Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces

Michel Beaudouin-Lafon
Dept of Computer Science
University of Aarhus
Aabogade 34
DK-8200 Aarhus N - Denmark
mbl@daimi.au.dk

ABSTRACT

This article introduces a new interaction model called Instrumental Interaction that extends and generalizes the principles of direct manipulation. It covers existing interaction styles, including traditional WIMP interfaces, as well as new interaction styles such as two-handed input and augmented reality. It defines a design space for new interaction techniques and a set of properties for comparing them. Instrumental Interaction describes graphical user interfaces in terms of *domain objects* and *interaction instruments*. Interaction between users and domain objects is mediated by interaction instruments, similar to the tools and instruments we use in the real world to interact with physical objects. The article presents the model, applies it to describe and compare a number of interaction techniques, and shows how it was used to create a new interface for searching and replacing text.

Keywords

Interaction model, WIMP interfaces, direct manipulation, post-WIMP interfaces, instrumental interaction

INTRODUCTION

In the early eighties, the Xerox Star user interface [27] and the principles of direct manipulation [26] led to a powerful graphical user interface model, referred to as WIMP (Windows, Icons, Menus and Pointing). WIMP interfaces revolutionized computing, making computers accessible to a broad audience for a variety of applications.

In the last decade, HCI researchers have introduced numerous new interaction techniques, such as toolglasses [5] and zoomable user interfaces [3]. Although some have been shown to be more efficient than traditional techniques, e.g., marking menus [19], few have been incorporated into commercial systems. A likely reason is that integrating new interaction techniques into an interface is challenging for both designers and developers. Designers find it faster and easier to stick with a small set of well-understood techniques. Similarly, developers find it more efficient to take advantage of the extensive support for WIMP interaction provided by current development tools.

CHI '2000 The Hague, Amsterdam

The leap from WIMP to newer "post-WIMP" graphical interfaces, which take advantage of novel interaction techniques, requires both new interaction models and corresponding tools to facilitate development. This paper focuses on the first issue by introducing a new interaction model, called *Instrumental Interaction,* that extends and generalizes the principles of direct manipulation to also encompass a wide range of graphical interaction techniques. The Instrumental Interaction model has the following goals:

- cover the state-of-the-art in graphical interaction techniques;
- provide qualitative and quantitative ways to compare interaction techniques, to give designers the basis for an informed choice when selecting a given technique to address a particular interface problem;
- define a design space in which unexplored areas can be identified and lead to new interaction techniques; and
- open the way to a new generation of user interface development tools that make it easy to integrate the latest interaction techniques in interactive applications.

After a review of related work, this paper analyzes the limits of current WIMP interfaces. The Instrumental Interaction model is introduced and applied to several existing interaction techniques as well as to the design of a new interface for searching and replacing text. Finally the paper concludes with suggestions for future work.

RELATED WORK

In this paper, an *interaction model* is defined as follows:

> An interaction model is a set of principles, rules and properties that guide the design of an interface. It describes how to combine interaction techniques in a meaningful and consistent way and defines the "look and feel" of the interaction from the user's perspective. Properties of the interaction model can be used to evaluate specific interaction designs.

Direct Manipulation [26] is a generic interaction model, while style guides, e.g., Apple's guidelines [2], describe more precise and specific models. Took introduced a model called Surface Interaction [29] and Holland & Oppenheim a model called Direct Combination [15].

An interaction model differs from the *architectural model* of an interface, which describes the functional elements in the implementation of the interface and their relationships (see

[12] for a review). User interface development environments have generated a variety of *implementation* models for developing interfaces (see [24] for a review), e.g. the widget model of the X/Motif toolkit [14] or Garnet's Interactors model [23]. MVC [18] is a well-known model that was created to support the Xerox Star user interface and has influenced many other architectural and implementation models. Whereas architectural models are aimed at interface *development*, an interaction model is aimed at interface *design*.

The *model-based* approach and its associated tools [28] help bridge the gap between interaction and architectural models by offering a higher-level approach to the design of interactive systems.

Device-level models such as logical input devices [11] or Card et al.'s taxonomy [7] operate at a lower level of abstraction than interaction models. Understanding the role of the physical devices in interaction tasks is a critical component of the definition of the Instrumental Interaction model.

At the theoretical level, Activity Theory [6] provides a relevant framework for analyzing interaction as a mediation process between users and objects of interest.

Finally, Instrumental Interaction is grounded in the large (and growing) number of graphical interaction techniques that have been developed in recent years, some of which are referenced in the rest of this article.

FROM WIMP TO POST-WIMP INTERFACES

The WIMP interaction model can be outlined as follows:

- application objects are displayed in document windows;
- objects can be selected and sometimes dragged and dropped between different windows; and
- commands are invoked through menus or toolbars, often bringing up a dialog box that must be filled in before the command's effect on the object is visible.

This section uses Shneiderman's principles of direct manipulation [26] to analyze WIMP interfaces:

1. Continuous Representation of objects of interest

Objects of interest are central to direct manipulation. They are the objects that the user is interested in to achieve a given task, such as the text and drawings of a document or the formulae and values in a spreadsheet. Principle 1 asserts that objects of interest should be present at all times. Since objects of interest are often larger than the screen or window in which they are displayed, WIMP interfaces makes them *accessible* at all times through scrolling, panning or zooming. This accessibility is hindered by the growing number of interface objects that are *not* objects of interest such as toolbars, floating palettes and menu bars. These use increasing amounts of screen real-estate, forcing the user to shrink the windows displaying objects of interest. Dialog boxes also often occlude significant parts of the screen, making the rest of the interface inaccessible to the user.

Finally, there are more objects of interest than meet the eye: in many applications users must manipulate secondary objects to achieve their tasks, such as style sheets in Microsoft Word, graphical layers in Adobe Photoshop or Deneba Canvas, or paint brushes in MetaCreations Painter. Once the user is familiar with the application, these objects become part of his or her mental model and may acquire the status of object of interest. Unfortunately, these are rarely implemented as first-class objects. Thus, for example, Word's styles are editable only via transient dialog boxes that must be closed before returning to the text editing task.

2. Physical actions on objects vs. complex syntax

Most computers have only a mouse and keyboard as input devices limiting the set of user actions to: typing text or "special" keys (e.g. function keys, keyboard shortcuts and modifiers), pointing, clicking, and dragging. Given the mismatch between this small vocabulary of actions and the large vocabulary of commands, WIMP interfaces must rely on additional interface elements, usually menus and dialog boxes, to specify commands. The typical sequence of actions to carry out a command is:

- select the objet of interest by clicking it;
- select a command from a menu or keyboard shortcut;
- fill in the fields of a dialog box; and
- click the OK button to see the result.

This is conceptually no different from typing a command in a command-line interface: The user must type a command name, file name (the object of interest), arguments (the fields in the dialog box) and return key (the OK button). In both cases the syntax is complex and cannot be considered direct manipulation of the objects of interest. In fact, WIMP interfaces directly violate principle 2 and often use *indirect* manipulation of the objects of interest, through (direct) manipulation of interface elements such as menus and dialog boxes.

3. Fast, incremental and reversible operations with an immediately-apparent effect on the objects of interest

The heavy graphical syntax imposed on the user results in commands that are neither fast nor incremental. Specifying a command is not fast because of the amount of time used for non-semantic actions such as displacing windows and flipping through tabs in a tabbed dialog. Inputting parameter values for a command is often inefficient because of the small set of interactors, such as when numeric values are entered as text. Finally, the specification is not incremental: users must explicitly commit to a command that uses a dialog box before seeing the result. If the result does not match the user's expectations, the whole cycle of command activation must be started over again. This is especially cumbersome when trial-and-error is an integral part of the task, as when a graphics designer selects a font size: specifying the point size numerically is annoying when the goal is to see the visual result on the page.

4. Layered or spiral approach to learning

The small number of interaction techniques used by WIMP interfaces makes it easy to learn the basics of any new application. However, interaction shortcuts, such as combining keyboard modifiers with mouse buttons to activate the frequent commands, are concealed and inconsistent across applications and make the transition from novice to power user more difficult.

Towards a new interaction model

Some commercial applications, especially those dedicated to creative tasks such as painting, graphic design or music, extend the basic WIMP model to address some of the shortcomings identified above. For example, some painting programs make brushes first class objects that can be edited and saved into files. Some text editors have inspector windows that display the state of the current selection and update it when the user enters relevant values. Techniques such as the HotBox [21] were designed to access larger numbers of commands.

These new interaction techniques illustrate the transition from WIMP to post-WIMP interaction: Windows are not used in zoomable interfaces, icons and text are replaced by richer representations in interactive visualization, menus are complemented by more powerful interaction widgets such as toolglasses, pointing and dragging are superseded by bimanual input and gesture input. Designing Post-WIMP interfaces that are more faithful to the principles of direct manipulation and that take advantage of novel interaction techniques requires new interaction models. To guide interface designers, these models should be:

- *descriptive*, incorporating both existing and new applications;
- *comparative*, providing metrics for comparing alternative designs (as opposed to *prescriptive*, deciding a priori what is good and what is bad); and
- *generative*, facilitating creation of new interaction techniques.

INSTRUMENTAL INTERACTION

As shown in the previous analysis, WIMP interfaces do not follow the principles of direct manipulation. Instead, they introduce interface elements such as menus, dialog boxes and scrollbars that act as *mediators* between users and the objects of interest. Users have a (limited) sense of engagement, as advocated by direct manipulation, because they manipulate these intermediate objects directly. This matches our experiences in the physical world: We rarely fingerpaint, but often use pens and pencils to write. We cook with pots and pans, hang pictures with hammers and power drills, open doors with handles and turn off lights with switches. Our interaction with the physical world is governed by our use of *tools*. Direct manipulation of *physical* objects of interest occurs when we bring them into our current context of operation, before we manipulate them with the appropriate tools, usually with two hands [13]. The Instrumental Interaction model is based on how we naturally use tools (or instruments) to manipulate objects of interest in the physical world. Objects of interest are called *domain objects*, and are manipulated with computer artifacts called *interaction instruments*.

Domain objects

In computer systems, applications operate on data that represent phenomena or objects. For computer users, this data is the primary focus of their actions. For example, when creating a text document, the focus of the user is on the text of the document. Everything else on the screen is there to support the user's task of editing the text document.

Domain objects form the set of potential objects of interest for the user of a given application. Domain objects have *attributes* that describe their characteristics. Attributes can be simple values or more complex objects. For example, in a 3D modeller, the position and size of a sphere are simple values (integer or real numbers), while the material of the sphere is a complex entity (color, texture, transparency, etc.). The user may shift the object of interest, concentrating on the material as the focus of the interaction. Similarly, text styles that describe the formatting attributes of text also may also obtain the status of objects of interest. Materials and styles are therefore also domain objects in their respective interfaces.

In summary, domain objects form the basis of the interaction as well as its purpose: Users operate on domain objects by editing their attributes. They also manipulate them as a whole, e.g. to create, move and delete them.

Interaction instruments

An *interaction instrument* is a mediator or two-way transducer between the user and domain objects. The user acts on the instrument, which transforms the user's *actions* into *commands* affecting relevant target domain objects. Instruments have *reactions* enabling users to control their actions on the instrument, and provide *feedback* as the command is carried out on target objects (Figure 1).

A scrollbar is a good example of an interaction instrument. It operates on a whole document by changing the part that is currently visible. When the user clicks on one of the arrows of the scrollbar, the scrollbar sends the document a scrolling command. Note that the transduction here consists of sending scrolling commands as long as the user presses the arrow. The reaction of the scrollbar consists of highlighting the arrow being pressed. The feedback consists of updating the thumb to reflect the new position of the document. In addition, the object also responds to the instrument by updating its view in the window.

Another example is an instrument that creates rectangles in a drawing editor. As the user clicks and drags the mouse, the instrument provides a reaction in the form of a rubber-band rectangle. When the user releases the button, the creation operation is actually carried out and a new domain object is created. The feedback of this operation consists in displaying the new object.

An instrument decomposes interaction into two layers: the interaction between the user and the instrument, defined as the physical *action* of the user on the instrument and the *reaction* of the instrument and the interaction between the instrument and the domain object, defined as the *command* sent to the object and the *response* of the object, which the instrument may transform into *feedback* to the user. The instrument is composed of a physical part, the input device, and a logical part, the representation of the instrument in software and on the screen.

Activating instruments

At any one time, an interface provides a potentially large number of instruments. However the user can manipulate only a few of them at the same time, usually only one, because of the limited number of input devices. In the most

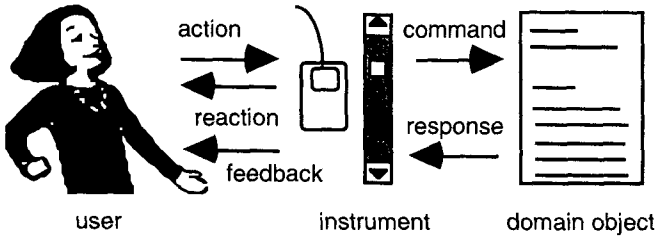

Figure 1: Interaction instrument mediating the interaction between a user and a domain object

common case of a keyboard and mouse, a single input device (the mouse) must be multiplexed between a potentially large number of instruments, i.e. a single physical part may be associated with different logical parts.

An instrument is said to be *activated* when it is under the user's control, i.e. when the physical part has been associated with the logical part. In the case of the scrollbar, the user activates the instrument by pointing at it and it remains active as long as the pointer is within the scrollbar. When creating a rectangle, the user activates the instrument by clicking a button in a tool palette and it remains active until another instrument is activated.

These two types of activation are quite different. The activation of the scrollbar is *spatial* because it is caused by moving the mouse (and cursor) inside the area of the scrollbar. The activation of the rectangle creation instrument is *temporal* because it is caused by a former action and remains in effect until the activation of another instrument. (This is traditionally called a mode). Each type of activation has an associated cost: Spatial activation requires the instrument to be visible on the screen, taking up screen real-estate and requiring the user to point at it and potentially dividing the user's attention. Temporal activation requires an explicit action to trigger the activation, making it slower and less direct.

Interface designers often face a design trade-off between temporal and spatial multiplexing of instruments because the activation costs become significant when the user must frequently change instruments. Using extra input devices can reduce these costs. For example, the thumbwheel on Microsoft's Intellimouse is a scrolling instrument that is always active. An extreme example is an audio mixing console, which may contain several hundred potentiometers and switches, each corresponding to a single function. This permits very fast access to all functions, which is crucial for sound engineers working in real-time and cannot afford the cost of activating each function indirectly. A large design space lies between a single mouse and hundreds of potentiometers, posing design challenges to maximally exploit physical devices and reduce activation costs.

Reification and Meta-instruments

Reification is a process for turning concepts into objects. In user interfaces, the resulting objects can be represented explicitly on the screen and operated upon. For example, a style in a text editor is the reification of a collection of text attributes; the notion of material in a 3D modeller is the reification of a set of rendering properties. This type of reification generates new domain objects such as styles and materials that complement the "primary" domain objects of the application domain.

Instrumental Interaction introduces a second type of reification: an interaction instrument is the reification of one or more commands. For example, a scrollbar is the reification of the command that scrolls a document. This link between the traditional notion of command and the notion of instrument makes it easy to analyze existing interfaces with the Instrumental Interaction model. It is also a useful guideline to identify instruments when designing a new interface. In the last part of this paper, this rule is used to reify the traditional search-and-replace command of a text editor into a search instrument.

The result of this reification rule is that instruments are themselves potential objects of interest. This is indeed the case in real life, when the focus of attention shifts from the object being manipulated to the tool used to manipulate it. For example a pencil is a writing instrument and the domain object is the text being written. When the lead breaks, the focus shifts to a new instrument, a pencil sharpener, which operates on the shifted domain object, the pencil lead. The focus may even shift to the pen sharpener, if we need a screwdriver to fix it. Such "meta-instruments" (instruments that operate on instruments) are not only useful for "fixing" instruments, but can also be used to organize instruments in the workspace, e.g. a toolbox, or to tailor instruments to particular tasks, e.g. turning a power-drill into a power-saw. In graphical user interfaces, common meta-instruments include menus and tool palettes used to select commands and tools, i.e. to activate instruments.

Properties of Instruments

An important role of an interaction model is to provide properties to evaluate and compare alternative designs. This can help interface designers who face difficult choices when selecting the interaction techniques for a particular application. The goal of defining properties of instruments is not to decide which instruments are good and which are bad, but to evaluate them so that designers can make an informed choice and so that researchers can identify and explore areas of the design space that are not mapped by existing instruments.

The literature on user interface evaluation techniques is considerable. Here, we use a particular type of evaluation based on properties. This is a common approach in software engineering and has also proved valid and useful for evaluating interactive systems [12]. The rest of this section introduces three properties of interaction instruments.

Degree of indirection

The degree of indirection is a 2D measure of the spatial and temporal offsets generated by an instrument. The *spatial offset* is the distance on the screen between the logical part of the instrument and the object it operates on. Some instruments, such as the selection handles used in graphical editors, have a very small spatial offset since they are next to or on top of the object they control. Other instruments, such as dialog boxes, can be arbitrarily far away from the object they operate on and therefore have a large spatial offset. A large spatial offset is not necessarily undesirable.

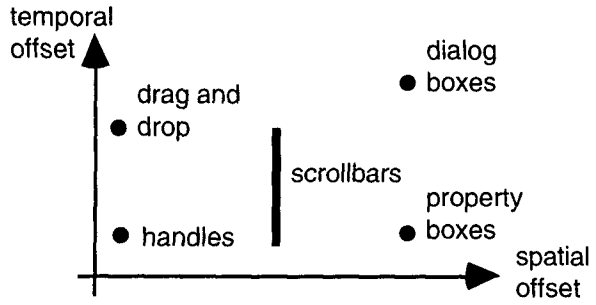

Figure 2: Degree of indirection

For example, placing a light switch far from the light bulb it controls makes it easier to turn on the light. Similar examples can be found in user interfaces.

The *temporal offset* is the time difference between the physical action on the instrument and the response of the object. In some cases, the object responds to the user's action in real-time. For example, clicking an arrow in a scrollbar scrolls the document while the mouse button is depressed. In other cases, the object responds to the user's action only when the action reaches closure. For example, the arguments specified in a dialog box are taken into account only when the OK or Apply button is activated. In general, short temporal offsets are desirable because they exploit the human perception-action loop and give a sense of causality [22].

Figure 2 shows the degree of indirection of various WIMP instruments on a 2D chart. Scrollbars occupy a range in the diagram. For example, some scrollbars provide immediate response when the thumb is moved while others only scroll the document when the mouse button is released. The figure shows that the degree of indirection describes a continuum between direct manipulation (lower-left corner) and indirect manipulation (upper-right corner).

Degree of integration

The degree of integration measures the ratio between the number of degrees of freedom (DOF) provided by the logical part of the instrument and the number of DOFs captured by the input device. This term comes from the notion of *integral tasks* [17]: some tasks are performed more efficiently when the various DOFs are controlled simultaneously with a single device. A scrollbar is a 1D instrument controlled by a 2D mouse, therefore its degree of integration is 1/2. The degree of integration can be larger than 1: controlling 3 rotation angles with a 2D mouse [16] has a degree of integration of 3/2. This property can be used to compare instruments that perform similar operations. For example, panning over a document can be achieved with two scrollbars or a 2D panner. The latter has a degree of integration of 1 and is therefore more efficient than two scrollbars, which have a degree of integration of 1/2 and incur additional activation costs.

Degree of compatibility

The degree of compatibility measures the similarity between the physical actions of the users on the instrument and the response of the object. Dragging an object has a high degree of compatibility since the object follows the movements of the mouse. Scrolling with a scrollbar has a low degree of compatibility because moving the thumb downwards moves the document upwards. Using text input fields to specify numerical values in a dialog box, e.g. the point size of a font, has a very low degree of compatibility because the input data type is different from the output data type. Similarly, specifying the margin in a text document by entering a number in a text field has a lower degree of compatibility than dragging a tab in a ruler.

	Indirection	Integration	Compatibility
Menus	--	+/-	--
Toolbars	-	+	+
Dialog boxes	--	+/-	-
Property boxes	+	+/-	-
Handles	++	++	++
Window titles	++	+	+
Scrollbars	+	-	-
Keyboard shortcuts	+	+	-
Drag & drop	++	++	++

Table 1: Comparing WIMP interaction techniques

APPLYING THE MODEL

This section uses the Instrumental Interaction model to analyze existing interaction techniques, both from WIMP interfaces and from more recent research. It demonstrates the descriptive power of the model. The generative power of the model is then illustrated by the design of a new instrument for searching and replacing text.

Analyzing WIMP Interfaces

The primary components of WIMP interfaces can be easily mapped to instruments and compared (Table 1):

Menus and *toolbars* are meta-instruments used to select the command or tool to activate. This use of meta-instruments slows down interaction and generates shifts of attention between the object of interest, the meta-instrument and the instrument. Contextual menus have a small spatial offset and are therefore more efficient than toolbars and menu bars. Toolbars, which can be moved next to their context of use, have a better spatial offset than menu bars.

Dialog boxes are used for complex commands. They have a high degree of spatial and temporal indirection. They often use a small set of standard interactors such as text fields for numeric values, resulting in a low degree of compatibility.

Inspectors and *property boxes* are an alternative to dialog boxes that have a lower degree of temporal indirection. Since they can stay open, they can be activated with pointing (positional activation) rather than selection in a menu (temporal activation).

Handles are used for graphical editing and provide a very direct interaction: low degree of indirection, high degree of compatibility and good degree of integration.

Window titles and borders are instruments activated positionally to manipulate the window (move, resize, iconify, zoom, close). *Scrollbars* control the content of the window. Because of their low degree of integration, they are not optimal, especially for panning documents in 2D. Also, their spatial offset generates a division of attention,

especially since they are activated positionally: the user must be sure to point at the right part of the scrollbar.

Keyboard shortcuts and *accelerator keys* are meta-instruments, used to quickly switch between instruments and save the activation costs of menus and toolbars. Some accelerator keys affect the way the current instrument works. For example, on the Macintosh, the Shift key constrains the move tool to horizontal and vertical moves and the resize tool to maintain the current aspect ratio.

Drag and drop is a generic instrument for transferring or copying information. Compared to traditional cut/copy/paste commands that use a hidden clipboard, it has a smaller degree of indirection. There is no spatial offset because the objects are manipulated directly and the temporal offset is low because there is feedback about potential drop-zones as the user drags the object.

Over the past few years, interaction techniques such as inspectors, property boxes, drag and drop and contextual menus have become more common in commercial applications. The above analysis explains why these techniques are more efficient than their WIMP counterparts, demonstrating a useful contribution of the Instrumental Interaction model.

Analyzing Post-WIMP Interaction Techniques

Table 2 summarizes the comparison of several post-WIMP interaction techniques. Interactive visualization helps users explore large quantities of visual data and make sense of it through filtering and displaying it to exhibit patterns [8]. These systems use two categories of instruments:

- navigation instruments specify which part of the data to visualize and how; and
- filtering instruments specify queries and display results.

A key aspect of these systems is a strong coupling between user actions and system response. In other words, these instruments must have a small temporal offset. For example, in the Information Visualizer [9], the instruments used to control Cone Trees and Perspective Walls provide immediate responses and use smooth animations to display changes in visualization parameters. In Dynamic Queries [1], double sliders are used to specify the range of query parameters; any change in a slider updates the display of filtered data.

Both navigation and filtering are usually multi-dimensional tasks: the user wants to control several dimensions simultaneously to navigate along arbitrary trajectories. This calls for the ability to manipulate several instruments simultaneously (which requires additional input devices) and/or for instruments with a high degree of integration. Current systems do not address this well. For example, Dynamic Queries permit only one side of a slider to be manipulated at a time, forcing the user to navigate along rectangular trajectories in the parameter space.

Zoomable user interfaces such as Pad++ [3] are based on the display of an infinite flat surface that can be viewed at any resolution. Exploring this surface requires navigation instruments to pan and zoom until the desired objects are in sight. Pad++ navigation instruments are activated by mouse buttons or modifier keys. This temporal activation is fast

	Indirection	Integration	Compatibility
Dynamic Queries	+	--	-
Pad++ navigation	++	+	+
Droppable tools	+	++	+
Toolglasses	++	++	++
Graspable interfaces	++	++	++

Table 2: Comparing post-WIMP interaction techniques

and provides access to navigation anywhere on the surface, unlike a scrollbar which requires positional activation. It also has high degrees of compatibility and integration. Editing the objects on the surface has led to Dropable Tools [4]: tools can be dropped anywhere on the surface and grabbed later. Activating these instruments is more direct than with a traditional toolbar because it does not involve a meta-instrument and the associated switch of attention.

A number of recent interaction techniques rely on new or additional input devices. This reduces activation costs by allowing several instruments to be active simultaneously. For example, the thumb wheel of the Intellimouse is always attached to a scrolling instrument. ToolGlasses [5] are semi-transparent palettes operated with a track-ball in the left (or non-dominant) hand. The right hand is used to click through the palette onto a domain object, therefore specifying both the action to perform and the object to operate on. Here the toolglass is a meta-instrument under the control of the left hand, while the instruments it contains are activated by the right hand. In the TTT prototype [20], a combination of three instruments can be active simultaneously: the toolglass itself, an instrument in the toolglass and a navigation instrument to pan and zoom the drawing surface. This makes it possible, for example, to pan and zoom while creating an object. The design exploits the trackball and mouse input devices to minimize activation costs, to reduce the degree of indirection and to increase the degree of integration.

Graspable interfaces [10] use physical objects as input devices to manipulate virtual objects. In effect, they transfer most of the characteristics usually found in the logical part of the instrument into the physical part. This approach was pioneered by Augmented Reality [30], which explores ways to reconcile the physical and computer world by embedding computational facilities into physical objects. Here, the domain objects, in addition to the instruments, have a strong physical component. This increases the degrees of compatibility and integration since interaction occurs in the real world.

Designing a Text Search Instrument

In most text editors, searching and replacing text uses a dialog box where the user specifies the string to be searched and the string to replace it with. The operation is controlled with buttons to find the next or previous occurrence and replace it or not. Undoing the command usually means restoring the search string everywhere it has been replaced. This results in a sequential form of interaction where the system prompts the user and forces him or her to decide what to do with each occurrence, generating a very large temporal offset.

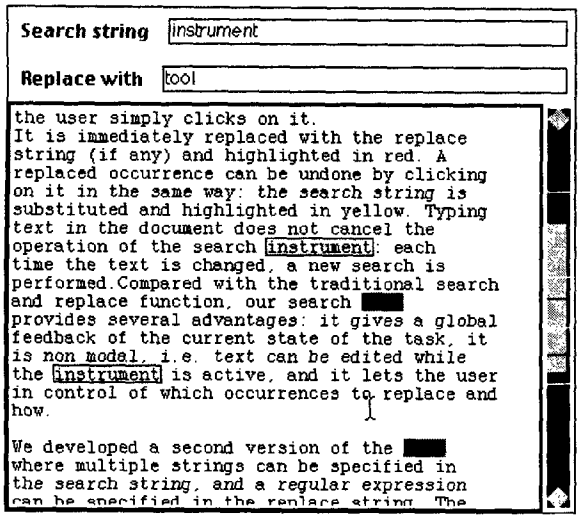

Figure 3: Search and replace instrument

An instrumental approach to search and replace has led to the following design, implemented in a Tcl/Tk prototype (Figure 3). The top part of the window is the logical part of the instrument, used to specify the search and replace strings. No buttons are necessary: as the user types a search string, the occurrences highlight in yellow both in the text window and in the scrollbar. To replace an occurrence, the user simply clicks on it, immediately replacing it with the replace string (if any) highlighted in red. Editing the replace string changes all the replaced occurrences. A replaced occurrence can be undone by clicking on it in the same way: the search string is substituted and highlighted in yellow. Typing text in the document does not cancel the operation of the search instrument: each time the text is changed, a new search is performed.

The scrollbar was modified to facilitate browsing. First, it includes tick marks representing the occurrences and giving an overview of the search. Second the arrow buttons were changed so that clicking on an arrow scrolls the document at a variable speed according to the distance between the cursor and the arrow: clicking on the up arrow scrolls the document slowly downwards (as in traditional scrollbars); moving the cursor up speeds up scrolling; moving the cursor down slows it down; moving the cursor further down, scrolling stops, then reverts. This reduces the division of attention that occurs when operating the various parts of a scrollbar while focusing on the document. It is similar in effect to the thumbwheel described earlier but does not require a separate input device.

The instrument provides feedback about the current state of the search/replace operation by highlighting *all* the occurrences in the text, as in the Document Lens [25], and in the scrollbar. This design results in a low degree of indirection: The spatial offset is reduced by getting rid of the traditional *Next*, *Previous* and *Replace* buttons; The temporal offset is reduced by displaying the occurrences as the user types the search string, by showing the effect of replacing an occurrence immediately and by allowing the user to undo any change, not necessarily in the order they were made.

The design of the scrollbar improves the degree of integration since it uses the vertical position of the mouse to control the speed and direction of scrolling (1 output DOF / 2 input DOFs) while traditional scrollbars only use the fact that the user clicked in the arrow (0 output DOF / 2 input DOFs). This design also improves the degree of compatibility since the mouse now works as a joystick. In effect, the scrollbar thumb and arrows are functionally equivalent: the thumb provides positional control while the arrows provide rate control of the visible part of the document.

Two variants of the search instrument were also developed. The first variant allows several search instruments to be active simultaneously, each independent of the others and using a different pair of colors to highlight occurrences. In the second variant, multiple strings can be specified in the search string and a regular expression can be specified in the replace string. The instrument highlights all the occurrences of all search strings at once. This variant was used to build the index of a book: a list of words to index was entered as a set of search strings. The replace string added the proper markup to include the occurrence in the index. Indexing the book became simply a matter of picking which occurrences to include in the index, taking advantage of the display of all occurrences to avoid putting in the index occurrences of the same word that are close together in the text. At any time it was possible to change the list of words in the index, the content of the text and the occurrences to index.

CONCLUSION AND FUTURE WORK

This article has introduced the Instrumental Interaction model, which generalizes and operationalizes Direct Manipulation. The model has been used to analyze WIMP interfaces as well as more recent interaction techniques and to design a new interface for searching and replacing text. This demonstrates the descriptive, comparative and generative power of the Instrumental Interaction model.

We are currently testing the model as we design a new graphical editor for Colored Petri Nets. The model is used both as a design guide and an evaluation tool to integrate existing interaction techniques and create new ones.

However further work is needed to develop the model in more detail and assess its limits. This requires a more thorough analysis of graphical interfaces and interaction techniques, the definition and evaluation of new properties, a taxonomy of interaction instruments, and an exploration of the design space defined by the model. Design principles are also needed to help designers create instruments and integrate them effectively into a user interface.

The other important area for future work is to make Instrumental Interaction useful not only to user interface designers but also to user interface developers by developing a user interface toolkit based on the model. This would support the adoption of novel interaction techniques in a wide range of applications and allow a shift from WIMP to post-WIMP interfaces.

ACKNOWLEDGMENTS

Many thanks to Wendy Mackay for numerous discussions about the model and its implications and for her help with

the English. Thanks to Peter Bøgh Andersen, Susanne Bødker, Yves Guiard and Austin Henderson for discussions and feedback on earlier versions of this paper.

REFERENCES

1. Ahlberg, C., Williamson, C., Shneiderman, B. Dynamic Queries for Information Exploration: An Implementation and Evaluation. In *Proc. ACM Human Factors in Computing Systems, CHI'92*, ACM Press, p.619-626, 1992.
2. Apple Computer. *Macintosh Human Interface Guidelines*, Addison-Wesley, 1992.
3. Bederson, B. and Hollan, J. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. In *Proc. ACM Symposium on User Interface Software and Technology, UIST'94*, ACM Press, p.17-26, 1994
4. Bederson, B., Hollan, J., Druin, A., Stewart, J., Rogers, D., Proft, D. Local Tools : an Alternative to Tool Palettes. In *Proc. ACM Symposium on User Interface Software and Technology, UIST'94*, ACM Press, p.169-170, 1994.
5. Bier, E., Stone, M., Pier, K., Buxton, W., De Rose, T. Toolglass and Magic Lenses : the See-Through Interface. In *Proc. ACM SIGGRAPH*, p.73-80, 1993.
6. Bødker, S. *Through the Interface. A Human Activity Approach to User Interface Design.* Lawrence Erlbaum Associates, 1991.
7. Card, S.K., Mackinlay, J.D., Robertson, G.G. A Morphological Analysis of the Design Space of Input Devices. *ACM Trans. Information Systems*, 9(2):99-122, 1991.
8. Card, S., Mackinlay, J., Shneiderman, B. *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann Publishers, 1998.
9. Card, S., Robertson, G., Mackinlay, J. The Information Visualizer, an Information Workspace. In *Proc. ACM Human Factors in Computing Systems, CHI'91*, ACM Press, p.181-187, 1991.
10. Fitzmaurice, G., Ishii, H., Buxton, W. Laying the Foundations for Graspable User Interfaces. In *Proc. ACM Human Factors in Computing Systems, CHI'95*, ACM Press, p.442-449, 1995.
11. Foley, J., Wallace, V., Chan, P. The Human Factors of Computer Graphics Interaction Techniques. *Computer Graphics and Applications*, 4(11):13-48, 1984.
12. Gram, C. & Cockton, G. *Design Principles for Interactive Software*, Chapman & Hall, 1996.
13. Guiard, Y. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 19:486-517, 1987.
14. Heller, D., Ferguson, P.M., Brennan, D. *Motif Programming Manual*, O'Reilly & Associates, 1994.
15. Holland, S. & Oppenheim, D. Direct Combination. *Proc. ACM Human Factors in Computing Systems, CHI'99*, ACM Press, p.262-269, 1999.
16. Jacob, I. & Oliver, J. Evaluation of Techniques for Specifying 3D Rotations with a 2D Input Device. *Proc. HCI'95 Conference, People and Computers X*, p.63-76, 1995.
17. Jacob, R., Sibert, L., McFarlane, D., Preston Mullen, M. Integrability and Separability of Input Devices. *ACM Trans. Human Computer Interaction*, 1(1), p.3-26, 1994.
18. Krasner, G.E. & Pope, S.T. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk80 System. *J. Object Oriented Programming* 1(3):26-49, 1988.
19. Kurtenbach, G. & Buxton, W. User Learning and Performance with Marking Menus. *Proc. ACM Human Factors in Computing Systems, CHI'94*, ACM Press, p.258-264, 1994.
20. Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton. W. The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency. In *Proc. ACM Human Factors in Computing Systems, CHI'97*, ACM Press, p.35-42, 1997.
21. Kurtenbach, G., Fitzmaurice, G.W., Owen, R.N., Baudel, T. The Hotbox: efficient access to a large number of menu-items. *Proc. ACM Human Factors in Computing Systems, CHI'99*, ACM Press, p.231-237, 1999.
22. Michotte, A. *La perception de la causalité.* Publications Universitaires de Louvain, 1946.
23. Myers, B.A. A New Model for Handling Input. *ACM Trans. Information Systems*, 8(3):289-320, 1990.
24. Myers, B.A. User Interface Software Tools. *ACM Trans. Computer-Human Interaction*, 2(1):64-103, 1995.
25. Robertson, G.G. & Mackinlay, J.D. The Document Lens Visualizing Information. *Proc. ACM Symposium on User Interface Software and Technology*, p.101-108, 1993.
26. Shneiderman, B. Direct Manipulation : a Step Beyond Programming Languages. *IEEE Computer*, 16(8), pp 57-69, 1983.
27. Smith, D., Irby, C., Kimball, R., Verplank, B., Harslem E. Designing the Star User Interface. *Byte*, 7(4), p.242:282, 1982.
28. Szekely, P., Luo. P., Neches, R. Beyond Interface Builders: Model-Based Interface Tools Model-Based. *Proc. ACM Human Factors in Computing Systems, INTERCHI'93*, ACM Press, p.383-390, 1993.
29. Took, R. Surface Interaction: A Paradigm and Model for Separating Application and Interface. *Proc. ACM Human Factors in Computing Systems, CHI'90*, ACM Press, p.35-42, 1990.
30. Wellner, P., Mackay, M., Gold R. Computer-Augmented Environments. *Special Issue of Communications of the ACM*, 23(7).