.

8.2 Formal Test

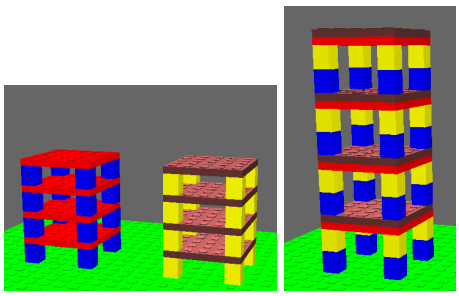
Twelve paid participants (6 females, 6 males, age range 18-39, average 25.08 years) out of a pool of graduate and undergraduate university students were recruited. All of the subjects were right-handed. In addition, all of them had experience with either 2D authoring tools or 3D games, or both. None of the participants had previous experience with our system. We consider this population to be a reasonable choice, given that many designers use computers on an everyday basis.

Test procedure

The test consisted of a practice session, two evaluation sessions, and a subjective evaluation session. First, subjects learned the general operations of the system and practiced simple 3D object movement with instruction provided by the experimenter. Each of the three interaction techniques was practiced until participants felt comfortable with them. The average time period for the whole practice session was 20 minutes.

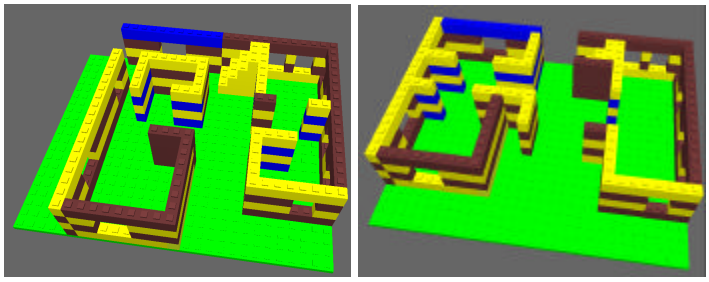
In the first evaluation session participants were asked to perform two different experimental tasks that evaluate the grouping techniques. The first task was to merge two 4-layer towers (Figure 10), and the second a rearrangement of a 3D floor plan (Figure 11). Each participant was asked to perform each task twice under all three conditions, intelligent separation, separation with anchoring with one mouse, and with two mice. In the two mice condition, the left hand mouse has the same functionality as the right one. A user can select an anchoring position with either hand and move other objects with the other hand. Users were informed about this feature. To combat learning effects the order of conditions was counterbalanced across subjects. As the two tasks are non-trivial and require some mental reflection, we considered the first set of three trials as practice, and analyzed only the last three trials for each task.

In the second evaluation session, participants constructed the Lego duck (as in



(a) Initial scene with two towers, and (b) Target scene.

Figure 10. Merging two 4-layer towers



(a) Initial 3D floor plan, and (b) Target 3D floor plan.

Figure 11. Rearrangement of 3D floor plan

Figure 9), with real Lego blocks and the Virtual Lego system. This was done to check the consistency of the results with the previous test.

For all tasks and participants, time-stamped mouse movements and all actions that the user performed were logged into a file. After finishing all the evaluation sessions, participants answered to a questionnaire on the usability of the presented interaction techniques.

Tasks

In the first task, subjects had to merge two 4-layer towers into one (Figure 10). 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 11). 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.

Only actions that are related to rearranging the scene are allowed, such as object separation, movement, and placement as well as navigation. Other actions, such as adding new blocks, rotation, or resizing were disabled during the tests.

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

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.

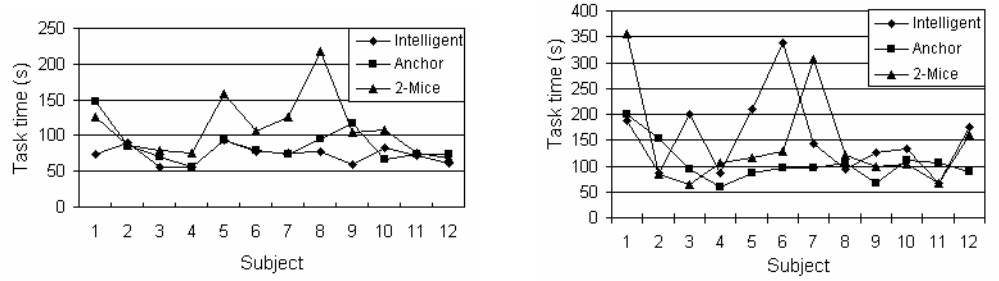


Figure 12. Completion time for (a) 4-layer task, and (b) floor plan task.

Decomposition of actions

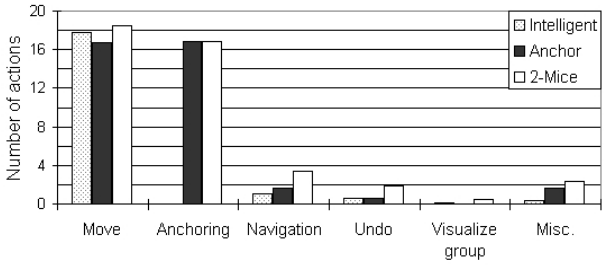
To gain an insight into how participants spent their time on each task, we decomposed time into the different action categories. The actions are move, anchoring, navigate, undo, and *visualize group*. Visualize group signifies that the user highlights a group via directional dragging, but cancels the operations before the actual separation occurs.

One noticeable result for the 2-mice condition is that users practically assigned exclusive roles to each hand. Usually, the left hand is used only for anchoring while the right performs all other actions (see Table 2).

Table 2. Average number of actions by hand in 2-mice condition.

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

To simplify further analysis, we merged the data for left-hand and right-hand actions. Most of actions performed are move, anchoring, and navigation (Figure 13). If we compare intelligent and anchor, it is clear that 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 between the completion time and the number of navigation actions in both the intelligent separation (a correlation coefficient of 0.96) and 2-mice condition (0.73), but not in the anchor condition (0.2). It seems that the users navigated more often in these conditions to check the result of their object movement. We hypothesize that this is because they didn't always understand the result immediately and needed to verify it. While this may be interpreted as that they did not master the interaction techniques completely, it also points out that intelligent separation is harder to understand than anchor technique.



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

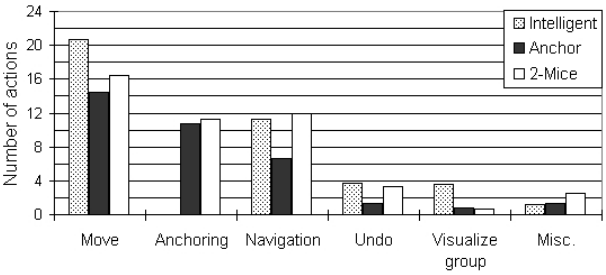


Figure 13. (b) Average number of actions in floor plan task

Number of move operations

In the 4-layer task, the number of move operations is not different ($F_{11,2} = 1.75$, $p > 0.1$) between the conditions. 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 resulted 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 12b).

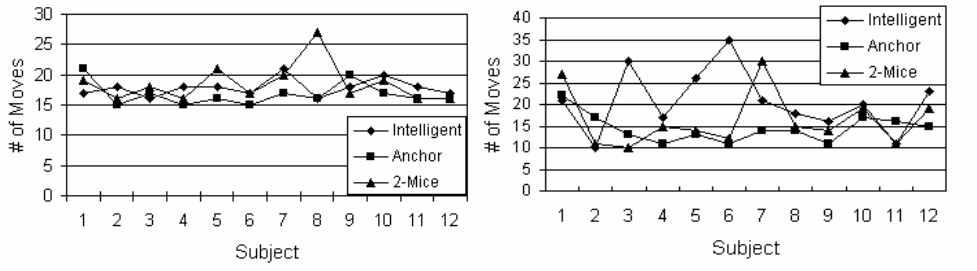


Figure 14. Number of move operations in (a) 4-layer task, and (b) floor plan task.

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 3). In particular, we focus on the time before and up to the first mouse click. We call this period *select time* and show it in italic in Table 3. We hypothesize that this select time would allow us to gain an insight into the overall complexity of selecting the right object for an action, which is a mixture of cognitive processing as well as motor action.

We computed the average select time by identifying the time before a move action from the log files and divided the sum of all the select times with the number of select actions for each subject. For both tasks, the difference of the average select time is strongly significant (4-layer: $F_{11,2}$=28.69, p<<0.01; floor plan: $F_{11,2}$=28.31, p<<0.01). 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 for anchor and 2-mice. This is surprising, given that the motor action for these is practically identical (moving the cursor over an object). 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.

Table 3. Mouse and keyboard commands.

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.

Results of duck construction task

In the formal user test the duck construction task with Virtual Lego was measured to be 2.72 times slower compared to real Lego. We hypothesize this is because all the participants in this test were relatively inexperienced with computers compared to the group in the pilot test (graduate students in computer science department), which resulted in 1.5 times slower performance of virtual Lego than real one. A more detailed analysis of the data revealed that subjects spent most of their time on selecting objects from the menu window, which contains only the seven different shapes required for the duck construction (Figure 9b). This seems to support our hypothesis. If we subtract the time spent selecting objects in the menu window, the overall slowdown factor reduces to 1.5 times compared to real Lego. This coincides with the result of the previous study.

Overall, our analysis of the results for the duck construction task shows that the manipulation of virtual Lego blocks with a 2D input device seems to be approximately

1.5 slower compared to the manipulation of real Lego. There are some fundamental differences between reality and the virtual system that cause this lower performance. For instance, real Lego provides better 3D perception as well as haptic feedback. Also, the relative movement distance was shorter in the real task, as subjects showed a tendency to move their model as close as possible toward the appropriate blocks with real Lego. However, in the virtual system they had to repeatedly move their cursor over the shape menu, which is positioned at a relatively larger distance from the working area.

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 noticeable change of opinion after each task. 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 drag directions. This is reflected in the larger number of mistakes with this condition. Additionally, for the 2-mice option with anchoring, most of the subjects commented that the mouse in the left hand feels unnatural.

Some of the participants said explicitly that they liked that objects moved on the (closest) visible surfaces. None of the participants commented on the placement

technique, 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 a surface that is not visible from the current viewpoint. Some participants also used the free space outside of the base plane (the “air”) as a temporary storage place when the working plane got too crowded or when they couldn’t immediately find the right place for an object.

Discussion

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

The select time for intelligent separation is the lowest regardless of the task. This contradicts our expectations. We expected that the select time for intelligent separation would be longer than that for other techniques, since a user would have to reflect on the drag direction before the action. Participants seemed to use intelligent separation with a *trial and error* approach. Conversely, in the separation with anchoring condition, participants spent more time reflecting on how to separate a group of objects. Note that in this condition users *cannot* separate connected objects if they do not place an anchor. The two explicit steps needed to perform separation with anchoring (first selecting the object to remain and then the actual movement of the “moving” group of objects) seemed to force the users to plan more ahead.

Also, intelligent separation is clearly more susceptible to mistakes. In the floor plan task only two users finished the task without serious errors. Some participants went through lengthy corrections that were triggered by a small mistake in a separation

action. This can happen, as intelligent separation will break off parts for every selection action.

Overall, the results presented above suggest that the intelligent separation requires a longer learning period than separation with anchoring. Although the quantitative results are preferable for separation with anchoring, we do not see evidence to abandon the intelligent separation technique completely. Since this technique does not require the extra (anchoring) step, we hypothesize that the performance and ease-of-use will improve once users master this technique. As indirect evidence we submit that one of authors has used the system extensively and clearly prefers intelligent separation because of the reduced effort. In addition, participants clearly showed a preference for intelligent separation during the 4-layer task. To clarify this issue, a long-term user test needs to be performed.

For the 2-mice condition, all the (right-handed) participants used their other (left) hand exclusively for anchoring. It is important to note that both mice actually had the same functionality, and participants were informed about this during the practice session. However, all users had difficulty with moving the cursor with the left hand, and complained that a mouse in the left hand feels awkward. To a certain degree, this result contradicts most previous studies on bi-manual tasks (Balakrishnan, 1999; Guiard, 1987; Hinkley, 1997). Users frequently looked at the left mouse and its cursor on screen to coordinate between their relative positions. We first speculated that the 2-mice condition should work well, since the technique uses the relative positions of two cursors. However, the 2-mice condition did not provide enough cues for the left hand's physical position in relation to the right hand.

Two possible ways to improve the 2-mice technique can be considered. The first option is to use a device where display and input coincide (*e.g.* tablet display). Users

will see the positions of their hands, and will not have to consciously coordinate between the input device and its cursor. The second option is to map the left hand mouse position relative to the right hand mouse position. As the left mouse moves farther away from the right mouse, the left hand cursor moves away from the other. At first this suggestion seems to contradict common conceptions about bi-manual tasks, namely that the left hand provides a frame of reference for the right hand. However, in our anchor technique, users are always aware of the position of the right hand, and their mental focus seems to be always on the right hand cursor. Therefore, using the right hand as a reference frame seems to be appropriate.

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

9. Discussion for Generalization

In this section, we speculate on generalizing our techniques to more conventional systems based on general polygonal models.

For the separation techniques, the fundamental question is whether the average user will easily understand how objects can be separated for composite objects that are connected at different (non-axis-aligned) angles. In the current system, the user composes a scene with blocks of the same size, which indirectly implies a bi-directional relationship. Therefore, grouping without any concept of hierarchy works naturally. If the scene is composed of different kinds of primitives with different size, such as blocks, wedges, and cylinders, then the relative position becomes an important property. For example, if a small object is attached to the vertical face of a big object, it “belongs” to it. This can be implemented via an appropriate hierarchy. To generalize the

techniques presented here it is then necessary to add hierarchical information according to relative positions. Even in this case, objects at the same level of the hierarchy should be grouped by a bi-directional relationship.

When a scene contains concave objects, we have to consider how a user will visually decompose the scene. Psychologists (Biederman, 1987; Marr, 1982) argue that people recognize the 3D structure of an object by decomposing the object into convex volume primitives. If this is the case, the system must use some form of decomposition of concave objects. However, this is non-trivial and it is not clear how to decompose general concave objects into parts that coincide with the user's expectations.

The group movement technique will generalize to arbitrary scenes, since it only depends on occlusion and depth information. The group placement algorithm is based on Kitamura's (1998a) algorithm, which was originally designed to find a constraining surface between two arbitrary polygonal shapes. As only blocks are represented in our system, only two criteria, face distance and overlap ratio, are used. To snap arbitrary objects at arbitrary angles more criteria must be added, such as the angle between two surfaces.

10. Conclusion and Future Work

In this paper, we presented a conceptual 3D design system that allows even novice users to rapidly build a structurally complex scene. We discussed the problems of manipulating 3D objects in a desktop VR system with a 2D mouse, mentioned the design rationale behind our system, and presented the user interface. We also described details of the new techniques for selection, manipulation, and placement of arbitrary groups of 3D objects. To verify the effectiveness of the proposed system, a series of

user tests were performed. The results showed that even novice users could rapidly perform meaningful tasks in 3D using our proposed techniques.

In the future, more user tests will be conducted once we improve the solutions for conditions that turned out to be inferior. For example, the 6 DOF tracker was very difficult to use due to hand fatigue, a lack of depth cues, and tactile feedback. We plan to use a 6 DOF haptics device to address this issue. The group separation task with 2-mice was difficult for most users. As previously discussed, this might be because our interface design did not provide enough awareness of the position of left hand. We plan to investigate different techniques that could improve the two-handed performance. Furthermore, we wish to extend our system to support at least rudimentary sketching capabilities. Finally, we also plan to investigate grouping features with arbitrary geometric shapes.

References

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

Anderson, D *et al.* (2000). Tangible Interactions and Graphical Interpretation: A New Approach to 3D Modeling. ACM SIGGRAPH: 393-402.

Balakrishnan, R., Hinckley, K. (1999). The role of kinesthetic reference frames in two-handed input performance. UIST: 171-178.

Biederman, I. (1987). Recognition-by-components: A theory of human image understanding, Psychological Review. 94(2):115-147.

Bier, E. (1990). Snap-dragging in three dimensions. ACM Computer Graphics, 24(2): 193-204.

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

Clowes, M. B. (1971). On seeing things, Artificial Intelligence, vol. 2, pp. 79-116.

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

Eggli, L., Hsu, C., Bruderlin, B. and Elber, G. (1997). Inferring 3D models from freehand sketches and constraints. Computer-Aided Design, 29(2): 101-112.

Fjeld, M. *et al.* (2002). Physical and Virtual Tools: Activity Theory Applied to the Design of Groupware. A Special Issue of CSCW: Activity Theory and the Practice of Design, 11(1-2): 153-180.

Gross, M.D. and Do, E.Y- L. (2000). Drawing on the Back of Envelope: A framework for interacting with application programs by freehand drawing. Computers & Graphics 24: 835-849.

Guiard, Y. (1987). Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. Journal of Motor Behavior 19: 486- 517.

Hinckley, K., Pausch, R. and Proffitt, D. (1997). Attention and visual feedback: the bimanual frame of reference. SI3D: 121-126.

LDraw, http://www.ldraw.org

LegoCAD, http://www.lego.com/dacta/legocad

IronCAD, http://www.ironcad.com

Ishii, H. and Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. ACM CHI: 234-241.

Issacs, P., Shrag, J. and Strauss, P.S. (2002). The design and implementation of direct manipulation in 3D. SIGGRAPH 2002 Course Notes.

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

Kitamura, Y., Itoh, Y. and Kishino, F. (2001b). Real-time 3D interaction with ActiveCube. ACM CHI, Extended Abstract: 355-556.

Lawson, B. (1990). How designers think. London: Butterworth Architecture, *2nd Ed.*

Lipson, H. and Shpitalni, M. (1996). Optimization Based Reconstruction of a 3D Object From a Single Freehand Line Drawing. Journal of Computer Aided Design, 28(8): 651-663.

Marr, D. (1982). Chapter 5. Representing shapes for recognition, in Vision (Freeman).

Piper, B., Ratti, C. and Ishii, H. (2002). Illuminating Clay: A 3-D tangible interface for landscape analysis. ACM CHI.

Purcell, A.T. and 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.

Qin, S.F., Wright, D.K. and Jordanov, I.N. (2000). From on-line sketching to 2D and 3D geometry: a system based on fuzzy knowledge. Computer-Aided Design, 32(14): 851-866.

Schumann, J., Strothotte, T., Laser, S. and Raab, A. (1996). Assessing the effect of non-photorealistic rendered images in CAD. CHI'96: 35-41.

Schweikardt, E. and Gross, M. (1998). Digital Clay: Deriving digital models from freehand sketches. Proceedings of ACADIA: 202-211.

Smith, G. and Stuerzlinger, W. (2001) Integration of Constraints into a VR Environment, VRIC 2001: 103-110.

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

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

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

Wickins, C.D. and Hollands, J.G. (1999). Chapter 4. Spatial displays, in Engineering psychology and human performance (Prentice-Hall, *3rd Ed.*)

Zeleznik, R.C., Herndon, K. and Hughes, J.F. (1996). SKETCH: An interface for sketching 3D scenes. SIGGRAPH: 163-170.