Group Selection Techniques for Efficient 3D Modeling

Ji-Young Oh*, Wolfgang Stuerzlinger**, Darius Dadgari**

* Optical Sciences, University of Arizona
Tucson, AZ, USA

** Computer Science, York University
Toronto, ON, Canada

ABSTRACT

Object selection and manipulation (e.g. moving, rotating) are the most basic tasks in 3D scene construction. While most research on selection and manipulation techniques targets single objects, we examine the concept of group selection in this paper.

Group selection is often given lesser importance than single object selection, yet is vital in providing users with a way to modify larger scenes with objects which are repetitive, sequential, or otherwise inherently understood as 'belonging together' by a user. We observed users manipulating objects in 3D scenes, and while doing so, they clearly expected that objects would be grouped based on their gravitational relationship. That is, all objects that are supported by some selected object will follow the motion of the selected object when manipulated.

In this paper, we present a system that efficiently supports the manipulation of groups of objects via a gravitational hierarchy. As this hierarchy is derived with a collision detector, the new grouping techniques do not require semantic or user specified information to work. The results of the evaluation show that using the gravitational hierarchy improves scene rearrangement significantly compared to conventional non-hierarchical methods. Finally, we discuss lessons learned from this study and make some suggestions on how the results can be incorporated into other systems.

CR Categories and Subject Descriptors: I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques, I.3.5 [Computational Geometry and Object Modeling]: Object hierarchies, J.6.0 [Computer-aided design (CAD)]
Additional Keywords: Group selection and manipulation

1 INTRODUCTION

Selection and manipulation of objects in 3D modeling systems are typically used to modify a virtual representation of real world objects such as furniture, buildings, or architectural elements – walls or pillars, etc. These types of scenes generally include some sort of ground surface to provide users with a frame of reference and a depth cue [1, 21]. This ground surface is generally used as a base for all construction.

In real world scenery, we can frequently observe scenes composed of repetitive or similar objects – e.g. rows of desks and chairs in an office, windows on a building, or houses in a suburban town. Even if some components in the scene do not look exactly the same, people tend to ignore those differences and group these objects into conceptual units to understand the overall structure, e.g. the way we can conceptually group a row of assorted books on a shelf.

3D scene modeling systems often replicate such real world situations where an element or a number of neighbouring elements are repeatedly used in different arrangements. Hence, selection and rearrangement of groups are important tasks in 3D scene modeling. In particular, we believe that group selection and manipulation are key ingredients for systems that can deal with larger scale problems, i.e. scenes consisting of many objects.

The focus of this paper is efficient and natural 3D group selection methods to help users explore various arrangements of a larger, complex scene. In this context, we are mainly interested in two very common types of grouping**:** *bi-directional* and *hierarchical groupings.* A bi-directional relationship is perceived when objects are arranged in a row – e.g. tables in a lecture hall or trees along a street. On the other hand, a hierarchical relationship is perceived when an object is attached to another object by gravity or a fixture – e.g. a cup on a tabletop or picture on a wall.

For bi-directional grouping, users generally use the conventional method of rectangular selection to select a group of objects in current 3D modeling systems. One disadvantage of this technique is it may also select objects that are hidden from the current viewpoint (i.e. objects behind other objects), and thus users have to frequently check the selected set of objects whenever they perform a grouping. In our informal observations of users naïve to 3D modeling, they seemed to be often surprised by the selection results after discovering that some unexpected objects were grouped. Phrased differently, we believe that the familiar paradigm of 2D rectangle selection transfers only imperfectly into the 3D domain, as perspective and occlusion complicate the issue.

We also would like to point out that the standard user interface mechanism of (shift-) clicking on each object in turn to define a group is hardly optimal. While it is one of the most general grouping techniques and can be used for arbitrary selections, we argue that it is suboptimal for many common scene structures with a hierarchical scene structure.

For hierarchical grouping the placement of an object usually determines what type of behaviour it can have, e.g. the type of surface that the object can be placed on or the constraint plane that the object can move on. Such type of behaviour can be explicitly supported via semantic definitions or explicit user specification. The disadvantages of using semantic information are: 1) it restricts the domain of scene construction to the range of provided semantic information, e.g. kitchen/office interior design and 2) it is difficult to deviate from the semantic constraints on the fly, e.g. a plate designed to go on tables cannot be placed on a wall. On the other hand, explicit user specification requires some level of user expertise, and thus this idea is only useful for domain experts such as machine part designers.

This paper introduces a new hierarchical grouping scheme, and its application to the manipulation of bi-directional groups. The presented manipulation techniques provide efficient group selection for general types of objects in a 3D modeling system. The new hierarchical grouping scheme makes use of contact information provided by a collision detector, and thus does not

require semantic information. This allows users to manipulate arbitrary scenes without requiring semantic information. Furthermore, users can play and experiment with many different compositions, since the system does not restrict object placement to particular surface types.

2 Previous Work

Many commercial 3D modeling systems are very complex applications and require substantial user training before they can be used effectively. Most of these systems include only some variant of rectangle selection and/or shift-clicking to support group selection. On the research side, many different solutions have been proposed to simplify the modeling process [5, 6, 9, 10, 16, 22]. Most of these systems are targeted at the creation of a single or a small number of objects. Typically, the modification of the overall structure of a scene through efficient group selection and manipulation is neglected in these prototypes. For example, in Chateau [10], users have to delete an edge and redraw it (potentially repeatedly) to modify the created scene, rather than selecting an edge and manipulating it directly to create the desired result. Therefore, these systems are limited when it comes to the creation and iterative manipulation of a large, complex structure. It is also noteworthy that a recent publication on Virtual-SAP [3], an architectural building design and simulation system for Immersive Virtual Environments, explicitly recognizes the selection and manipulation of object groups as an important problem in their "future work" section.

In the following, we discuss only 3D modeling systems that address the issue of group selection and/or manipulation, as this is a key issue for modeling of larger scenes.

Sketch [22] supports lassoing for selection of groups of arbitrary objects. However, the accuracy of a lassoing gesture, a quick circling motion, is somewhat limited and may lead to selection errors. Furthermore, the problem of perspective and occlusion still exists, similar to the problems with rectangle selection.

Another approach for grouping can be found in DDDoolz [20], 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 colour. The authors motivate their choice with the fact that different colours are used to visualize different architectural elements. However, in other application domains, where colour can be an important aesthetic feature, this may not be appropriate. Hence, this mechanism seems to be unnecessarily restrictive and time-consuming to use as each block needs to be coloured individually.

Several researchers also attempted to provide rapid assembly and rearrangement of a 3D scene using a library of predefined objects and accompanying knowledge-base, e.g. for machine assembly [11], or building interior design [4, 15, 18, 19]. Real-world behaviours such as gravity or collision are usually simulated to facilitate the task. One of the first was the Object Association approach [4]. In this system, predefined information on the behaviour of each object, such as vertical or horizontal movement constraints, is used to restrict how objects can be moved.

Mive [15, 18, 19] investigates the use of a several forms of semantic constraints to rearrange predefined objects efficiently. Hierarchical grouping was facilitated directly by the potential placement of objects to their usual locations (e.g. phone on table) [15, 18] and these constraints also improved user interaction greatly. The same kind of ideas is also available in the Smartscene technology in Multigen-Paradigms products [17]. The Mive system also introduced a new 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) [19]. Objects satisfying these dual constraints snap together, and can be manipulated as a group. However, all constraints in the Mive system must be pre-defined and the grouping process supports only 2D relationships. Furthermore, the dual constraint manipulation technique applies only to groups of *identical* objects.

In the Virtual Lego system [13], bi-directional grouping techniques were investigated in a simple block world. Here, the ideas of dual constraints from the Mive systems are extended towards full 3D bi-directional grouping techniques. Using the two grouping techniques *directional dragging* and *anchoring and grouping*, users were able to manipulate complex compositions in a block world efficiently. The main limitation of the grouping techniques in the Virtual Lego system is that they are only applicable to scenes with rectangular block shapes. This leaves the question if the presented techniques will work for general types of objects open.

The grouping techniques newly presented in this paper build directly upon the work in Virtual Lego. Hence, the next subsection will revisit the relevant parts of the Virtual Lego system in detail.

2.1 Bi-directional Grouping in a Block World

The Virtual Lego system [13] was built based on the observation that people can build large and complex scene by manipulating various groups of relatively simple Lego blocks. Figure 1 is an image of the user interface of the Virtual Lego system.

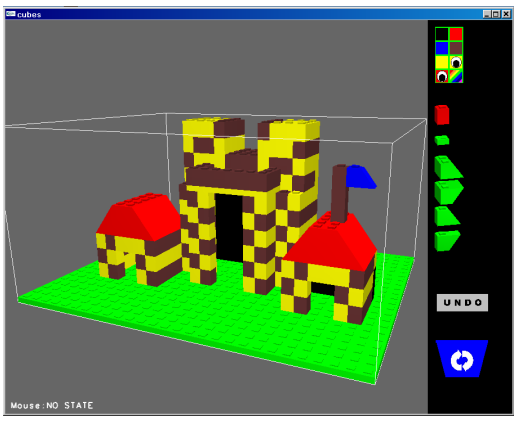


Figure 1. Screen capture of the Virtual Lego system.

In Virtual Lego, two grouping techniques were introduced: *directional dragging* and *anchoring and grouping*. Both techniques allow users to select a subset of connected components.

In *directional dragging*, the user drags an object, as if he/she is tearing one part from the other. If the dragging direction is towards a (connected) object, then it gets selected. Separation of selected subsets occurs only if the movement amplitude exceeds a threshold. Figure 2 demonstrates this scheme with a simple 2x2 arrangement of Lego blocks. For example, in Figure 2b, dragging downwards and to the right, selects only the bottom right block. Conversely, in Figure 2c, a drag downwards selects both blocks on the bottom as they are connected. In Figure 2d, the user is moving towards all blocks and hence, all 4 blocks are selected. Finally, in Figure 2e and f, the user moves further and further to the right, which eventually separates the two rightmost blocks.

Another way to describe this technique is that we assume that objects are "sticky" in that if an object is in contact with the currently selected object and is not in the direction opposite to the

dragging direction, then that object also becomes selected. Overall, this technique allows the selection of many different parts of large clusters of block objects.

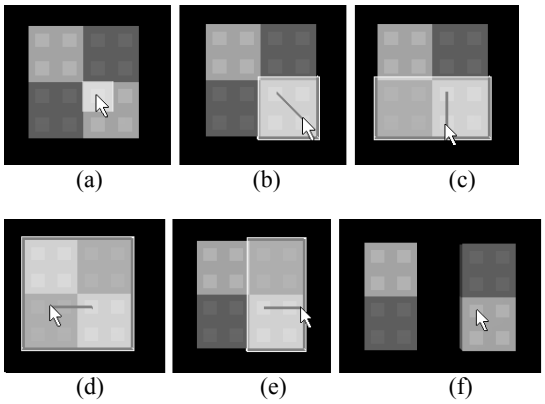


Figure 2. Directional dragging visualized by a line from the starting point to the current cursor position. (a) Start of separation by clicking on a block, (b)-(e) different drag directions form different groups, (f) dragging further separates a group of blocks.

Anchoring and grouping (Figure 3) is a variant of directional dragging. As explained in [13], in user studies on directional dragging, users sometimes inadvertently broke existing compositions as any drag may break a group. Hence, the *anchoring and grouping* technique introduces an explicit additional step to define a group: *anchoring* and *dragging*. First, the user places one (or more) anchors onto object(s) that should *not* be in the group (Figure 3.b). Then, the user can click and drag another adjacent object to select a group (Figure 3.c). The positional relationship between the anchored object and the dragged object is considered to be a dragging direction, which is used to define the selected group of objects, just like in the directional dragging scheme. If the user does not place an anchor, all connected objects are selected as a group. With this grouping method, users are less likely to "break" a composition by mistake.

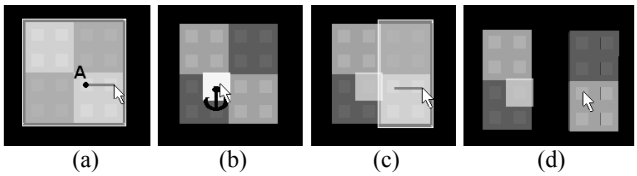


Figure 3. Anchoring and grouping. (a) Dragging into any direction without placing an anchor selects the whole (connected) object. (b) The user places an anchor visualized by the white rectangle, (c) clicks on another block and drags rightwards, and (d) once a certain threshold is reached the group is separated.

These techniques provide a natural way of selecting "an object" as perceived by a user, since people generally recognize a connected group of elements as an object in Virtual Lego. For example, people may perceive the scene in Figure 1 as a "castle" that is composed of a "main gate" and two "guard houses". However, for the system, the scene is nothing more than a collection of connected Lego blocks. Using the presented grouping methods, users can select what they consider to be a meaningful part of the scene and manipulate it as a unit.

While these two techniques work well in a world consisting of many small blocks, they implicitly assume that all objects are of the same type and the same size.

3 Group Selection Techniques for General Geometry

In the user evaluation of Virtual Lego [13], the test focused on bi-directional grouping techniques and showed that the techniques work well when objects have approximately the same size and regular shapes. However, when there are objects with various sizes and different shapes, for example, blocks, wedges, and cylinders, then the relative position becomes an important property. No previous work addresses this issue.

3.1 A General Group Selection Techniques

As discussed in the introduction, we believe, based on our observation of users manipulating geometry, that the hierarchical relationship resulting from gravity affects the grouping of objects. That is, users automatically expect that objects supported by an object via gravity will follow the motion of the supporting object. Note here that objects at the same level of hierarchy may also be grouped via a bi-directional (i.e. side-by-side) relationship. As an analogy, computational vision researchers [2, 12] hypothesized that humans recognize an object by decomposing it into many smaller simplified components. They also suggested hierarchical decomposition of objects as a scheme to recognize a category of an object (e.g. human, dog, etc.). Note that side-by-side relationships describe both objects that are in physical contact and those that are not.

This section explains our grouping techniques using a simple office scenario. However, we immediately point out that the methods discussed below do not use any semantic knowledge about the scene, such as whether the computer monitor in Figure 4 should always be constrained to the top surface of the desk. Therefore, the user can place objects in any configuration, for example, two computer monitors on top of each other or a keyboard on a monitor. We expect that this freedom will aid the user overall by facilitating experimentation with various configurations of a scene.

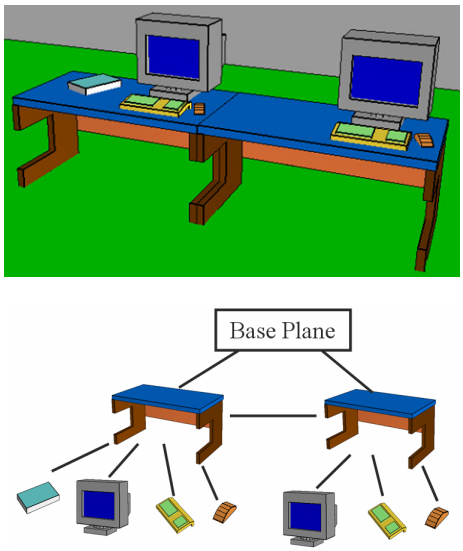


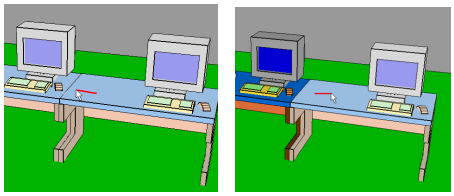
Figure 4. An office scene and it's contact graph

To support the idea of a gravitational hierarchy, we use a collision detector to build a contact graph of a scene. The contact graph is built starting from the base plane as a root node, providing the frame of reference for the gravitational hierarchy. As illustrated in Figure 4, the child-parent relationship in the graph represents attachment relative to the base plane. The sibling

relationship represents a side-by-side attachment between objects that share the same contact object as a parent. As a result, the contact graph represents both hierarchical and bi-directional relationships at the same time.

As in Virtual Lego, we interpret the location of the mouse cursor as the reference position, and then allow the user to continue with a dragging motion to select a group of objects. Dragging within a small, static distance triggers highlighting of the currently selected group. The selection is computed using the contact graph and the direction of the drag from the user's input. Once the user is satisfied with the current selection, he/she can drag the mouse further from the initial reference position to separate the selected group and move it to another place. For the 3D movement, we use a previously presented technique where the object follows the mouse cursor position naturally, while the object stays constrained to the surfaces of the rest of the scene and avoids collisions [14].

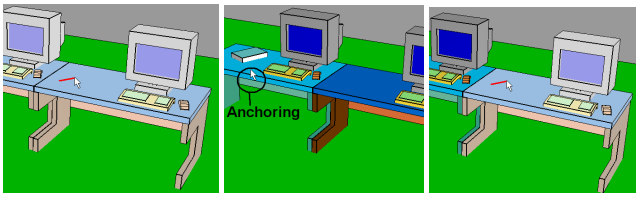
The difference between directional dragging, and anchoring and grouping is in how the binding of the bi-directional relationship in the contact graph can be disconnected. In directional dragging (Figure 5), dragging away from the side-by-side contact relationship disconnects the binding. The child objects that belong to the dragged objects are also selected into the group due to their hierarchical relationship.



(a) (b)

Figure 5. Directional dragging. (a) Objects in the dragging direction are selected and all supported objects are selected together. (b) Dragging away de-select the objects that are not in the dragging direction.

In anchoring and grouping, dragging in any direction will select all the connected components when there is no anchored object (Figure 6.a). To disconnect the side-by-side binding, the user places an anchor onto objects that will not belong to the group, and then selects the desired objects (Figure 6.b, c).



(a) (b) (c)

Figure 6. Anchoring and grouping. (a) Dragging in any direction selects all the connected objects in the contact graph. (b) By placing an anchor and (c) dragging on other part disconnect the contact siblings.

When two objects support an object, that object has two parents in the contact graph (Figure 7). In fact, an object can have an arbitrary number of parents. Due to the gravitational hierarchy, when any parent is selected, all child objects are also selected in our system. That means that the monitor in the middle will be selected when either of the two desks in Figure 7 is selected.

The user can also explicitly specify when he/she does not want the object to be in the selected group by using either of the presented grouping methods. For example, in directional dragging, the user can drag objects that are underneath some object that they do not wish to select by dragging "downward", which only selects the part underneath said object as part of the group. In anchoring, the user can simply anchor the object that they do not want to be in the group.

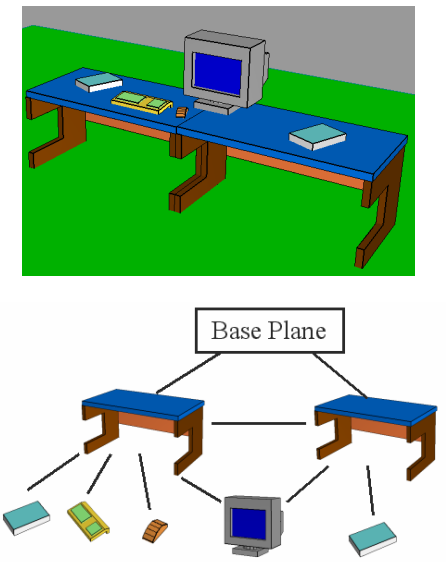


Figure 7. An object that has multiple parents

Finally, in both the directional dragging and anchoring and grouping technique, the user can release the button before separation occurs, i.e. after the drag direction has been used to select the object group but before the selected group is moving away. In this case, the highlighted group remains selected and the user can now use other operations (such as rotation) to manipulate the object group.

3.2 Hierarchical Rectangular Selection

In conventional rectangular selection, all the objects that are inside a user specified rectangular region on the screen are selected as a group. The selected group also includes any object that is not visible from the current viewpoint. As mentioned in the introduction, the main disadvantage of this technique is that the user always has to check if he/she selected the desired objects by examining it from different viewpoints. In addition, it is difficult to select small objects that are attached to something else, since the grouping action may result in other (larger) objects to be selected, too.

Hierarchical rectangular selection modifies the behaviour of conventional rectangular selection by applying the information from the contact graph. In this technique, if any object is determined to be inside the user specified rectangular region, then its children objects in the contact graph are also added to the group. For example, if the user specifies a rectangular region around the right desk in Figure 7, then the monitor and the book are also added to the group. The user has an option to de-select any object that belongs to the group later by clicking on it, as in conventional rectangular selection. In scenes with hierarchical configurations, this technique has the advantage that the user does not have to check if "naturally" attached objects are also selected.

3.3 Implementation

This section explains the implementation of the contact graph and the two bi-directional grouping techniques: directional dragging, and anchoring and dragging.

The requirements for the contact graph data structure should allow for easy retrieval of parents of a node, easy identification of sibling nodes in contact, and easy comparisons of hierarchical levels. These requirements make it unnecessary to build a conventional graph data structure that has the typical "next" or "children" nodes to support navigation of the graph. Therefore, in our implementation, the contact graph exists only conceptually.

In order to implement this, we build a list of nodes, each containing: 1) the object, 2) hierarchical level number (with the base plane defined to be at level zero), and 3) a list of parent objects. Siblings that are in contact are easy to determine with the following criterion: if the hierarchy level numbers of two objects are the same and if they are in contact, then they are siblings in contact with each other.

Figure 8 shows the pseudo-code for building our contact graph data structure. First, the system determines the list of components connected to the selected object, resulting in a contact list. A collision detector [8] is used for this process. The collision detector only produces binary feedback whether the two objects are in collision or not, and it only identifies objects overlapping in space, not the contact to each other. Therefore, we feed the collision detector with the mesh objects scaled up by a small factor (in our system, scaled up by 1.02). To identify the contact surfaces, for any two colliding objects, we look for faces of both objects that overlap in space and are facing towards each other. These procedures could be eliminated if a more advanced collision detector that returns contact surfaces of objects within certain threshold distance, e.g. [7] would be used, however at a larger cost of performance compared to the collision detector we are currently using.

Second, the system generates the hierarchy information (*hierarchyList* in Figure 8). This is done by navigating the contact list starting from the base plane, and by assigning the hierarchy level of the object based on the contact relationship. That is, the objects that are in contact with the base plane are assigned as objects in the second level of the contact graph. The third level objects are those that are in the contact with any object in the second level, etc.

As discussed before, the difference between the directional dragging and the anchoring and dragging techniques is the way siblings in contact are disconnected. In terms of implementation, we identify a *cutting plane* that clearly cuts between siblings in the contact graph that are in contact. The difference between the two grouping schemes here is how the cutting planes are identified.

In directional dragging, the system projects the normal direction of each face of the selected object into screen space. The face whose normal is most opposite to the dragging direction is selected as the position of the cutting plane, but the normal vector of that face is flipped to define the orientation of the cutting plane. To provide users with the option to pull away an object that is in contact with multiple objects, we allow diagonal directional dragging. In this situation, there can be a multiple contact planes, whose normal directions are approximately equal in angle to the dragging direction. We handle this by defining multiple cutting planes internally.

In anchoring and grouping, the cutting plane is defined by the contact surface between the anchored object and the dragged object. Since there can be multiple anchored objects, there can be multiple cutting planes.

```
hierarchyList: The contact graph that we need for the selection
               of a group
hierarchyNode: A node of hierarchyList
               struct hierarchyNode { theObject,
                                      currentLevelNumber,
                                      parentList }
contactList: List of connected objects
parentList: List of objects in one level higher than the current
            navigation level
siblingList: List of objects that are in the current level
parentToNodeList: List of objects that are in contact with the
            current object from the parentList

currentLevel ← 0
hierarchyList.add( hierarchyNode ( baseplane,
                                   currentLevel,
                                   null) )
contactList ← all objects connected to the currently selected
              object
parentList.add (baseplane)
siblingList ← [empty]
parentToNodeList ← [empty]

While parentList is not empty
    currentLevel ← currentLevel + 1
    For each node n in contactList
        If n.visit is false
            parentToNodeList ← contact objects in the parentList
            If parentToNodeList is not empty
                hierarchyList.add ( hierarchyNode( n,
                                                   currentLevel,
                                                   parentToNodeList) )
                siblingList.add(n)
                n.visit ← true
            End If
        End If
    End For

    parentList ← siblingList
    siblingList.remove_all
End While
```

Figure 8. Pseudo-code for the contact graph

Once the cutting planes are identified, the system retrieves the contact graph to collect the selected objects based on the contact relationship and the boundary condition defined by the cutting plane. The group starts from the selected object. If any object has a higher hierarchy level number (lower in the tree) and its parent is already in the group, then the object is added to the group (Figure 10, objects B). If any object has the same hierarchy level and any of its vertices is on the positive side of the cutting plane, then the object is added to the group (Figure 10, object C). If any object has a lower hierarchy level, then the system checks if its child contact object is already in the group and if the object is on the positive side of the cutting plane. If the object satisfies these two conditions, then the object is added to the group. Figure 9 is the pseudo-code explaining how to determine the group of objects based on the user input and the contact graph. Figure 10 outlines the algorithm for the directional dragging technique.

Directional dragging: cutting_plane from dragging direction and the selected object to position the plane	*Anchoring & grouping:* cutting_plane from the contact surface between the anchored object and the selected object

```
currObject ←selected object
navList ← empty
connectedList.add ( currObject )

While currObject is not null
   For each object in hierarchyList

      do_add ← false
      If currObject.hierarchy < object.hierarchy
           If  object.isparent( currObject) = true
                    do_add ← true
      Else If currObject.hierarchy = object.hierarchy
              and contact( currObject, object) = true
              and isInCuttingPlane( object ) )
                    do_add ← true
      Else If  currObject.hierarchy > object.hierarchy
               and isInCuttingPlane ( object ) = true
               and object.hierarchy
                    = (any object in connectedList).hierarchy )
                     do_add ← true
      End If

      If  do_add = true
          navList.add( object )
          conncectedList.add( object )
      End If

   End For

   currObject ← navList.remove (object)
End While
```

Figure 9. Pseudo-code for grouping

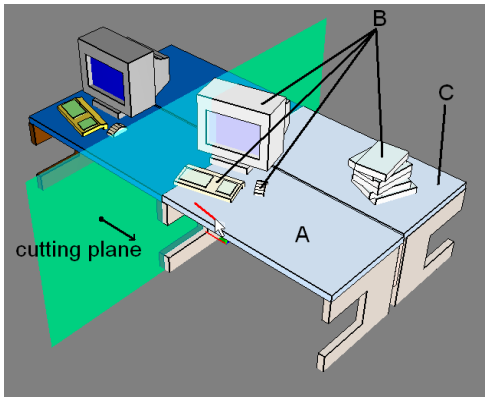


Figure 10. Group selection by directional dragging: The cutting plane is determined by the geometry of A and the dragging direction. Object A is selected and dragged into the direction. Objects B are selected since they are children of the existing group, and C is selected since it is a sibling of A and it is on the positive side of the cutting plane.

3.4 Example Scenes

The scenes depicted in Figure 11 are some of the examples that were built using the grouping techniques discussed above. During the construction of these scenes, users frequently selected a group of objects and repositioned it to reach the desired arrangement.

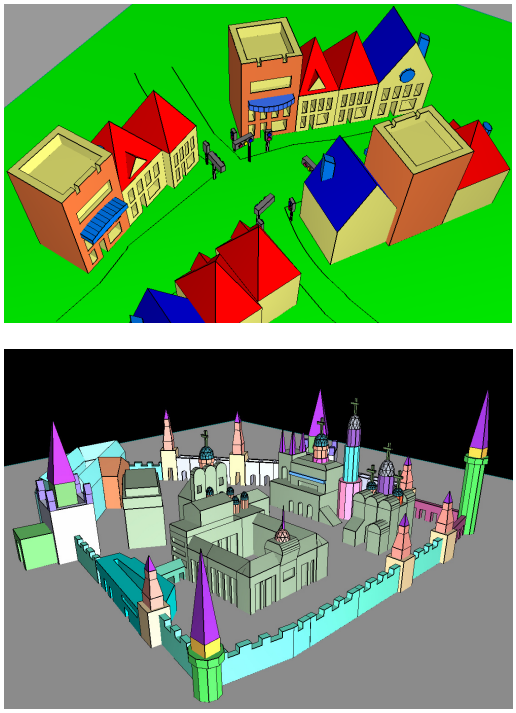


Figure 11. Example scenes

4 User Evaluation

We performed a 3 (grouping method) x 2 (hierarchy) factorial test, where the following three grouping methods were compared: directional dragging, anchoring and grouping, and rectangular selection. Rectangular selection was chosen as a reference, as this technique is the most commonly used group selection technique in both 3D and 2D interfaces. The hierarchy/non-hierarchy factor was added to investigate its effect on grouping techniques. As strong learning effects were observed in a pilot test for the hierarchy factor, we chose to perform a between subjects experiment for the hierarchy factor and a within subjects experiment for the grouping method factor.

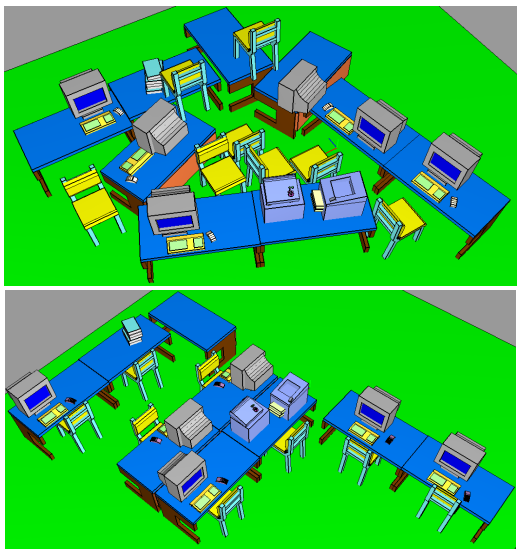


Figure 12. The task for the user study. (a) Initial scene, (b) Target scene.

Twelve paid subjects (4 females, 8 males, age 20-35) from the local university were recruited and each of them was randomly assigned to the hierarchy or non-hierarchy group (6 to each condition). All users had no prior experience with little to no 3D tool experience.

The task was to "clean up" a scene by aligning chairs and desks in an office room as illustrated in Figure 12. To complete the task, users had to perform multiple group selection, movement, and rotation operations. The task was designed to maximize the effect of grouping methods, while still being reasonably close to a real world task.

4.1.1 Results

Table 1 and Figure 13 show the task completion time for each technique. The difference among the three grouping techniques was not significant ($F_{2,10}$=1.63, p>0.05). The difference between hierarchical and non-hierarchical grouping was marginally significant ($F_{1,10}$=4.70, p≈0.055). This was mainly due to the insignificant difference between anchoring with hierarchical grouping and anchoring with non-hierarchical grouping ($F_{1,10}$=0.28, p>>0.05). If the anchoring technique is excluded from the analysis, the difference between hierarchical and non-hierarchical grouping was significant ($F_{1,10}$=9.69, p≈0.011). Anchoring seemed to operate most stably regardless of the existence of hierarchy support or not.

In addition, there was a significant interaction between the hierarchy factor and the grouping techniques factor ($F_{2,10}$=4.86, p<0.05). This means that the performance of the grouping techniques was affected by the hierarchy factor. Interestingly, the rectangular selection technique performed the worst when there was no hierarchal grouping, and it performed the best when using hierarchical grouping.

Table 1. Average completion time (sec) for each technique

	Directional dragging	Anchoring	Rectangular selection	Average
Hierarchy	252.99	273.2	248.83	258.35
No hierarchy	378.61	301.57	392.75	357.64
Average	315.80	287.40	320.79	

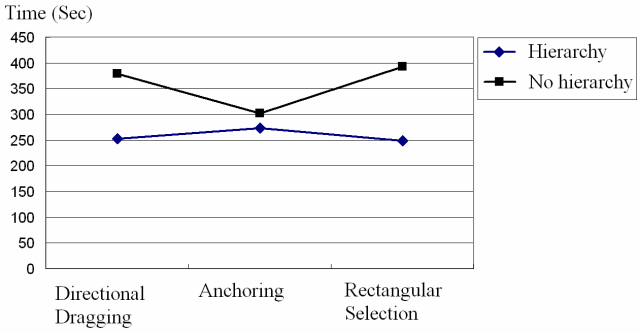


Figure 13. Completion time for each technique

4.1.2 Discussion

Overall, the hierarchy factor improved grouping operations for the directional dragging and rectangular selection techniques, but not for the anchoring and grouping method.

In directional dragging, dragging within a small distance triggers highlighting of the selected group. If the user drags the mouse further away, then the selected group separates and moves away, following the mouse motion. If the user stops at the highlighted status before dragging further, the highlighted group stays selected so that users can perform other operations, such as rotation. Users seemed to be confused by this continuous operation switch and complained that they had to look carefully to check if the desired group is selected before dragging further to move the group. Therefore, users frequently moved unintended groups of objects, which necessitated (potentially lengthy) corrective actions. This problem was also observed in a previous user evaluation of the Virtual Lego system [13]. Overall, we see this result as an indication that most users cannot easily adapt to the notion of directional dragging to select objects.

In anchoring and grouping, the relative position between the anchored object and the selected object is used to identify the selected group. Users felt that it was easier to predict which group would be selected based on the anchored objects. It is very interesting to note here that the support of hierarchy did not significantly improve the user performance for anchoring and grouping as it did for the other techniques. We see this as an indication that users had fewer problems predicting the results of a selection operation with this technique. We attribute this to the fact that in anchoring and grouping connected objects are grouped together by default, as this provides a similar functionality as hierarchical grouping does. Also, we observed that a few users did not understand how anchoring worked in the hierarchy condition. For example, they anchored all neighbouring objects that, due to the hierarchy, would have stayed in place anyways, thus increasing the completion time of the anchoring technique under the hierarchical condition. From this we hypothesize that anchoring can provide a good solution for group selection when it is difficult to derive hierarchical information due to the lack of a reference point – e.g. for a scene with objects floating in the air.

It is interesting to note that hierarchical grouping significantly improved the rectangular selection technique compared to the traditional, non-hierarchical method. One possible reason is that when there is no hierarchical information provided, users had to carefully check to determine if there were any objects hidden behind other objects, or they had to make sure that they selected all of the objects they desired. Users frequently neglected this confirmation step and later found that small, missed objects were floating in the air or attached at surprising places, which also resulted in sometimes lengthy correction episodes. As evidenced by the results, with the rectangular selection technique supported by a gravitational hierarchy, users can select groups of desired objects without much attention to hidden objects. The significant difference between the hierarchical and non-hierarchical factors suggests that rectangular selection supported by a gravitational hierarchy is a more intrinsically obvious selection method for users that have no training in conventional 3D packages and/or lack an understanding of 3D graphics.

4.2 Improvement on 3D Grouping Techniques

Based on the observation that the hierarchical grouping factor is important in scene manipulation, we propose a new simple grouping technique, called *multi-click grouping*.

In this technique, the first click on an object selects a gravitational hierarchical group – the object and objects on top of it, the second click on the object selects all the connected components that the object belongs to, and the third click on the object selects only the object. Clicking on the object again will revert back to the first alternative. Depending on the configuration of the scene, this loop of selections can have less grouping options. For example, if the selected object has another object on top of it, but no object connected in a bi-directional relationship, then the

selection result will iterate only between the hierarchical selection and the single object selection options.

This technique is analogous to the multi-click technique for text selection in Microsoft Word. There, the first click places the cursor onto the nearest character, the second click selects the word that the character belongs to, the third click selects the whole paragraph that the word belongs to, and the forth click loops back to the first option. However, in our multi-click group selection method, we placed the single object selection as the third in the order. We did this, as, according to our observations, hierarchical selection is more frequent than single object selection in 3D manipulation. If this does not hold in general, we can easily accommodate different scenarios by allowing the user to change the order of group selections via a configuration dialog.

Clearly, the new technique can also be combined with the traditional shift-click method to provide a way to select arbitrary groups.

The advantages of this scheme are: 1) it provides a quick way to select frequently selected groups (i.e. connected components), 2) it is orthogonal to other conventional grouping techniques and hence introduces minimum overhead into the user interface, and 3) it conforms with to the fact that most users cannot easily adapt to the idea of directional dragging.

While our initial experiences with an implementation of this idea are promising, multi-click grouping requires a more formal evaluation to determine whether it can really provide a better grouping scheme. This is clearly outside of the scope of this paper, and remains a topic for future work.

5 CONCLUSION

This paper presented several grouping techniques for efficient scene manipulation. We introduced the idea of hierarchical groupings to describe objects on top of each other, and bi-directional groupings for objects in proximity of each other. The novel aspect of these groupings is that they can be dynamically built based on a contact graph, instead of relying on semantic information or explicit user specification. Then we applied the idea of hierarchical grouping to several group selection techniques for 3D scenes.

The user evaluation results suggested that a gravitational hierarchy is a very important feature for the efficient selection and manipulation of object groups in scenes that resemble real world situations. The anchoring and grouping method also performed well in general, but did not outperform e.g. rectangle selection supported by a gravitational hierarchy.

Finally, and based on some of the observations in the user study, we proposed a new multi-click technique for 3D group selection. However, this technique needs to be verified through a user evaluation in the future.

REFERERNCE

[1] Balakrishnan, R., and Kurtenbach, G. (1999). Exploring Bimanual Camera Control and Object Manipulation in 3D Graphics Interfaces. Preceedings of CHI'99, ACM, 56-63.

[2] Biederman, I. Recognition-by-Components: A Theory of Human Image Understanding. Psychological Review, 1987, 94, 115-147.

[3] Bowman, D.et al, Virtual-SAP: An Immersive Tool for Visualizing the Response of Building Structures to Environmental Conditions. VR 2003: 243-250.

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

[5] Dijk, C., New insights in computer-aided conceptual design. Design Studies, 1995, 16, pp. 62-80.

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

[7] Ehmann, S.A. and Lin, M.C., Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching.In Proc. International Conf. on Intelligent Robots and Systems, 2000.

[8] Gottschalk, S.,Lin, M.C.and Manocha, D., OBB-Tree: A Hierarchical Structure for Rapid Interference Detection, Proc. of ACM Siggraph'96.

[9] Igarashi, T., Matsuoka, S., Tanaka, H.,Teddy: A sketching interface for 3D freeform design. SIGGRAPH'99.

[10] Igarashi, T., Hughes, J., A Suggestive Interface for 3D Drawing, ACM UIST'01:173-181.

[11] Jung, B., Latoschik, M., Wachsmuth, I., Knowledge-based assembly simulation for virtual prototype modeling. IECON'98 – Proceedings of the 24thAnnual Conference of the IEEE Industrial Electronics Society Vol. 4, IEEE, 1998, 2152-2157.

[12] Marr, D. Visual information processing: the structure and creation of visual representations, Philosophical Transactions of the Royal Society of London, Series B, Biological Science, 1980, 290 (1038)

[13] Oh, J.-Y., Stuerzlinger, W., A System for desktop conceptual 3D design, Virtual Reality 2004, 7: 198-211.

[14] Oh, J.-Y., Stuerzlinger, W., Moving objects with 2D input devices in CAD systems and desktop Virtual Environments, Graphics Interface 2005.

[15] Salzman, T., Stachniak, S., Stuerzlinger, W., Unconstrained vs. Constrained 3D Scene Manipulation, *Engineering for Human-Computer Interaction*, Eds. M. Little, L. Nigay, 207-219, May 2001.

[16] Sachs, E., Roberts, A., Stoops, D., 3-Draw: A tool for designing 3D shapes. IEEE Computer Graphics & Applications, 1993: 18-26.

[17] Smartscene Technology, available in Multigen-Paradigm products, see: http://www.multigen-paradigm.com.

[18] Smith, G., Salzman, T., Stuerzlinger, W., 3D Scene Manipulation with 2D Devices and Constraints, *Graphics Interface 2001*, Eds. J. Buchanan, B. Watson, 135-142, 2001.

[19] Stuerzlinger, W. & Smith, G., Efficient manipulation of object groups in virtual environments, IEEE VR 2002, 251-258.

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

[21] Wang, Y., & MacKenzie, C.L., The Role of Contextual Haptic and Visual Constraints on Object Manipulation in Virtual Environments. ACM CHI 2000: 532-539.

[22] Zeleznik, R.C., Herndon, K. & Hughes, J.F., SKETCH: An interface for sketching 3D scenes, SIGGRAPH'96.