Jian Chen*
Department of Computer Science
Brown University
Providence, RI 02912

Doug A. Bowman
Department of Computer Science
Center for Human-Computer Interaction
Virginia Tech
Blacksburg, VA 24061-0002

Domain Specific Design of 3D Interaction Techniques: An Approach for Designing Useful Virtual Environment Applications

Abstract

Few production virtual environment (VE) applications involve complex three-dimensional (3D) interaction. Our long-term collaboration with architects and engineers in designing 3D user interfaces (3D UIs) has revealed some of the causes: existing interaction tasks and/or techniques are either too generic when isolated from the application context, or too specific to be reusable. We propose a new design approach called *domain specific design* (DSD) that sits between the generic and specific design approaches, with an emphasis on using domain knowledge in 3D interaction techniques. We also describe an interaction design framework encompassing generic, domain specific, and application specific interaction tasks and techniques. This framework can be used by designers to think of ways to produce domain specific interaction techniques. We present a particular DSD method, and demonstrate its use for the design of cloning techniques in a structural engineering application. Results from empirical studies demonstrate that interaction techniques produced with domain knowledge in mind outperformed other techniques by improving task efficiency, work flow, and usefulness of the 3D UI.

1 Introduction

A great deal of work has been done, especially in the past decade, on the design of effective three-dimensional (3D) interaction techniques and user interfaces (UIs) for virtual environments (VEs). In the mid-1990s, many interaction techniques were invented for the so-called "universal tasks" of travel, manipulation, selection, system control, and symbolic input (Bowman, Kruijff, LaViola, & Poupyrev, 2004; Foley, Wallace, & Chan, 1984; Hand, 1997). This led to most of today's interaction research efforts on interaction styles, metaphors, and cognition for these tasks (Bowman et al., Plumlee & Ware, 2006; Zhai, Buxton, & Milgram, 1994).

Despite this research on 3D interaction, few production VE applications involve complex 3D interaction. Our experience suggests that at least two barriers exist. First, existing interaction tasks are generic and thus do not directly address application scenarios; second, the generic nature of tasks leads to the

Presence, *Vol. 18, No. 5, October 2009*, 000–000

*Correspondence to jchen@cs.brown.edu.

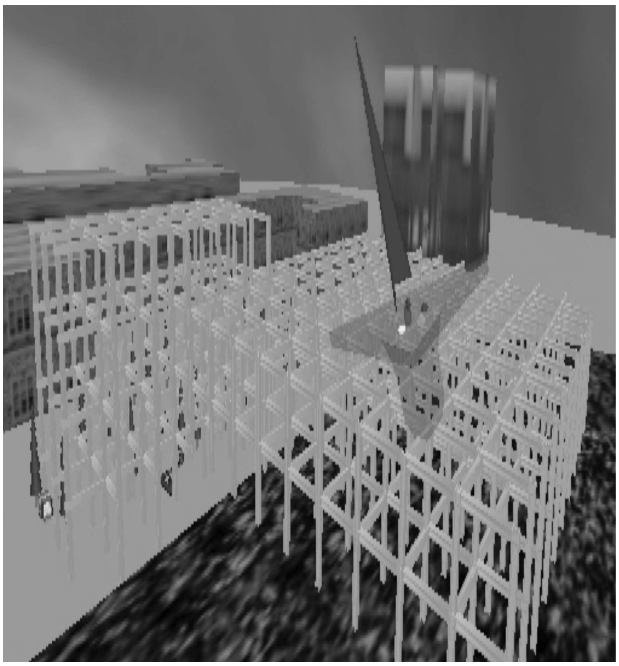

Figure 1. *A structure built by an architect using the Virtual-SAP application. Modeling such a large structure would be time-consuming if not impossible using conventional 3D interaction techniques in VEs; with our domain-specific interaction techniques, it takes less than 2 min.*

design of generic interaction techniques that may be impractical. For example, Go-Go (Poupyrev, Billinghurst, Weghorst, & Ichikawa, 1996) is a manipulation technique that is generic and should be applicable to any 3D UI. When used for modeling real-world buildings in a structural engineering application, however, Go-Go is not practical when the number of building elements exceeds a few dozen, such as the structure shown in Figure 1.

F1

We claim that one reason for the lack of broad usefulness of 3D interaction techniques is that these techniques have failed to exploit domain knowledge. Most techniques designed for universal tasks are too generic to be practical. Design methods, such as those related to usability engineering (UE), can be used to improve the work flow of the 3D UI, but do not address the design of 3D interaction techniques directly. In addition, developing 3D interaction techniques is complex and expensive, as directly migrating techniques from 2D to 3D does not always produce usable and useful techniques and the 3D design space is too large to investigate without any guiding principles.

In this paper we propose a domain specific design (DSD) approach that directly addresses the design of 3D interaction techniques and circumvents the limitations of generic design. DSD uses domain knowledge to structure design tasks and techniques. We also propose a new framework that classifies interaction design into three levels based on domain specificity, aiding in the assessment of a technique's reusability and design of domain specific techniques. We describe a three step DSD method that can be used by designers. Finally, we present the results of several studies indicating the effectiveness of the DSD approach.

2 Related Work

3D interaction is a type of human-computer interaction (HCI) in which tasks are performed directly in a 3D spatial context (Bowman et al., 2004). The value of 3D interactivity has been supported by much prior research. For example, effective 3D interaction has been found to improve task performance by an order of magnitude over conventional 2D input for some scientific visualization tasks (van Dam, Forsberg, Laidlaw, LaViola, & Simpson, 2000); interactivity is also claimed to provide better spatial understanding for training and modeling, and to improve the user experience (Allison, Wills, Bowman, Wineman, & Hodges, 1997).

The principal way to study 3D interaction is to design techniques for the "universal 3D tasks" of travel, manipulation, selection, system control, and symbolic input (Bowman, Gabbard, & Hix, 2002). Research in this area has addressed such issues as the empirical design and evaluation of displays (Lin, Duh, Parker, Abi-Rached, & Furness, 2002), design and comparison of novel interaction techniques (Bowman et al., 2004), evaluation in a specific application setting (Lin & Loftin, 2000), and the use of constraints and various

aids (e.g., navigation aids, Darken & Cevik, 1999; and physical props, Ullmer & Ishii, 1997).

Another approach to studying 3D interaction is to design a 3D UI for a specific application. For example, Hix and her co-authors (1999) used an iterative design method to develop a 3D UI for a military training application. In this design method, designers uncover usability problems through empirical studies and then tweak (often existing) techniques to work in that situation. Such "situated" design locates the design in a familiar world with specific practices, artifacts, goals, and plans. The goal, however, is to improve the overall usefulness of a 3D UI rather than to design interaction techniques. In fact, it has been argued that the UE approach does not suggest design solutions (Myers, 1994; Wolf, Rode, Sussman, & Kellogg, 2006) and designers do not always intend to create novel interaction techniques (Beaudouin-Lafon, 2000; Wolf et al., 2006). To our knowledge, few studies have attempted to advance design methods for 3D interaction techniques.

Understanding of 3D interaction design is still limited. Design guidelines exist, but they are targeted mostly at the standard universal tasks rather than real-world task scenarios (Bowman et al., 2004; Gabbard, 1997; Kalawsky, 1999; Sutcliffe, 2003). Our work addresses the design of domain specific 3D interaction techniques that apply to real-world tasks.

Several attempts have been made to guide the design process. Kaur, Maiden, and Sutcliffe (1999) extended Norman's seven stage HCI model to VEs to measure and improve the user's action sequences related to the work flow of the target use and developed a model integrating interaction and design guidance for VEs. Mills and Noyes (1999) proposed generic design issues for VEs, and Drettakis, Roussou, Reche, and Tasigos (2007) proposed improving work flow in order to increase the use of VEs in highly interactive applications. Hix and her collaborators extended conventional UE design practices to 3D UI design (Hix & Hartson, 1993; Hix et al., 1999). Our work focuses instead on improving the design of 3D interaction techniques by using domain knowledge to help us understand what to design.

Domain knowledge has proved its value in 3D modeling tools. For example, Wonka, Wimmer, Sillioin, and Ribarsky (2003) define semantic knowledge to transform, scale, and extrude texture when modeling large structures. Artifact knowledge, such as constraints, has been applied to define physically correct object behavior, for example, that objects sit on top of each other rather than floating in space (Bukowski & Sequin, 1995; Oh & Stuerzlinger, 2005; Wonka et al.). Domain knowledge has also been used in designing intelligent interaction techniques to improve work flow. For example, Feiner and MacIntyre (1993) designed a rule-based system for maintenance tasks in which a pointer automatically indicates the user's intended target. However, domain knowledge has not been used extensively in designing 3D interaction techniques and framing interaction tasks. Our work establishes a domain specific approach to 3D interaction design and introduces the concepts of domain specificity and domain specific design to guide designers' work and thinking. Thus, our aim is not to design intelligent interaction or improve work flow per se, but rather to use general domain knowledge to frame more meaningful 3D tasks and to produce more effective and efficient 3D interaction techniques for real-world use.

3 Motivating Example: VE for Structural Design

Our long-term collaboration with architects led to the design of Virtual-SAP, a head mounted display (HMD) based immersive VE for structural design, analysis, and visualization (Bowman et al., 2003). Architects using this program expect to construct realistic buildings, run stability tests using finite element methods, and modify the structure until a satisfactory design is achieved.

The first implementation of Virtual-SAP followed a usability engineering process for iterative design, prototyping, and evaluation (Hix et al., 1999). Task analyses were conducted with domain experts actively involved in collecting designer needs and making usability studies (Setareh, Bowman, & Kalita, 2005). Results from the task analyses were used for functional decomposition

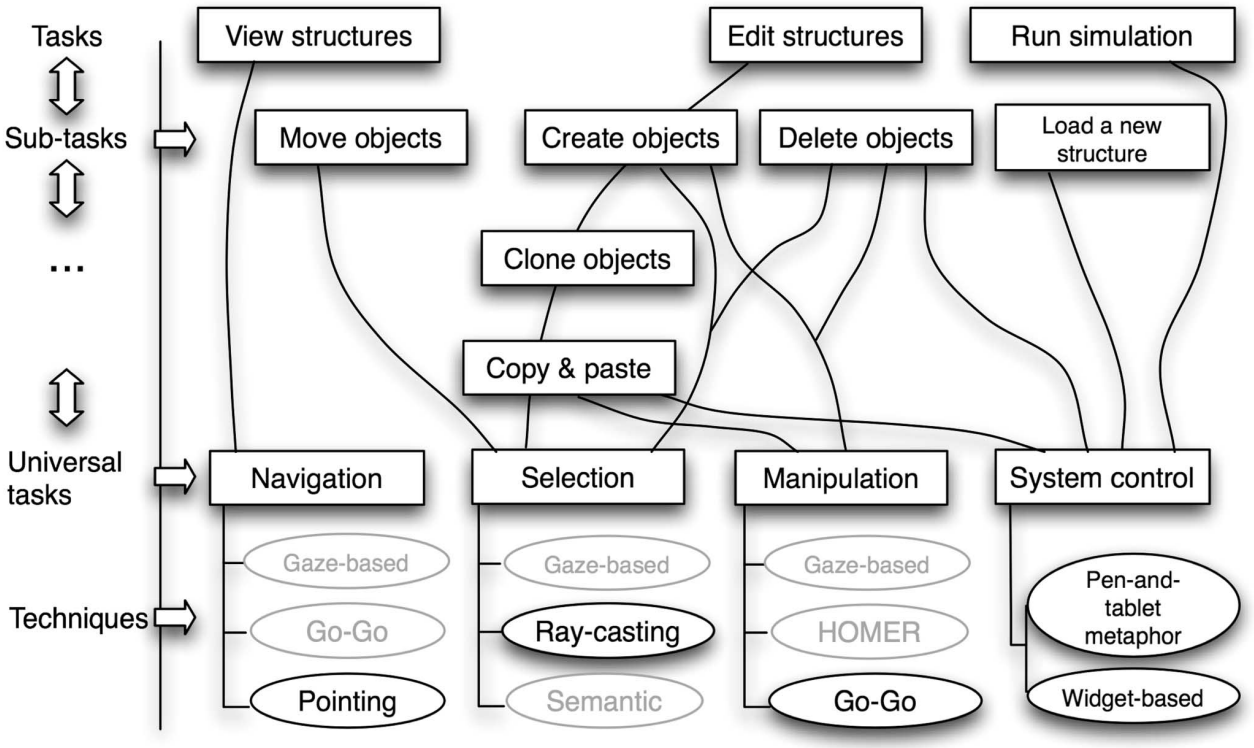

Figure 2. *Task decomposition in the original Virtual-SAP using the task composition and decomposition approach.*

and composition, a common method in design disciplines (Schön, 1983; Simon, 1981; see Figure 2). First, a top-down approach breaks the high-level activity (such as "edit structure") into small executable steps (such as "create objects" and "delete objects") based on work flow analyses. This decomposition continues until a technique is reached that can be used for a lower-level task. For example, "create objects" was decomposed into "clone objects" and further decomposed to "selection" and "manipulation." Finally, a bottom-up approach was used to adapt subsolutions (techniques such as ray-casting and Go-Go, Poupyrev et al., 1996) to obtain a group of interaction techniques supporting the application goal.

The initial design resulted in a pen-and-tablet-metaphor user interface for architectural modeling. The user held a tablet and used a stylus to operate widgets in order to load structural elements (beams, columns, and slabs). Having grabbed an object, the user could place it with Go-Go (Poupyrev et al., 1996). Finally, the user could run and observe simulations and edit the structure all within the same environment.

The results of several user studies suggested that the interface was relatively effective and had few usability problems (Bowman et al., 2003). When we gave the application to students and engineers for real-world structural design, however, we found that the UI was not very useful for complex structure modeling: it could take users hours to create and move the hundreds of beams and columns in a moderately complex structure.

This example demonstrates that decomposition and composition can result in less than optimal 3D UI design when only generic interaction techniques are available. We realized that the design issue was not to tweak existing interaction techniques but rather to define new tasks and techniques that meet the application require-

ments. Existing categories of interaction tasks, such as object selection and placement, were too generic to describe the needs of the application. But designing new tasks and techniques for each new application would not be practical or efficient. This led to our overall research question: how can we design useful and effective 3D interaction tasks and techniques without customizing them for each application?

It might seem that this problem could be addressed with further iterations of the usability engineering (UE) method, focusing specifically on the task of cloning, as in most conventional 2D interface design approaches or user centered design practice. We would argue, however, that UE is of limited value in producing effective interaction without a reasonably rich set of domain specific interaction techniques from which to draw.

Ultimately, all 3D UIs must be to some degree application specific in order to ensure usability and usefulness for real-world work. But the design of 3D interaction techniques need not always be application specific. For interaction techniques to be reusable and useful in designing 3D user interfaces, three design requirements must be met:

- Interaction tasks must ensure adequate representation of the domain in use.
- Interaction techniques must be useful in other applications without significant modification.
- A design method must allow designers to create a collection of interaction tasks and techniques that are supported by the usage guidelines.

4 Existing Approaches to 3D Interaction Design

Before introducing domain specific design, let us consider the characteristics of existing design approaches for 3D interaction. The approaches can be classified into two categories: generic and application specific design (see Figure 3). This classification makes use of the concept of specificity, which indicates the amount of knowledge used from the application domain in the interaction design. We describe interaction using

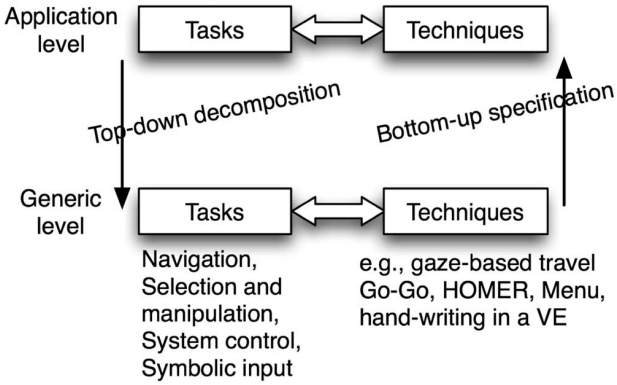

Figure 3. *Conventional two-level design framework.*

tasks (what to design for) and *techniques* (how to accomplish the tasks).

4.1 Generic Design Approach

We call a technique *generic* if it is designed for a generic (universal) task, with no specific application context. Thus, the specificity for this type of task or technique is the lowest. The pros and cons of this design approach include a well defined design space, widely available evaluation results, and low specificity/high reuse.

1. **Well Defined Design Space** Various taxonomies are available to describe the structure, action sequences, and parameter space of a generic task, and to classify generic techniques into useful categories (Bowman & Hodges, 1999; Poupyrev, Weghorst, Billinghurst, & Ichikawa, 1997).
2. **Widely Available Evaluation Results** Test beds have been designed to measure task performance and user behavior. Designers have plentiful resources from which to chose (Bowman, Johnson, & Hodges, 1999; Poupyrev, Weghorst, Billinghurst, & Ichikawa, 1998; Tan, Robertson, & Czerwinski, 2001).
3. **Low Specificity/High Reuse** The techniques are designed not with any particular application or

application domain in mind, but rather to work with any application. When used in real-world cases, however, significant modifications might be required to meet application requirements. In some cases, generic techniques and tasks will be completely inadequate to meet application needs.

When the generic design approach is used, high-level user activities are decomposed into simple subtasks to form a hierarchical representation of user activity. Then the designer decides how to integrate solutions to the generic tasks into a final solution to the high-level design task.

This approach was used in the original Virtual-SAP application described in Section 3. The modeling task was decomposed into manipulation and selection tasks. We then selected several practical generic interaction techniques (e.g., Go-Go) and considered cross technique consistency issues to generate the final application.

4.2 Application Specific Design Approach

The clearest way to improve usability in any particular application is to design the interaction tasks and techniques specifically for that application. A technique designed for an application specific task, called an *application specific technique,* often uses specific application domain knowledge (see Section 5.1) that might not be reusable. This approach typically requires starting from scratch with each application and using a complete usability engineering process to produce a usable 3D UI. Design with this approach has the properties of (1) less well-defined design space, (2) techniques designed with context and activity in mind, (3) high specificity/low reuse, and (4) high cost.

1. **Less Well-Defined Design Space** Because there is no interaction technique for a given application specific task, designers must perform task analysis to determine design goals. Well-known methods include field studies, interviews, and ethnomethodological studies.
2. **Techniques Designed with Context and Activity in Mind** Because the tasks are framed within a specific application context, the techniques are generally usable and useful for that context only.
3. **High Specificity/Low Reuse** There are many definitions of reuse, such as the reuse of concept solutions, and best practice software modules and patterns. In this paper, reuse simply means the reuse of interaction techniques, that is, how easy it is to plug in and play a certain technique in an application. Interaction techniques designed within a specific application context will be highly useful and usable in that context. However, it may be difficult to reuse them in other applications without modification. Thus, an interaction technique designed for a particular use is often too specific to be broadly applicable.
4. **High Cost** Designing for every application is costly. Making a range of techniques available that designers simply plug into an application will be more cost-effective. The design often requires integration of and smooth transitions among several interaction techniques that best fit a specific application. For example, Hix et al. (1999) designed several interaction techniques for battlefield visualization applications. Some components of this type of interaction technique can be reused, because some fundamental design principles are invariant over many applications, but the technique as a whole is not reusable.

We developed the DSD approach in order to balance the advantages and disadvantages of generic and application specific interaction design.

5 Domain Specific Approach to 3D Interaction Design

We define domain specific design (DSD) as an interaction design approach in which tasks or techniques are designed with domain knowledge in mind. Our approach is similar in spirit to procedure modeling (Bukowski & Sequin, 1995) regarding the use and clas-

sification of domain knowledge. For example, it is possible to describe the appearance of a building (e.g., the shape of the land, land use, and streets) using semantics. Our work, however, focuses on interactivity rather than automatic model generation.

5.1 Domains and Domain Knowledge

Before discussing DSD, it is useful to define what a domain is. We define a *domain* as a subject matter area of study that can have subdomains. Often, a domain has its own processes, data, objects, and artifacts that behave according to the rules of that domain. Thus, the study of a domain, or domain analysis, can determine the generalized characteristics of a particular domain. In particular, domain knowledge has the following characteristics that make it suitable for interaction design: Stability, intuitiveness, feasibility, and reusability.

1. **Stability** Domain knowledge is stable and reusable, since multiple applications can share common knowledge. UI designers can benefit from a stable collection of tasks and techniques based on domain knowledge, despite changing application requirements (Chen & Bowman, 2006; Chen, Bowman, Lucas, & Wingrave, 2004; Sutcliffe, Benyon, & van Assche, 1996).
2. **Intuitiveness** Domain knowledge is generally intuitive or systematic for uses in the domain. An interface designed in line with that knowledge typically provides a better mental model and understanding of the environment. Users' familiarity with the task definition can make the interface more intelligible and predictable.
3. **Feasibility** Domain knowledge can be collected effectively by knowledge elicitation techniques (Gordon, 1994; Gordon, Schmierer, & Gill, 1993; Wielinga, Schreiber, & Breuker, 1992).
4. **Reusability** Since knowledge is shared among applications, techniques can be reused within a domain or even in another domain with similar characteristics.

Domain knowledge can be general or particular. To illustrate, consider the task of cloning in the sense of building modeling. The concept of cloning is general domain knowledge, since the distinction between a window, a beam, or a chair is irrelevant—they are equally good for cloning. However, the shape of beams in welding operations is particular domain knowledge, since differently shaped beams would sustain different loads. This knowledge would lead to the design of techniques that would be difficult to reuse in other situations. The DSD approach targets the general type of domain knowledge to make the techniques more reusable.

5.2 Domain Specific Tasks and Interaction Techniques

Domain specific tasks and techniques are two components of DSD. A domain specific task is an abstraction of domain activities that can be shared among applications in the targeted domain. Activities define what users usually do in the real world, while goals and subgoals of an activity are described as tasks through task analysis (Diaper, 1989; Kirwan & Ainsworth, 1992; Moran, 1983). A domain specific interaction technique is a way to accomplish an interaction task that makes use of domain knowledge. The goal is to increase the effectiveness of interaction.

5.3 Three-Level Design Framework

We extend the two-level framework shown in Figure 3 by adding a third, domain level. This 3D interaction design framework (see Figure 4) describes the design of tasks and techniques at the generic, domain, and application levels and specifies paths designers can take in designing and/or choosing interaction techniques for an application. This framework can be used to describe all three of the design approaches we have discussed (generic design, application specific design, and DSD). The goal of the DSD approach is to reach the "Techniques" box at the domain level. These domain specific 3D interaction techniques can be used in real-world applications within the particular domain.

F4

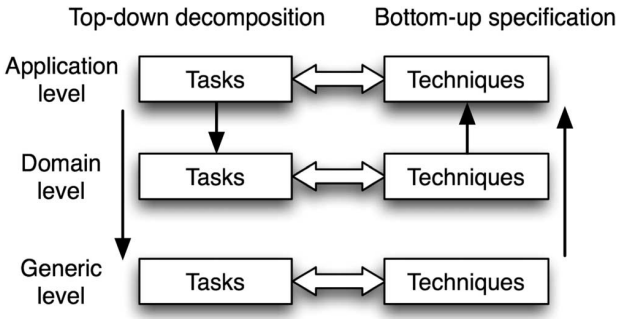

Figure 4. *3D interaction design framework.*

5.4 Advantages and Disadvantages

DSD attempts to maximize the advantages and avoid the disadvantages of the generic and the application specific methods. DSD has the characteristics of a less well defined design space, reusable domain specific techniques, and high initial cost, low long-term cost.

1. Less well-defined design space: As in application level design, the design space for domain knowledge and domain specific interaction does not exist. However, such knowledge can be collected and captured using the same methods as in application level design.
2. Reusable domain specific techniques: The techniques are designed with a domain in mind, and thus can be reused in other applications in the same domain without significant modifications. For example, the cloning operation (see Section 7) can be used in any application requiring multiple object creation.
3. High initial cost/low long-term cost: The initial design can be costly because it requires knowledge space formation and significant evaluation. However, the long-term benefits are also high because the resulting techniques are reusable. DSD becomes cost-effective once we have a range of domain specific interaction techniques that designers can use.

It is important to note that the three design approaches (generic, application specific, and domain-specific) are not distinct, but rather form a continuum. In fact, the same task can often be described in terms of any approach. For example, the domain specific task of cloning might be specialized from the generic task of numeric input. One benefit of separating the three design approaches is to assist designers in thinking about design goals and the utility of each approach, allowing them to choose an approach that is appropriate to the application they are designing.

6 Domain Specific Design Method

The previous section described the DSD approach at a high level. In this section, we present a more detailed DSD method—a three-step process that designers can follow to design domain specific 3D interaction techniques. In this DSD method, the designer performs three steps: knowledge elicitation and representation, task identification, and design interaction techniques.

6.1 Knowledge Elicitation and Representation

The first step of the DSD method is to describe knowledge representation based on target use. Knowledge is first acquired during the design and evaluation process, from the conversation with domain experts, and from observation of work and artifacts in the domain. Designers ask experts to identify the bottlenecks with existing tools (2D or 3D), leading to new design solutions. Designers explore the range of activities that experts conduct, the function of artifacts, and the pros and cons with conventional tools and media. Domain experts can also be invited to comment on the design of the evolving 3D UI on a regular basis. The knowledge thus elicited can be used to help frame tasks and choose parameters that are critical to improve the usefulness of the domain specific interaction techniques.

Such iterative design and testing provides valuable insights for the generation of domain specific tasks and techniques and for choosing critical design parameters to form a knowledge base. Getting experts to use the

interactive system is also important because people are more likely to recognize problems while interacting.

We are aware of a growing body of literature that is directly concerned with understanding use in context as a basis for design, including ethnographic approaches, scenarios, critical incidents, and personal construct. We did not make use of a specific method, since each of these techniques is suited to particular situations and delivers particular kinds of results. There is no guiding principle indicating which techniques are suitable or unsuitable for eliciting certain types of knowledge. We use a mix of these approaches.

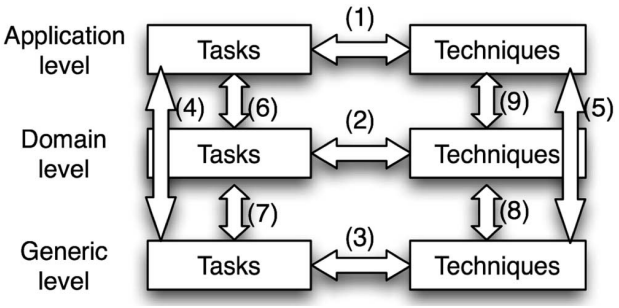

Figure 5. *Alternative design in the 3D interaction design framework.*

6.2 Task Identification

The second step of the DSD method is to specify interaction tasks based on domain knowledge and its representation. Tasks are formally described by means of a taxonomy, an analytical description of features meaningful in an application domain. The taxonomy typically describes the set of parameters that characterize the identified task and possible values for those parameters. This description serves as a knowledge base for designing interaction tasks and techniques.

6.3 Interaction Technique Design

The last step of the DSD method is to design interaction techniques for the domain specific tasks. We use test bed design and evaluation (Bowman et al., 1999) to design domain specific interaction techniques. In the test bed, we design a range of interaction techniques for the domain specific task by following different paths (see Figure 5) in our design framework and by integrating different flavors of domain knowledge.

F5

Specialization, generalization, direct design, and mixed decomposition and composition approaches are some of the possibilities described by the framework shown in Figure 5. In *specialization,* we start with the domain specific task, recharacterize it as a generic task, select a generic interaction technique for that task, and then specialize the generic technique by adding domain knowledge (paths 7, 3, and 8). In *generalization,* we cast the domain specific task as an application-specific task, design an application specific technique for that task, and then remove some specific domain knowledge to design domain specific interaction techniques from the application level (paths 6, 1, and 9). In the *direct design* approach, we can design domain specific techniques directly by following horizontal path 2. In *mixed decomposition and composition,* we recast a domain level task to a generic task then reuse generic techniques without modification (paths 7 and 3).

One advantage of using this framework is to make it possible to reuse existing generic-level or application-level techniques and to cover a broad range of techniques. The framework also covers existing interaction tasks and techniques and can be used to describe interaction design in the overall 3D UI design process. In our original Virtual-SAP design, we used this approach (starting with application specific tasks, then following paths 4, 3, and 5) to design the 3D interaction techniques. Following path 4 decomposes an application level task into the generic level; following path 3 produces a generic level interaction technique; following path 5 lets one specify an interaction technique to fit an application.

7 Using DSD for Virtual-SAP

We illustrate the design method by our case study application, Virtual-SAP. As described in Section 3, the purpose of Virtual-SAP is to let the user model building structures while immersed in a VE. We use the domain

Table 1. *Domain Knowledge Organization*

AQ: 1

Category	Characteristic
Characteristics of domain objects	
Geometrical	Standard size and shape
Physical	Objects have mass, material, texture, etc.
Mechanical	Objects are rigid
Characteristics of the environment	
Spatial	Repetitive and symmetrical patterns, large in size
Dynamic	Simulation followed objects' properties
Topology	Object relationship
Characteristics of the users	
Mental model	Workflow/domain of study
Attitude	Cognitive and noncognitive abilities
Expertise	Novice, expert

specific task of cloning (Chen & Bowman, 2006; Chen, Bowman, Lucas, et al., 2004; Chen, Bowman, Wingrave, & Lucas, 2004) as an illustration of the use of the DSD method. Choosing the architecture domain and cloning also lets us use the knowledge framed in 2D desktop tools, so that we can examine the effectiveness of the framework and the DSD method, rather than the effectiveness of knowledge acquisition.

7.1 Knowledge Elicitation and Representation

We used interviews with domain experts, observation of experts and students in the architecture and structural engineering domains, and comments from users of our early prototypes to elicit domain knowledge. We classified the domain knowledge into three categories, based on object, environment, and user's characteristics (see Table 1). These factors were found to be important for a structural engineering application while iterating our design with structural engineers. The most important knowledge for our application is probably the geometric arrangement of structural elements (e.g., beams, columns, slabs, and walls, etc.) in space. These objects obey certain geometrical, physical, and mechanical rules that form the knowledge base we used in designing interaction techniques.

T1

We implemented several geometry constraints based on this domain knowledge.

Coincidence The system defines point coincidence and element coincidence. Beams and columns are connected. Two elements have the same algebraic representation, the same position, and the same orientation.

Parallelism Objects follow regular shapes. Beams and columns that are parallel remain parallel.

Distance and Size Constraints Distance can be specified between elements and this distance defines the size of an object. Any distances used in the system are integers or integers plus 0.5 mm for ease of manufacture.

Alignment Objects are snapped to a grid and thus can automatically change their length when the size of the grid is updated.

7.2 Task Identification

By knowledge elicitation with people and artifacts in the architecture domain, we found that many struc-

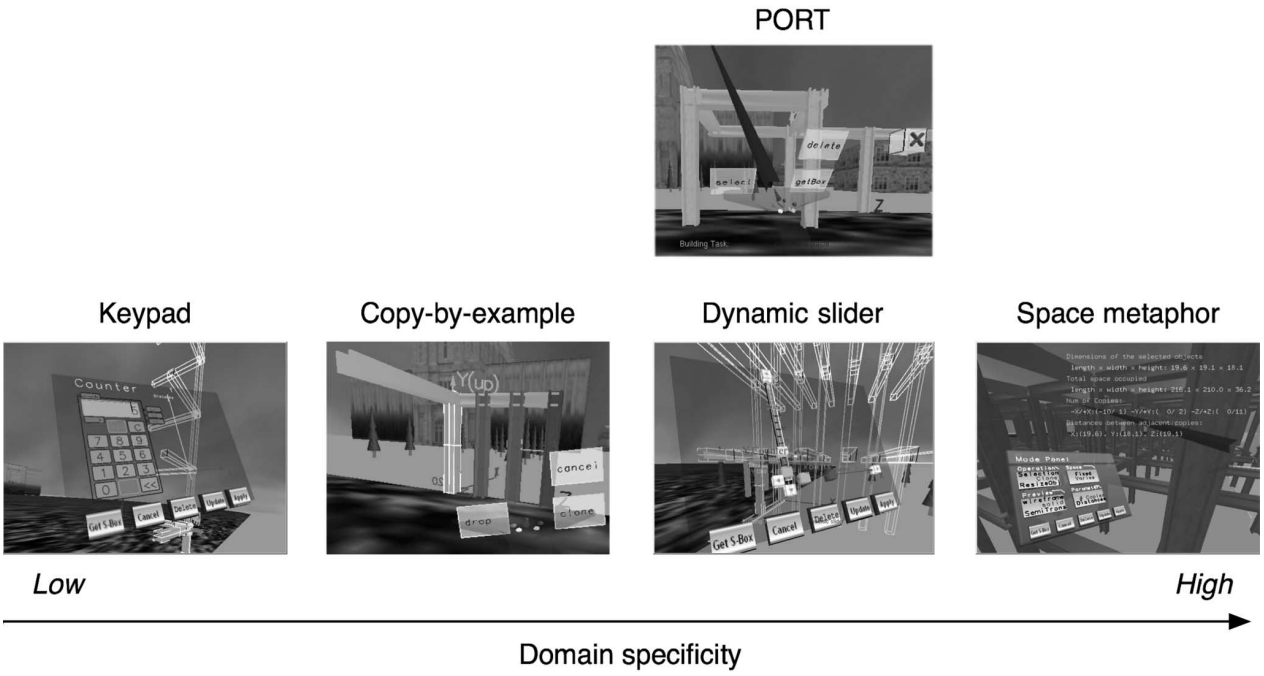

Figure 6. *Various cloning techniques sorted by domain specificity.*

tures contain spatially repeating elements. Thus the cloning task was identified. We have defined cloning as the task of generating multiple copies of spatially distributed objects in an environment (Chen, Bowman, Lucas, et al., 2004). This is a similar concept to instancing multiple copies used in desktop computer-aided design tools, such as AutoCAD, SketchUp, and others. Cloning is a higher-level task compared to selection and manipulation. Interaction techniques for cloning allow users (here architects) to take the techniques as a whole rather than performing the individual small steps of manipulation to create copies. It might seem obvious to design for cloning; however, our experiences suggested this was not the case. In the initial design, we tended to choose a set of good existing generic interaction techniques rather than designing new techniques.

We further learned that these elements can be described by a set of parameters, including the distance between spatially distributed copies, number of copies, shapes of the cloned objects, and visual attributes such as color and size. These parameters were placed in a formal taxonomy of the cloning task (Chen, Bowman, Lucas, et al., 2004).

7.3 Interaction Technique Design

We designed a series of domain specific interaction techniques at various levels of specificity by making use of domain knowledge (see Figure 6). Complete descriptions of these domain specific interaction techniques appear in our previous publications (Chen & Bowman, 2006; Chen, Bowman, Lucas, et al., 2004). Here, we use these techniques to illustrate how we have made use of the framework to produce a range of interaction techniques.

F6

7.3.1 Specializing Generic Techniques. One way to design domain specific interaction techniques is to specialize generic interaction techniques by adding domain knowledge. For instance, the copy-by-example technique was a specialization of the generic HOMER technique (Bowman & Hodges, 1997).

Copy by example has relatively greater domain specificity compared to HOMER because it uses repetition, a common characteristic in building and construction. Instead of placing each object individually, copy by example only requires the user to place the first copy; the rest are placed automatically when the user issues a "paste" command by a button click. The user needs to reposition only if the pattern changes—the program calculates the next position accordingly. With this technique, users can build stairs, towers, and buildings that are not necessarily square in shape.

7.3.2 Designing Directly for Domain-Level Tasks. We can also design techniques directly for the cloning task using horizontal path 2 from Figure 5. For example, PORT takes into account several constraints of structural design, such as the square shape and the repetitive distribution of architectural elements. The dynamic slider techniques uses a pen-and-tablet interface while PORT uses a direct pointing technique to specify the number of copies and the distances in each direction. The user can push a joystick on a handheld wand device to change the number of copies and the distances between copies. Both methods support continuous input and the axes of the sliders and the directions of wand pointing indicate the direction of input in an intuitive manner.

Another example is the space metaphor interaction technique, which uses PORT and also allows users to explicitly define the horizontal span (dimensions in space that a building will occupy) before cloning. The span constraint was suggested by our architect collaborators. Using the space metaphor interaction technique, adjusting the distance between columns or beams changes the number of copies, so the building is always confined to the prespecified area. Any objects out of the span are automatically erased. This technique had the highest level of domain specificity among those we designed.

7.3.3 Mixed Decomposition and Composition Approach. One way to think about the cloning task is to recast it (path 7 in Figure 5) as a numeric input task, which is a symbolic input task at the generic level; therefore any interaction techniques that allow users to input numbers should work. The keypad technique we designed for cloning used very little domain knowledge; users simply specify which building axis they are working with and input a number.

8 Evaluation of DSD

We conducted a series of user studies on the effectiveness of the DSD approach in the case study application. Details are given elsewhere (Chen & Bowman, 2006); here we briefly summarize the evaluation method and results. Although these studies do not compare DSD directly with other design approaches, they provide helpful insights into the relative usability and usefulness of 3D interaction techniques produced by DSD and by more traditional design methods.

8.1 The Effect of a Domain Specific Task

The purpose of the first experiment (Chen, Bowman, Lucas, et al., 2004) was to study the effects of considering a domain specific task during design. A domain specific cloning technique (PORT) and a generic object manipulation approach (HOMER) were compared. Users performed three tasks ranging from simple (fewer than 10 elements) to complex (200 or more elements) building modeling. We found that the domain-specific task was suitable for describing users' activities in modeling complex structures. PORT outperformed HOMER for large building modeling.

8.2 The Effect of Domain Knowledge Use

The second experiment studied the effect of using domain knowledge in designing interaction techniques (Chen & Bowman, 2006). Comparing the five domain specific interaction techniques presented in Section 4, we found that the usefulness of the interaction was associated with specificity—the greater the domain specificity, the better the task performance when modeling a large number of structural elements. As Figure 7 shows, F7

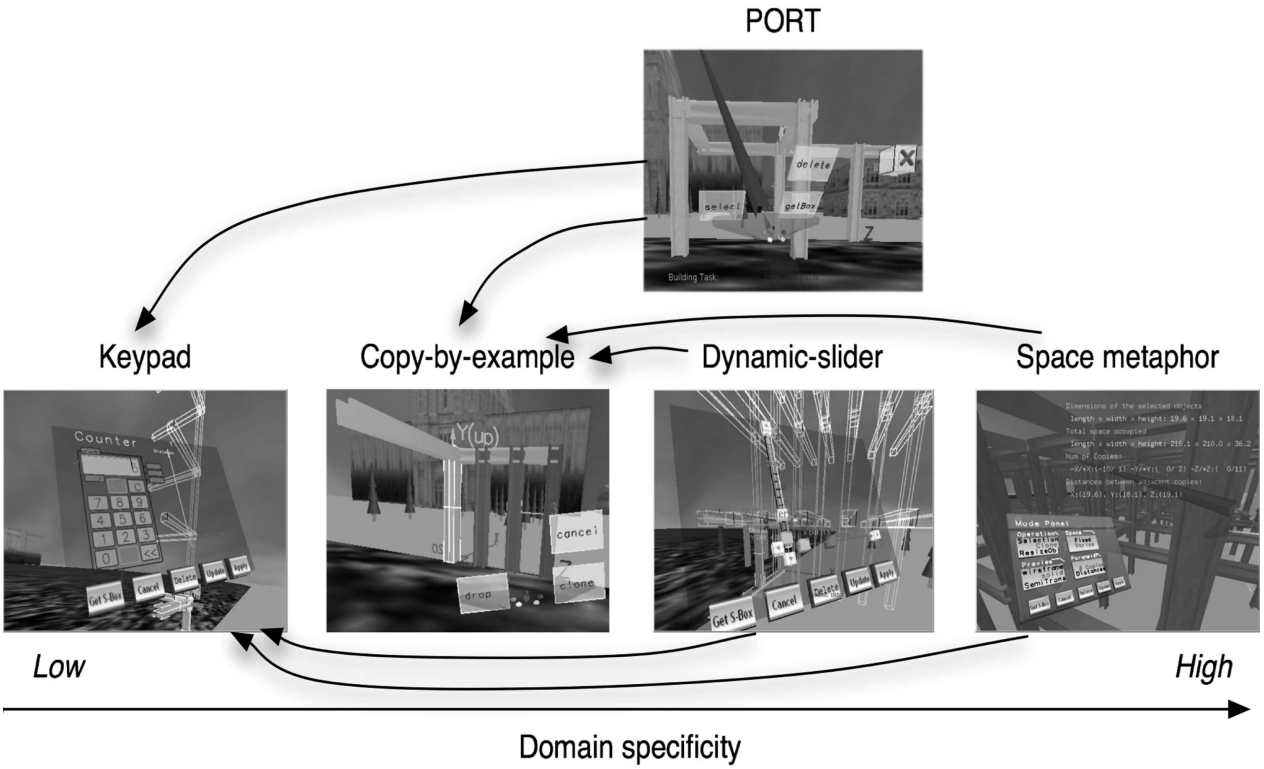

Figure 7. *Techniques designed with higher specificity lead to shorter task completion time than lower ones when generating hundreds of architectural elements. Lower specificity increases flexibility in the range of objects that architects can build. Arrows indicate a statistically significant difference for overall task completion time.*

the space metaphor, PORT, and dynamic slider techniques (those with greater domain specificity) were significantly faster than the copy-by-example and keypad techniques for a modeling task where thousands of elements were to be built. On the other hand, techniques with less domain specificity support a wider range of tasks. For instance, the copy-by-example technique can be used to build structures of any shape that has repetitive patterns, since it does not require a regular square shape. The keypad technique is flexible enough to be used in any numeric input task.

Participants commented that they preferred copy by example for simple tasks and PORT or the space metaphor for complex modeling work. When participants created only one element, they perceived the task as requiring a single object copy and placement rather than a cloning operation.

8.3 The Effect of DSD on a Complete Application

DSD will not be practical if the interaction techniques developed cannot be used in real-world applications. We therefore designed a new version of the Virtual-SAP application by integrating several new cloning techniques. The keypad, copy-by-example, and PORT cloning techniques were incorporated so as to provide alternatives to the user in modeling a structure. Figure 8 shows the new design with multiple object selection and cloning. A pen and tablet metaphor user interface is also used in this design. A user study was conducted to examine whether results produced in DSD increase the overall usefulness of Virtual-SAP. F8

The empirical evaluation (Chen, 2006) was conducted with two groups: a control group using a version

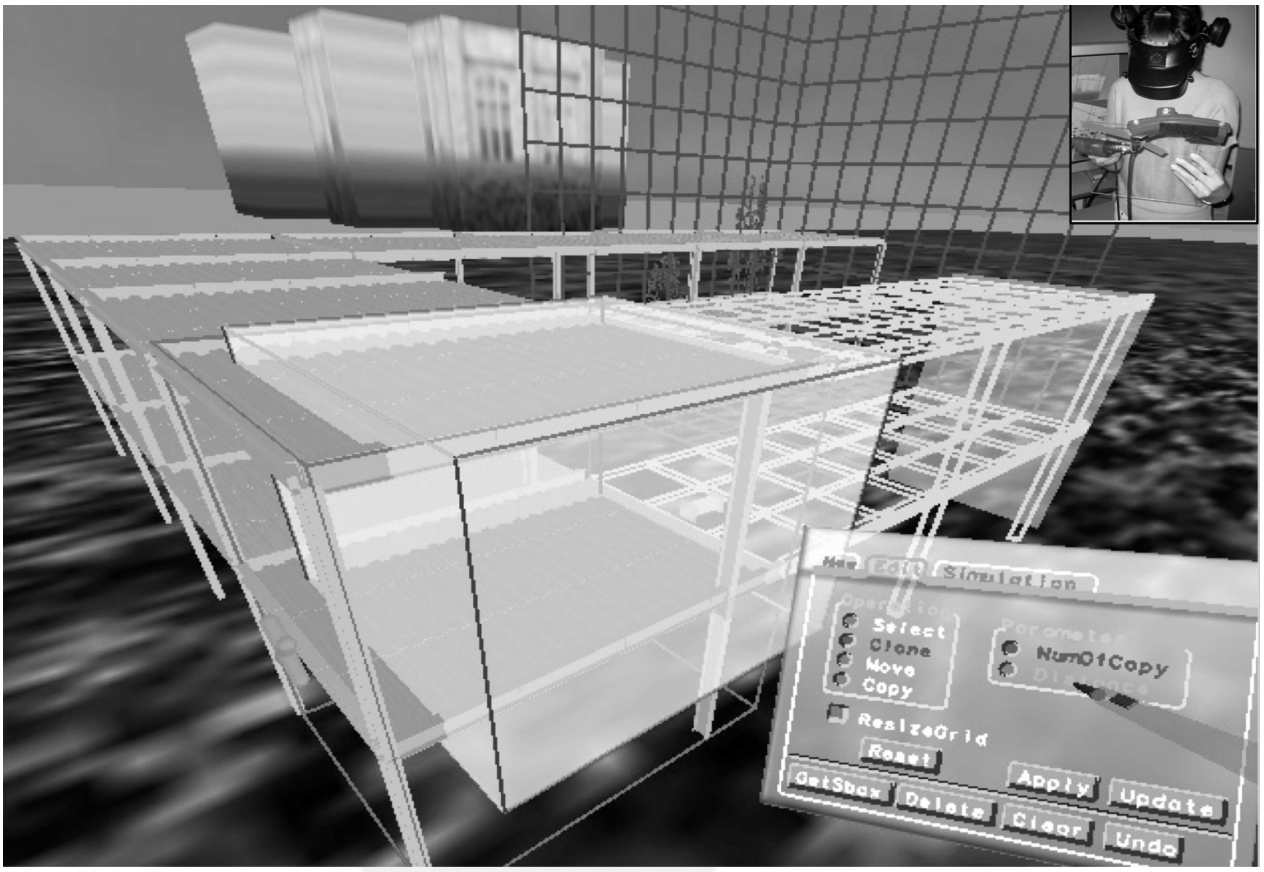

Figure 8. *New Virtual-SAP user interface. The semi-transparent box was used for multiple object selection: objects intersecting with the box were selected. The wireframe structure was generated using one of our cloning techniques.*

of Virtual-SAP without support for cloning and an experimental group using an application version that was identical except for the inclusion of a cloning technique (PORT). Both user interfaces had rich functionality, such as choosing architectural elements, placing objects, and visualizing earthquake simulations. The VE system was HMD-based.

Participants were architectural engineering students and were grouped into pairs. In each pair, one participant used the system to build a structure that he or she had previously drawn on paper, while the other participant watched the process on a monitor. They were allowed to talk about the modeling process. We collected participants' verbal communication with their partners, task performance, comments, and the structures they designed.

The results showed that participants from the domain specific group modeled a wider range of complex buildings. Many of the conversations in the domain group were regarding activity-relevant information (e.g., what is the horizontal span of a building), as compared to action-based information (e.g., how to place a beam) in the generic group. As expected, the domain specific group had better task performance.

8.4 Discussion

The results from the experiments favor the user interfaces produced by DSD. Considering domain specific tasks and using domain knowledge in design led to higher performance on individual tasks, and integrating

domain specific interaction techniques into an existing application led to greater overall usefulness.

Of course, increasing domain specificity can also reduce the reusability of an interaction technique. For example, the keypad can be used for any numeric input task, not just in the architecture domain. Copy by example is flexible enough to be used to form shapes similar to spiral stairs and frame structures for bridges.

Our evaluation of interaction techniques and the Virtual-SAP application provides empirical support for the effectiveness and general principles of DSD, although we cannot claim from these results that DSD is superior to other design methods for 3D interaction. An alternative way to compare design methods would be to compare the training time and resources needed for different approaches. We did not do this because we believe that the increased cost of DSD arises from the cost of collecting domain knowledge in order to frame domain specific tasks and thus design domain specific interaction techniques. Such costs exist, in the form of task analysis, in any effective design method.

9 Conclusions and Future Work

We have described DSD, a design approach for 3D interaction techniques with the ultimate goal of providing effective and efficient 3D UIs for real-world applications. We have presented a 3D interaction design framework that describes the relationship between DSD and the existing generic and application specific design approaches. The framework can help designers understand and think more systematically about 3D interaction tasks and 3D interaction techniques. We have also introduced a three-step DSD method that designers can follow for the development of domain specific 3D interaction techniques. Finally, we illustrated the use of the DSD approach and method through a case study in the architecture domain, and presented evidence for the effectiveness of DSD through empirical studies.

Major empirical findings from this research include the following. First, an interaction technique for a domain specific task increased usability over an interaction technique designed for generic level tasks. Second, in general, techniques designed with more domain specificity outperformed techniques with less domain specificity for domain specific tasks. Third, integrating a domain specific technique into an application yielded an effective and efficient interface.

The current research is limited in that we have conducted only one case study and did not perform and compare formal domain knowledge acquisition and organization methods. In addition, our ability to produce many interaction techniques might be due in part to our expertise rather than to the DSD method alone. DSD is a high level method, not a detailed process. We leave these topics as future work.

We also plan to further address the reuse of domain specific interaction techniques. One possibility is to create a mapping between knowledge representation and interaction techniques, and build a collection of knowledge possessed by domain experts. In this way, knowledge and the relevant interaction techniques can be processed, stored, and improved over time.

We also hope to broaden the impact of our theories of domain specific interaction techniques. For example, the domain of visual analytics that includes multivariate and multidimensional data sets is a promising area for future research in domain specific interaction.

Finally, we plan to explore other types of specificity to improve 3D interaction. For example, emerging technologies such as large, tiled, high-resolution displays may require display specific 3D interaction techniques to ensure usability.

Acknowledgments

This work was supported by the National Science Foundation under grant NSF-IIS-0237412. The authors gratefully acknowledge the support of Dr. Mehdi Setareh, who was actively involved in the design and evaluation process. The authors also appreciate those who gave their valuable time to participate in the empirical evaluations.