7. International Immersive Projection Technologies Workshop
9. Eurographics Workshop on Virtual Environments (2003)
J. Deisinger, A. Kunz (Editors)

# The ToolFinger: Supporting Complex Direct Manipulation in Virtual Environments

Gerold Wesche

Fraunhofer Institute for Media Communication IMK, Sankt Augustin, Germany

**Abstract**

*Virtual environments (VEs) such as the Responsive Workbench were designed to support interactive, complex applications. The possibility of direct manual interaction with virtual objects, using both hands in a natural, intuitive way is one of the great advantages of such configurations. The related work on interaction techniques for these environments is mainly concentrated on hand-oriented, spatial manipulation. With increasing complexity of applications, application control issues become more relevant, including the problem of tool selection. In this paper, we propose a new interaction technique, optimized for handling tool selection tasks comfortably, called the ToolFinger. The ToolFinger is a hand held 3D widget, resembling a pen, to which a set of tools is assigned in a way that selecting and applying a tool are no longer two separated tasks. The ToolFinger concept allows a fluent integration of tool selections and tool switches with the workflow of geometry editing. It benefits complex manipulation tasks in applications such as immersive geometric modelling, or interactive visualisation.*

Categories and Subject Descriptors (according to ACM CCS):
H.5.2 [Information Interfaces and Presentation]: User Interfaces – Interaction styles
I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques
I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Virtual reality
J.6 [Computer-Aided Engineering]: Computer-aided design (CAD)

Additional Key Words and Phrases:
Virtual environments, Application control, System control, Tool selection, Immersive modelling, Responsive Workbench

## 1. Introduction

Among projection-based virtual environments, table-like systems such as the Responsive Workbench [10] or the Immersa Desk [6] are configured as working environments, providing a virtual workspace where both hands directly interact with virtual objects. For example, the horizontal projection screen of a Responsive Workbench forms a manipulation space located in front of the user, so that his hands can directly grab and manipulate virtual models.

At the first sight, these characteristics make workbench-like systems very attractive to support applications that have a high degree of interactive complexity, like 3D visualisation or even immersive modelling. The interaction techniques developed for direct manipulation of geometry and data in such systems in many cases use metaphors from the physical world, which makes them very intuitive. The arguments for the suitability of VEs are largely based on this way of designing interaction.

From using 2D desktop applications, however, we know that we are involved in many overhead tasks, such as choosing a function, selecting another object, or issuing a command, while performing our main task. Consider e.g. a computer-aided design system, where the user wants to sculpt a free-form object. In order to perform this task, editing tools not only have to be applied to the model, they also have to be *selected*, or *switched*. Therefore, beside the main task, the user frequently performs actions in which a command is applied to change either the mode of interaction or the application state. Such actions are defined as *system control*, or *application control* [11], and they are always part of an application.

© The Eurographics Association 2003.



delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

Considering complex manipulation tasks in a VE in this context, it becomes clear that appropriate 3D interaction techniques for application control must be integrated into the user interface, in order to fully exploit the potentials of workbench VE systems for direct manual interaction. In contrary to most manipulation tasks, solutions to that problem are far from being obvious. On a 2D desktop system, applications are normally controlled via a WIMP-style interface (Windows, Icons, Menus, and Pointers). It is commonly accepted as the standard interaction method, with the consequence that the related concepts unalteredly appear in many VE applications as well. However, inherently two-dimensional concepts cannot simply be transferred to VEs, since the extra third dimension involves many more degrees of freedoms compared to a 2D environment [11].

In this paper, we focus on the problem of how to perform application control in the context of complex object manipulation tasks in workbench-like systems. The most important action that frequently occurs while a virtual object is being manipulated is *selecting a tool*, including *switching between tools*. In applications such as immersive modelling, a large set of different editing tools can exist, which should be kept ready for instant selection and immediate application to an object.

We propose a new interaction technique, which we name the *ToolFinger*. The ToolFinger is an interaction widget that has been designed specifically to integrate tool selections and tool switches into complicated workflows that are typical for modifying and editing geometry. The ToolFinger supports the quick selection and immediate application of editing tools applied consecutively to objects in workbench-like VE systems. In our context, an item representing a function to perform a specific task on an object is referred to as a *virtual tool*, or shortly, as a *tool*. Usually, a tool is hand held, and transfers hand motion into a change of the state of objects.

Note that the ToolFinger is a software-based solution, due to the rather simple input devices that are common to most VE systems. The ToolFinger merely requires a hand-held input device, whose position and orientation in space is tracked, and which has at least one button. Most suited is a pen-like device, such as the commercially available and widely used stylus, or any similar device. This property makes our technique applicable in most projection-based systems.

The ToolFinger technique integrates two actually separated tasks, namely tool selection, and tool application, into one seamless flow of action that especially benefits object manipulation using several tools. In traditional menu-based solutions, the user would have to interrupt an editing task and break his focus of attention each time he would need a new tool. In contrast to this, the ToolFinger reduces the sequence of interaction steps that are necessary to change the tool significantly. The approach pursued by the ToolFinger is, firstly, that a tool is selected very close to the region of space where it is applied to an object, and secondly, that tool application immediately and seamlessly follows tool selection. Moreover, all this is performed by just one hand, usually the dominant hand, so that the non-dominant hand is free to position and orient the scene appropriately, thus enabling two-handed interaction.

The paper is organized as follows: In the next section, related work on application control in VEs is summarized. Next, we present the ToolFinger technique and describe its design issues. It follows a discussion on further work, and the conclusion.

## 2. Related work

Although the application control components of the user interface crucially influence the usability of VE applications, specifically designed techniques are rare, even for table-like work environments.

Bowman et al. [2] give a categorisation about the existing methods, which include graphical menus, voice commands, gestures, and gestural interaction. For the purpose of application control, including tool selection, a few non-traditional approaches have been proposed.

Coquillart and Grosjean [8] propose a promising approach to application control. They transfer the principle of keyboard hotkeys to the Responsive Workbench, using a cube-shaped widget, the Command & Control Cube or CCC. It consists of a 3D grid of small cubes with which commands are associated. A command can be activated by positioning a selection pointer within the corresponding cube, using a tracked input device. The authors emphasize that this technique allows controlling the application even in "eyes-off" mode, due to its regular structure. The control cube is the first approach that introduces a shortcut paradigm to VE interaction. Compared to the ToolFinger, the CCC is a general approach to application control. Our technique more specifically focuses on object manipulation and editing that occur e.g. in immersive modelling systems, and spatially relates the interaction widget with the edited object.

The virtual palette [4], and a similar approach by Schmalstieg et al. [15], can be used for two-handed application control. The virtual palette resembles a transparent plate with a handle, held by the non-dominant hand. It is used together with a stylus in projection-based environments. With that configuration, items can be selected on the palette using the stylus.

Mine et al. [13] explore innovative, body-centered menus. The menu position and orientation is defined relative to the user in the virtual environment. The authors found that these techniques can significantly enhance user performance, due to the proprioceptive cues, i.e. the person's sense of position and orientation of the body and the arms.

© The Eurographics Association 2003.

Hand-oriented menus [12, 16] are a widely used method for controlling applications. Liang, Green, and Shaw [12, 16] use a ring menu to select tools in an interactive modelling system. The ring menu is represented as a circular object, on which several items are placed. To select a tool, the user can rotate the hand, until the desired icon is activated.

A Pinch Glove-based menu for application control tasks is presented by Bowman and Wingrave [3], and implemented in an immersive virtual environment. The menu texts appear close to each finger, oriented in the finger direction. Touching a finger with the thumb of the same hand, i.e. pinching, causes a selection. The Pinch Glove is a glove device that is equipped with contacts at each fingertip, which are sensible to pinching. This technique therefore is another possible solution to tool selection. However, additional hardware is needed. Furthermore, the necessity of pinching makes it difficult to use a stylus-like input device with the same hand.

Cutler et al. [5] describe how two-handed direct manipulation can be performed on a Responsive Workbench in a natural way. Both hands are instrumented with Pinch Gloves. Tool switches are accomplished explicitly by pinching, or by picking up tools from a toolbox that is located at the front of the display area. In certain situations, tool transitions occur implicitly by reaching in with the second hand, thereby switching from one-handed to two-handed mode.

Forsberg et al. [7] present a modelling framework running on a table-sized display. The interaction paradigm is based on physical props and multimodal input. Transitions between a variety of 2D and 3D interaction techniques are supported. To switch a prop, users put down one prop and pick up another. Transitions between virtual tools are accomplished either by speech recognition or by a drawn gesture.

Most of the techniques summarized here are based on moving the arm or the hand in order to put down or pick up an item or a prop from some location in the environment. Note that the user needs to change the focus of attention each time when reaching towards a new tool.

Apart from these contributions, the main methods of application control and tool selection in VEs are based on 2D widgets that are used in a similar way as within desktop environments. However, techniques related to our approach have been proposed in 2D environments as well.

The Toolglass [1] is a widget that can appear, as a sheet of glass, between an application area and a cursor. The Toolglass area is subdivided into regions to which a set of tools, or a set of alternative selections for an attribute, e.g. colours or contour shapes, is assigned. By positioning the widget relatively to the application so that the selection region overlaps an object, that selection can be applied, by moving the cursor inside. The principle of relating regions of interaction widgets with objects is very similar to our ToolFinger approach. However, the ToolFinger widget incorporates both a set of implicit selection regions and the cursor, which are separated

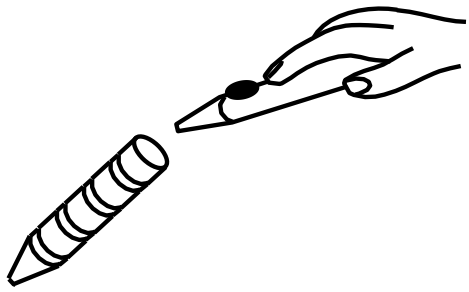

**Figure 1:** *The design of the ToolFinger, shown with the hand and the input device. Thick sections correspond to selectable tools. They are separated by thin sections.*

in the Toolglass approach. Consequently, the ToolFinger is operated with just one hand, in order to have the other hand available for orienting and positioning the model appropriately.

The FlowMenu is a new kind of marking menu [9], for use with a pen device directly on large display devices such as wall-mounted displays. The FlowMenu consists of eight octants placed around a central rest area. Starting from that area, the user enters the fields of the menu with a pen, eventually activating sub-menus, without leaving the display area. Sequences of interactions can be arbitrarily long. They smoothly integrate command selection, text entry and direct manipulation, e.g. touching letters for quickwriting, or crossing octant lines clockwise or counterclockwise to scale an object. As in our approach, the FlowMenu aims at integrating consecutive tool selections and direct manipulation tasks fluidly. It is designed for display surfaces in a 2D environment, whereas our ToolFinger widget directly interacts with objects in a 3D virtual environment.

## 3. The ToolFinger

The key idea of the ToolFinger approach is to subdivide a selection pointer into several sections and interpret an intersection of an object with one of those sections of the pointer as a tool selection. With each section of the ToolFinger, a tool is associated; see Figure 1.

### 3.1. Purpose

Although the interactive complexity of applications running in table-like work environments tends to increase, there is a lack of novel interaction techniques specifically designed for handling complex object manipulation in a VE more efficiently. In these applications, various functions are typically defined for each object class. To mention just a few, most objects can be moved, copied, or deleted. Moreover, the application might support various geometric modelling functions, such as smoothing, deforming, or moving control

© The Eurographics Association 2003.

points. Complex editing tasks are characterized by frequent changes of the current tool, an action that can distract the user from his main work if it is not properly supported. The main purpose of the ToolFinger is to provide an efficient method of dealing with that kind of interaction.

### 3.2. Design

The ToolFinger is an interaction widget shaped like a virtual pointer, or "finger", and formed by connecting several coloured cylindrical sections. There are thin and thick sections. A thick section has a length of about 1.3 cm, whereas a thin section has half the length of a thick section. The diameter of the sections is 1.3 cm. The tip of the ToolFinger has a shape similar to a cone, although its function is equivalent to that of a thick segment. Choosing that kind of design, the ToolFinger resembles a drawing tool, such as a pencil, which indicates its purpose.

Each thick segment of the finger corresponds to a specific manipulation tool, e.g. copy, move, delete, or moving control points. In order to prevent an abrupt change of the tool when moving the ToolFinger along the object, thin segments that have no tool assigned are placed in between two thick segments.

The ToolFinger is connected to the stylus tracker and therefore follows the hand of the user. Normally, on a Responsive Workbench, the hands are located above and in front of the scene. In order to support interacting with objects positioned that way, the ToolFinger can be tilted slightly downwards relative to the stylus axis, see Figure 1. The shown ToolFinger would have a length of about 9.1 cm.

### 3.3. Use

Principally, the way of using the ToolFinger is very similar to the use of the familiar pick tool in a virtual environment. With the pick tool, usually represented by a virtual pointer following the hand, the user points at an object, presses a button, and moves the object to another location. With the ToolFinger, the user selects an object using that section that corresponds to the desired tool, presses the stylus button and performs the action, holding down the button. After that, the button is released, so that the ToolFinger can be applied in the same way again.

In order to select the right tool, the user is supported with a visual feedback mechanism. When a segment of the ToolFinger intersects an object, a short text describing the functionality of the associated tool appears right above the stylus, see Figure 2. It is therefore easily possible to browse through the available functions by moving the ToolFinger across the object, since a tool is only selected when the button is pressed.

Compared to the use of menus or toolboxes, where picking up a tool and applying the tool are separate steps, the

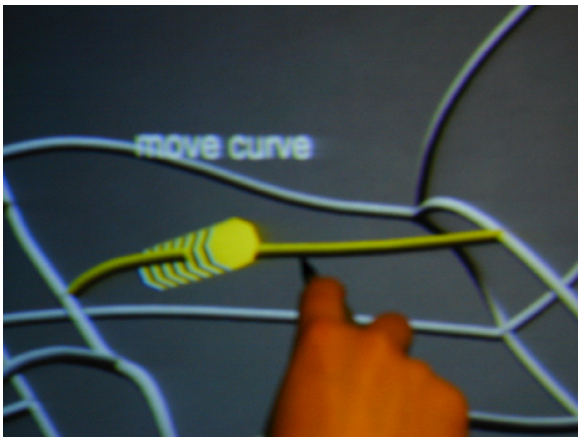


**Figure 2:** *Selecting the move tool*

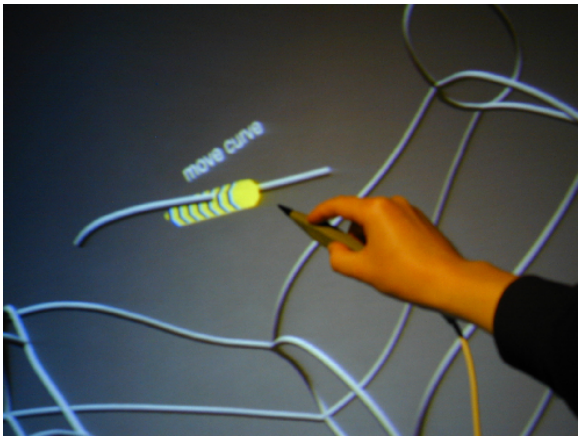


**Figure 3:** *Moving the curve immediately after tool selection*

ToolFinger approach integrates tool selection with tool application. The tool can be applied immediately after touching the object with the associated finger segment, by pressing the button. The use of the ToolFinger is illustrated in Figure 2 and 3. Suppose the user wants to move a curve. Using the ToolFinger technique, just one corresponding movement of the arm is sufficient to perform the whole task: The user grabs the curve with that section of the ToolFinger that is associated with the move tool (Figure 2), presses the button and moves the curve (Figure 3). Releasing the button places the curve at the current location.

### 3.4. Advantages

The integration of an application control task into the flow of action of manipulation tasks is the main contribution of the ToolFinger approach. It implements a fluent transition between selecting a tool and applying it. The ToolFinger technique especially benefits complicated editing tasks that require many subsequent transitions between different tools.

© The Eurographics Association 2003.

Three main characteristics distinguish the ToolFinger approach from traditional menu-based tool selection methods:

- The spatial area where the tool is selected and where it is applied is nearly the same. The hand always acts in vicinity to the object.
- The available manipulation tools are all attached to the hand at the same time.
- The ToolFinger can be completely controlled by the dominant hand alone.

These characteristics benefit performance and user comfort in the following way:

- The user's focus of attention can stay concentrated on the editing task. He can always look at the object that he is manipulating.
- There is no need for the user to interrupt his main task, since he is freed from moving away his arm in order to reach a menu item.
- Two-handed interaction, which is the predominantly used technique on a Responsive Workbench, is much supported by the ToolFinger. Since the non-dominant hand is left out of the tool selection process, it is always free to align the model appropriately.
- There is no need for a separate tool drop technique that would add increased complexity to the kind of editing tasks described, since the ToolFinger is ready for reuse after the button has been released.

Obviously, other tool selection methods have similar advantages, e.g. speech recognition or pinching. However, the use of speech input is often regarded as problematic, e.g. [7] reports frequently misinterpreted spoken commands and ambient noise affecting the system. Concerning the use of Pinch Gloves, most VE applications do require a hand-held tracked input device, which would cause pinching with the same hand to be rather impracticable.

### 3.5. Requirements

ToolFinger interaction requires accurate low-jitter tracking of the input device. The normally used manipulation space in a table-like environment is given by the reach of the arms, which corresponds to the limited working volume of usual electromagnetic tracking systems.

In our Responsive Workbench environment, we mounted the transmitter of the Polhemus Fastrak system at the front of the table, so that the stylus device sensor normally is within roughly 1 m of the transmitter. Within this range, ToolFinger interaction is not affected by jitter, and the accuracy is sufficient even for interacting with detailed geometry.

### 3.6. Limitations

As we have seen, ToolFinger interaction is always related with a virtual object that is being manipulated. The ToolFinger therefore is not generally applicable for all tasks related with function selection or application control; instead, it is a special purpose technique. Consider e.g. object creation, and suppose a user wants to choose the class to create an object from, e.g. a curve, a cube, or a sphere. In such situations, it is not possible to choose the desired tool using the ToolFinger. Other application control tasks might benefit from the ToolFinger approach, but its way of use would not be obvious.

Another situation where the ToolFinger might not be ideal is applying the same tool to several objects. Consider e.g. deletion of objects. Using the ToolFinger would require the selection of the delete tool for each object separately. It would be more comfortable to preselect the delete tool from a menu and just touch each object.

Consequently, there is a need of a combination of different application control techniques, including the ToolFinger, e.g. menus or equivalent approaches providing individual functions together with ToolFingers.

### 3.7. Applications

We developed the ToolFinger in the context of an immersive geometric modelling application, which is described in detail in [17, 18]. The user can sketch simple free-form models from scratch directly on a Responsive Workbench, by drawing spline curves, creating a curve network, and sculpting the resulting surfaces. User interaction with this application frequently requires the selection of editing tools. Separate tool sets are defined for curves and surfaces, according to the following table:

| class | tools |
|---|---|
| curves | *smooth, sharpen, drag, edit curve points, copy, move, mirror, delete* |
| surfaces | *drag, smooth, sharpen, delete* |

Therefore, we defined two ToolFingers, one for curves, consisting of eight segments (see figures), and another one for surfaces, with four segments. The segments (shown in light colour) are separated from each other by thin pieces to avoid sudden transitions between tools assigned to neighbouring segments. Suppose a user wants to elaborate a curve or a surface until he is satisfied with its shape, alternately using the tools *smooth*, *sharpen* and *edit curve points*. The user can easily edit the object by using different segments of the same finger. Tool transitions occur implicitly. Note that traditional menu-based tool switching would require interrupting the workflow each time when a new tool would be needed.

The use of the ToolFinger is not restricted to shape design. It could usefully be applied in 3D visualisation applications

© The Eurographics Association 2003.

as well, e.g. for showing the results of a crash test simulation. The non-dominant hand could position the car model, whereas the ToolFinger, used by the dominant hand, could easily choose different representation modes by pointing on individual parts of the car model with the corresponding section of the ToolFinger.

## 4. Further Work

Further extensions and improvements of the ToolFinger technique are possible.

With the ToolFinger, the combined selection of the object and the tool needs to be accomplished in 3D, although selecting a tool from a set is conceptually 1D. A mechanism that constraints the movement of the ToolFinger to stay in contact with the object while the user is moving it would ease browsing through the set of tools. Then, the intersection of an additional segment of the ToolFinger with the object would allow reactivating free movement.

Since several kinds of objects usually appear in a scene, each associated with a different set of functions, i.e. with a different ToolFinger, there should be a method of how to select the right ToolFinger. A possible solution is automatic selection of the right tool set when the pointer intersects an object, however, we currently switch manually between different ToolFingers.

Another possible improvement is related with visual feedback. If the user was able to remember the location of the section associated with a certain tool, an instant tool application would be possible. Currently, the name of the tool only appears when the user intersects an object. The user could be further supported by colouring or varying the geometry of individual sections of the ToolFinger. In addition to that, sections belonging to tools that are needed very frequently could be made longer or thicker. This would result in a more distinct appearance as well, which would illustrate the possible choices of selection more clearly.

The ToolFinger concept could be extended to better support the direct application of the same tool, i.e. without switching the tool, to several objects, as discussed in section 3.6. For this, the chosen tool could be assigned to the tip of the ToolFinger as the default tool. In this manner, the ToolFinger could be applied just as a usual tool that was picked up from a toolbox.

The size of the set of tools that can be handled by the ToolFinger is obviously limited. A size of about 8–10 tools seems to be appropriate in order not to exceed a reasonable size of the widget and to keep its sections long enough. A solution for supporting the selection from a larger set might be connecting a limited amount of ToolFingers, forming a *ToolHand*. As the name indicates, the fingers of a ToolHand would be spread, just like the fingers of a stretched hand.

An open question related to that is what would be the "optimal" size of a section of the ToolFinger, since the geometry of a section influences the usability of the technique. The size has to be chosen according to the number of tools, the geometry of the scene, the characteristics of the tracking system, and the fine motor skills of a user. Detailed experiments in this field have to be done.

## 5. Conclusion

With the ToolFinger interaction technique for virtual environments, we presented a possible solution to the problem of how to integrate the task of tool selection into the workflow of tool application. The ToolFinger can increase the performance in situations, when several tools need to be used interchangeably, and in quick succession. Therefore, the ToolFinger is an interaction technique highly adapted to complex manipulation tasks that occur in applications such as geometric modelling or interactive visualisation.

Since the ToolFinger was not designed as a general-purpose technique, we also discussed the limitations of our approach, and found that there is a need of a combination of several application control techniques.

Concerning interaction metaphors, note that the ToolFinger concept is *not* based on familiar work situations that occur in the physical world. Instead, the ToolFinger concept makes use of the fact that both the scene and the tools are virtual objects. Note that, in reality, it hardly would be possible to hold more than one tool in hand at the same time, e.g. to hold a screwdriver, a file, and a hammer while working on a piece of material. Metaphors based on the physical world are often used to derive interaction concepts for virtual environments. This can restrict our possibilities when interacting with virtual objects and tools (see also [14] for a discussion on artificial techniques).

As demonstrated by the ToolFinger, application control techniques based on artificial concepts are an alternative to traditional approaches, and could help to deal with the increasing functionality of virtual reality applications.

## Acknowledgements

Many people contributed to this work directly or indirectly.

We would like to thank Ernst Kruijff and Gernot Goebbels for many fruitful discussions about 3D user interface issues. Many thanks to the development group of our virtual reality framework "Avango", and to Bernd Fröhlich, who implemented classes for 3D interaction on the Responsive Workbench, from which classes for the interaction methods of this work have been derived.

We would also like to thank Martin Göbel and Martin Reiser who support this project. Many thanks to Hans-Peter Seidel for his valuable comments on the modelling system that this work is based on.

© The Eurographics Association 2003.

Klaus-Günter Rautenberg assisted us with taking pictures of our method on the Responsive Workbench.

This work is partly funded by the Deutsche Forschungsgemeinschaft (DFG) under grant number GO 856/2-2.

**References**

1. E.A. Bier, M.C. Stone, K. Pier, W. Buxton, and T.D. DeRose. Toolglass and Magic Lenses: The See-Through Interface. In *SIGGRAPH '93 Proceedings*, pages 73–80, 1993.
2. D. Bowman, E. Kruijff, J. LaViola Jr., M. Mine, and I. Poupyrev. 3D User Interface Design: Fundamental Techniques, Theory, and Practice. In *SIGGRAPH 2000 Course Notes*, July 2000.
3. D. Bowman and C. Wingrave. Design and Evaluation of Menu Systems for Immersive Virtual Environments. In *Proceedings of IEEE Virtual Reality 2001*, pages 149–156, Yokohama, Japan, March 2001.
4. S. Coquillart and G. Wesche. The Virtual Palette and the Virtual Remote Control Panel: A Device and an Interaction Paradigm for the Responsive Workbench. In *IEEE VR 99 Proceedings*, pages 213–216, Houston, Texas, USA, March 1999.
5. L. Cutler, B. Fröhlich, and P. Hanrahan. Two-handed Direct Manipulation on the Responsive Workbench. In *Proc. Symposium on Interactive 3D Graphics*, 1997.
6. M. Czernuszenko, D. Pape, D. Sandin, T. DeFanti, G. Dawe, and M. Brown. The ImmersaDesk and Infinity Wall Projection-Based Virtual Reality Displays. *Computer Graphics*, 31(2):46–49, 1997.
7. A.S. Forsberg, J.J. LaViola Jr., and R.C. Zeleznik. ErgoDesk: A Framework for Two and Three Dimensional Interaction at the ActiveDesk. In *Proceedings of the Second International Immersive Projection Technology Workshop*, pages 11–12, Ames, Iowa, USA, May 1998.
8. J. Grosjean and S. Coquillart. Command & Control Cube: a Shortcut Paradigm for Virtual Environments. In *Immersive Projection Technology and Virtual Environments 2001 Proceedings*, pages 1–12, May 2001.
9. F. Guimbretiére and T. Winograd. FlowMenu: Combining Command, Text, and Data Entry. In *UIST 2000 Proceedings*, pages 213–216, 2000.
10. W. Krüger, C. A. Bohn, B. Fröhlich, H. Schüth, W. Strauss, and G. Wesche. The Responsive Workbench: A virtual work environment. *IEEE Computer*, 28(7):42–48, 1995.
11. E. Kruijff. System Control. In D. Bowman, E. Kruijff, J. LaViola Jr., M. Mine, and I. Poupyrev, editors, *3D User Interface Design: Fundamental Techniques, Theory, and Practice. SIGGRAPH 2000 Course Notes*, pages 147–165, July 2000.
12. J. Liang and M. Green. JDCAD: A Highly Interactive 3D Modeling System. *Computers and Graphics*, 18(4):499–506, 1994.
13. M. Mine, F. Brooks, and C. Sequin. Moving objects in space: exploiting proprioception in virtual-environment interaction. In *Proceedings of SIGGRAPH '97*, pages 19–26, 1997.
14. I. Poupyrev. The Art of Designing 3D Interfaces. In D. Bowman, E. Kruijff, J. LaViola Jr. M. Mine, and I. Poupyrev, editors, *3D User Interface Design: Fundamental Techniques, Theory, and Practice. SIGGRAPH 2000 Course Notes*, pages 177–194, July 2000.
15. D. Schmalstieg, L.M. Encarnacao, and Z. Szalavari. Using Transparent Props for Interaction with the Virtual Table. In *Proceedings of the 1999 ACM Symposium on Interactive 3D Graphics*, pages 147–154, 1999.
16. C. Shaw and M. Green. Two-Handed Polygonal Surface Design. In *Proceedings of ACM UIST '94*, pages 205–212, 1994.
17. G. Wesche and M. Droske. Conceptual Free-Form Styling on the Responsive Workbench. In *VRST 2000 Proceedings*, pages 83–91, Seoul, Korea, October 2000.
18. G. Wesche and H.-P. Seidel. FreeDrawer–A Free-Form Sketching System on the Responsive Workbench. In *VRST 2001 Proceedings*, pages 167–174, Banff, Alberta, Canada, November 2001.

© The Eurographics Association 2003.