

GRASPABLE USER INTERFACES

by

George W. Fitzmaurice

A thesis submitted in conformity with the requirements
of the Degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Copyright © 1996 George W. Fitzmaurice

Graspable User Interfaces

Ph.D. Thesis, 1996
George W. Fitzmaurice
University of Toronto,
Dept. of Computer Science

Abstract

This dissertation defines and explores Graspable User Interfaces, an evolution of the input mechanisms used in graphical user interfaces (GUIs). A Graspable UI design provides users concurrent access to multiple, specialized input devices which can serve as dedicated physical interface widgets, affording physical manipulation and spatial arrangements. Like conventional GUIs, physical devices function as “handles” or manual controllers for logical functions on widgets in the interface. However, the notion of the Graspable UI builds on current practice in a number of ways. With conventional GUIs, there is typically only one graphical input device, such as a mouse. Hence, the physical handle is necessarily “time-multiplexed,” being repeatedly attached and unattached to the various logical functions of the GUI. A significant aspect of the Graspable UI is that there can be more than one input device. Hence input control can then be “space-multiplexed.” That is, different devices can be attached to different functions, each independently (but possibly simultaneously) accessible. This, then affords the capability to take advantage of the shape, size and position of the physical controller to increase functionality and decrease complexity. It also means that the potential persistence of attachment of a device to a function can be increased. By using physical objects, we not only allow users to employ a larger expressive range of gestures and grasping behaviors but also to leverage off of a user's innate spatial reasoning skills and everyday knowledge of object manipulations.

In this thesis the concept of Graspable user interfaces is defined. Support for the concept is provided from the psychological literature. Instantiations of the concept are found in existing user interfaces. A task analysis of an existing interface's input activities and how to convert these to Graspable user interface devices is presented. The possible uses and implementation difficulties of bricks, a specific Graspable user interface are investigated. Finally, the advantages of two of the Graspable UI properties over conventional time-multiplexed generic input devices is measured by two controlled experiments.

Acknowledgments

Cooking is a hobby of mine. During the last months of my thesis, I took a cooking class to keep my sanity and to keep me well fed. I was quite surprised at the similarities between cooking and producing a thesis. Cooking involves lots of experimenting. Many times you can salvage a dish if you have gone a little bit crazy with your spices and herbs, put in the wrong proportions or let it cook a bit too long. Sometimes nothing can be done and you have to discard what you've done and start over. Any good dish starts off with a good stock and a good idea. The main course often consists of a nice piece of meat which you sear with high heat to lock in the flavors, crisping and caramelizing the exterior. Your committee, somewhat responsible for supplying the heat, helps you deglaze the pan using the stock, scrubbing away the encrusted bits. The end result is a clean pan and a flavorful sauce. Next, you reduce the liquid to intensify the flavors and add flour, a thickener, to give it more body. Add garnishes, and you are done. A wonderful thesis is left for you to enjoy.

I am very grateful to my committee members: Mark Chignell, Calvin Gotlieb, Marilyn Mantei, Michiel van de Panne and Christine MacKenzie (my external examiner) who helped refine the concepts and presentation of this thesis. Special thanks go to Marilyn, my internal examiner, who emphasized the need for rigor and precision in my thesis through tireless reviews of my dissertation document and in-depth conversations. I am indebted to my supervisor, Bill Buxton, who allowed me to take on a challenging thesis topic and supplied me with enthusiasm, support and creative insight. This kept me motivated through the tough times and this thesis would not be possible without his generous efforts.

I am quite fortunate to have been surrounded by smart, funny and caring friends and colleagues who made significant contributions to my thesis and my well being. Gordon Kurtenbach, whom I consider a great friend and researcher, read early drafts of my thesis, listened to my many concerns, dissipated them with ease and gave sage advice. William Hunt made me laugh, introduced me to vegetarian

cuisine, and provided solid friendship and advice as we both progressed through the Ph.D. program. Hiroshi Ishii supplied lots of creativity and energy to my thesis; collaboration with him was a true pleasure and a gift (by the way, most of the hand drawings in this thesis are his). Shumin Zhai lent his research expertise, especially on statistics, and I truly value our collaborations and friendship. I am also grateful to Beverly Harrison for her research insights and friendship and to Thomas Baudel who commented on early drafts of the thesis, and provided technical assistance.

I would also like to thank members of the Dynamic Graphics Project (DGP) and the Input Research Group which provided a friendly and resourceful environment for conducting this research. I am also grateful to Andy van Dam who encouraged me to pursue a Ph.D. and to Ron Baecker who recruited me and helped begin my Ph.D. studies in Toronto. Trips back to Boston to see my family and friends helped rejuvenate me and keep me grounded; thanks especially to Francis Janes, Brian Terpstra, Ken Seiss and Mary Bramante. Finally, I would like to thank Ravin Balakrishnan, Tim Brecht, Lucia Dow, Marc Griffiths, Diane Whitehouse, Kevin Schlueter, Russell Owen, George Drettakis, Sara Zuberec, Rhona Charron and my housemates, past and present (Kapil Kamra, Janet Selby, Marie-Christine Gagne, Jennifer Meechan and Marcus Doyle) for keeping me sane and in good spirits.

Alias|Wavefront provided support and resources for this thesis and their contributions are much appreciated. Wacom Technologies supplied us with many sample tablet sensors for us to experiment with and to build prototype input devices. The Arnott Design group constructed two devices (the stretchable square and ruler) for use in both of the experiments in Chapter 6. Ferdie Poblete designed and built the first foam-core model of the stretchable-square which was used in the exploratory study.

Lastly, I would like to thank my siblings (Jean, Julie, Elizabeth) and my parents (George and Rose) who provided endless supplies of encouragement and advise. This thesis is dedicated to them, especially my parents who instilled in me the value of education and the rewards and opportunities it can generate.

Thanks folks and Bon Appetite!

For my parents, George and Rose,
and my sisters, Jean, Julie and Elizabeth

Table of Contents

Chapter 1: Introduction	1
1.1 Motivation.....	1
1.2 Basic concepts.....	2
1.2.1 Graspable functions not devices.....	2
1.2.2 User interface characterization.....	4
1.3 Core defining properties.....	9
1.3.1 Space-multiplexed input and output.....	9
1.3.2 Concurrency.....	12
1.3.3 Strong specific devices.....	13
1.3.4 Spatially aware computational devices.....	13
1.3.5 Spatial reconfigurability of devices.....	14
1.4 Weak instantiations of Graspable UIs.....	14
1.5 Thesis statement and overview.....	16
Chapter 2: Theoretical foundations	17
2.1 Epistemic and Pragmatic action.....	18
2.2 Intelligent use of space.....	21
2.3 Things that make us smart.....	24
2.4 Notations that make us smart.....	27
2.5 Sensorimotor integration.....	27
2.6 Two handed interactions.....	28
2.7 Summary.....	28
Chapter 3: Related research and systems	31
3.1 Summary of Graspable UI Properties.....	31
3.2 Computer Augmented Environments.....	35
3.2.1 Ubiquitous Computing.....	35
3.2.2 Digital Desk.....	37
3.2.3 Mosaic.....	38
3.2.4 KARMA.....	38
3.3 Physical Manipulation Interfaces.....	39
3.3.1 Synthetic physical manipulations.....	39
3.3.2 3-Draw.....	40
3.3.3 Passive Interface Props.....	41
3.3.4 Dinosaur input device.....	42
3.3.5 LegoWall.....	43
3.3.6 Behavior Construction Kits.....	45
3.3.7 Programmable brick.....	46
3.3.8 AlgoBlock.....	47
3.3.9 Phantom chess.....	47
3.3.10 Wacom Character devices.....	48

3.4 Summary	50
Chapter 4: A Design Evolution – from GUI to a more Graspable UI.....	53
4.1 Keyframe animation and current GUI design	53
Four general control categories.....	53
4.2 Matching input devices to tasks	55
4.3 Stages of Evolution	57
Stage 0: Status quo	57
Stage 1: MIDI sliders for character control.....	58
Stage 2: Space ball for time and view controls.....	59
Stage 3: Space mouse for time and view controls	60
Stage 4: VCR time controls	64
Stage 5: Mobile scrubwheel	66
4.4 User Evaluation.....	68
4.5 Summary	71
Chapter 5: Bricks – a detailed implementation and case study.....	73
5.1 Exploratory studies.....	74
5.1.1 LEGO separation task.....	74
5.1.2 Domino sorting task	76
5.1.3 Physical manipulation of a stretchable square	77
5.1.4 Comparison Using MacDraw Application.....	79
5.1.5 V-Blocks: Physical vs. Virtual Manipulation	80
5.1.6 Curve Matching.....	82
5.1.7 Mock-up and simulations	82
5.2 Prototype 1: Bricks for drawing.....	83
5.2.1 Implementation	83
5.2.2 GraspDraw application.....	84
5.3 Prototype 2: Bricks for curve editing.....	88
5.3.1 Implementation	89
5.3.2 Development in larger application context.....	90
5.4 Prototype 3: FlipBricks	93
5.5 Summary	96
Chapter 6: Empirical Evaluations: space-multiplex input.....	99
6.1 Experiment 1: Manipulating physical/logical devices.....	100
6.1.1 Design.....	100
6.1.2 Hypotheses.....	102
6.1.3 Method.....	103
6.1.4 Results and discussion	107
6.2 Experiment 2: Acquiring physical/logical devices.....	114
6.2.1 Design.....	114
6.2.2 Hypotheses.....	116
6.2.3 Method.....	116

6.2.4 Results and discussion	122
6.3 Summary	129
Chapter 7: Conclusions	131
7.1 Summary	131
7.2 Contributions.....	134
7.3 Limitations, challenges and open issues.....	135
7.4 Future work	137
7.5 Closing remarks	138
Appendix A: An overview of Prehension	141
A.1 What is Prehension	141
A.2 Phases of Prehension	143
A.3 Studies on Prehension	144
Appendix B: Design variations for Bricks	147
B.1 Bricks without dynamic virtual context	147
B.2 Not just bricks.....	148
B.3 Revealing affordances of virtual and graspable objects.....	148
B.4 Virtual task to Real task back to Virtual tasks	149
B.5 Wiping with a brick	150
B.6 3D controller brick for Toolglass	151
B.7 System Reciprocity: Self-Propelled bricks.....	151
Appendix C: Experiment 1 – Coordination analysis	153
Appendix D: Experiment 1 – Statistical Results	157
Appendix E: Experiment 2 – Statistical Results.....	159
References	161

The bottleneck in improving the usefulness of interactive systems increasingly lies not in performing the processing task itself but in communicating requests and results between the system and its user. The best leverage for progress in this area therefore now lies at the user interface, rather than the system internals. Faster, more natural, and more convenient means for users and computers to exchange information are needed. On the user's side, interactive system technology is constrained by the nature of human communication organs and abilities; on the computer side, it is constrained only by input/output devices and methods that we can invent. The challenge before us is to design new devices and types of dialogues that better fit and exploit the communication-relevant characteristics of humans.

Faster, more natural – and particularly less sequential, more parallel – modes of user-computer communication will help remove this bottleneck [Jacob et al., 1993].

Chapter 1: Introduction

1.1 Motivation

Direct manipulation [Hutchins et al., 1986] is a fundamental concept of human-computer interaction. Surprisingly, it is a difficult concept to precisely define (as well as find standard metrics for measuring) the “directness” or the “manipulability” of a computer interface. Nevertheless, direct manipulation is often a primary goal for many interface designers. Shneiderman describes direct manipulation interfaces as having the following three properties:

1. Continuous representation of the object of interest.
2. Physical actions or labeled button presses instead of complex syntax.
3. Rapid incremental reversible operations whose impact on the object of interest is immediately visible [Shneiderman, 1982, p. 251].

Conventional graphical user interfaces (GUIs) are based on the concept of direct manipulation. However, we argue that the level of directness and manipulation for GUIs have not evolved or changed much in the last ten years. We still use a keyboard and mouse, with icons and menus and our gestural vocabulary ranges from a small set of actions such as point, click and drag. Has the GUI reached its final evolution?

We argue that improving the “directness” and the “manipulability” of the interface can be achieved by improving the input mechanisms for graphical user interfaces. Current graphical user interfaces (GUIs) [Hutchins et. al. 1986; Marcus 1990; Johnson et. al. 1983] are designed to operate with a minimal set of physical tools (i.e., a keyboard and mouse). As our computer tasks become more complex, intricate and demanding, we may benefit by having access to specialized physical tools and redefining how such tools interact with the underlying software. This is the topic

explored in this thesis. The user interface that results, we call the “Graspable User Interface.”

In the simplest definition, a Graspable User Interface is a physical handle to a virtual function where the physical handle serves as a dedicated functional manipulator. The term Graspable UI refers to both the ability to physically grasp an object (i.e., placing a hand on an object) as well as conceptual grasping (i.e., to take hold of intellectually or to comprehend). At the very least, Graspable UIs can serve as physical embodiments and representations of common graphical user interface elements (such as file icons, windows, menus or push buttons). As well, Graspable UIs have the potential to aid users in manipulating abstract representations of objects or functions on a display.

1.2 Basic concepts

Graspable UIs provide users concurrent access to multiple, specialized input devices which can serve as dedicated physical interface widgets, affording physical manipulation and spatial arrangements. Like conventional graphical user interfaces (GUIs), physical devices function as “handles” or manual controllers for logical functions on widgets in the interface. However, the notion of the Graspable UI builds on current practice in a number of ways. With conventional GUIs, there is typically only one graphical input device, such as a mouse. Hence, the physical handle is necessarily “time-multiplexed,” being repeatedly attached and unattached to the various logical functions of the GUI. A significant aspect of the Graspable UI is that there can be more than one input device. Hence input control can then be “space-multiplexed.” That is, different devices can be attached to different functions, each independently (but possibly simultaneously) accessible. This then affords the capability to take advantage of the shape, size and position of the physical controller to increase functionality and decrease complexity. It also means that the potential persistence of attachment of a device to a function can be increased.

1.2.1 Graspable functions not devices

We are proposing a conceptual shift in thinking about physical input devices not as graspable *devices* but instead as graspable *functions*. In the traditional sense, almost all physical input devices are “graspable” in that one can physically touch and hold them. However, we are exploring the utility of designing the physical devices as graspable functions. That is, a graspable function consists of a specialized physical input device which is bound to a virtual function and can serve as a functional

manipulator. The difference between a GUI and a Graspable UI which uses graspable functions is shown in Figure 1.1. With traditional GUIs there are often three phases of interaction: (1) acquire physical device, (2) acquire logical device (e.g., a UI widget such as a scrollbar or button) and (3) manipulate the virtual device. Alternatively, with Graspable UIs, we can often reduce the phases of interaction to: (1) acquire physical device and (2) manipulate the logical device directly. This is possible because the physical devices can be persistently attached to a logical device. Thus, the devices serve as dedicated graspable functions.

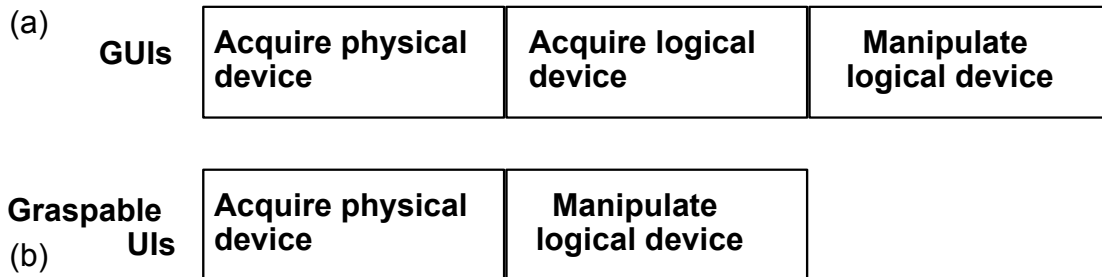


Figure 1.1. Phases of interaction for (a) traditional GUIs and (b) Graspable UIs.

Having a dedicated physical input device for every function can be costly and potentially inefficient. Figure 1.2 shows an example of two input configuration styles: the time-multiplexed mouse and the space-multiplexed audio mixing console.

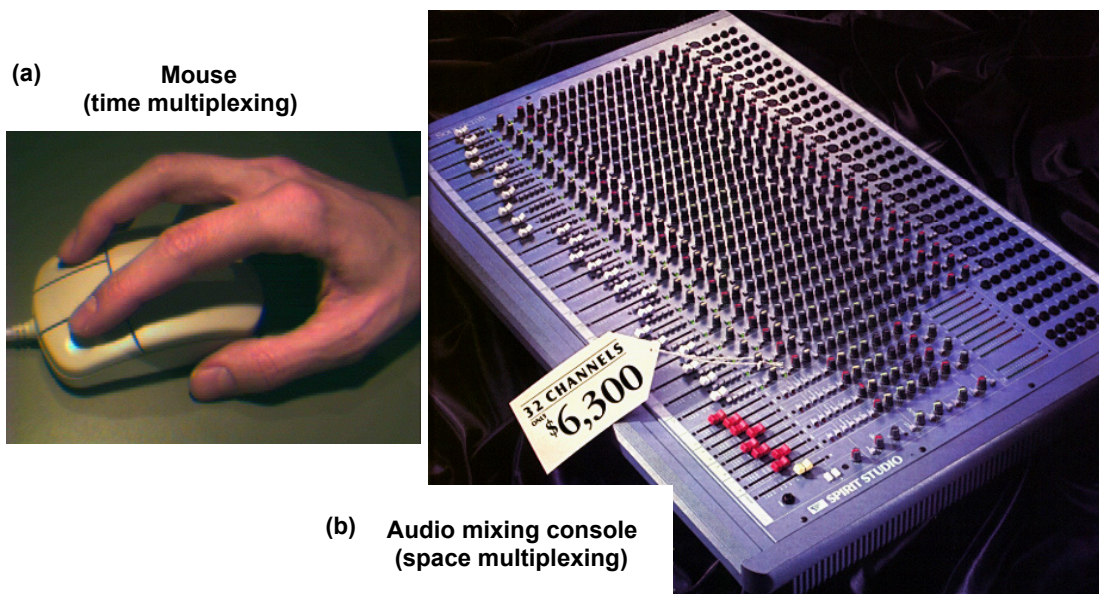


Figure 1.2. Two extreme input configuration styles: (a) the mouse, a time-multiplexed design and (b) an audio mixing console, a space-multiplexed design.

The mouse is a generic all-purpose pointing device which is constantly attached and detached to logical devices. In contrast, the audio mixing console has hundreds of physical transducers (e.g., sliders, dials, buttons) each assigned a function. Which input configuration is more desirable, more direct or more manipulable? We believe the ultimate benefits lie somewhere in between these two extremes.

1.2.2 User interface characterization

We now describe a user interface characterization of a Graspable UI to provide the feel and flavor of the concepts developed in the thesis. The Graspable UI consists of using generic physical handles, what we call Bricks [Fitzmaurice et. al. 1995], as handles to virtual objects. The bricks are approximately the size of LEGO^a bricks. In the default configuration, multiple physical bricks sit and operate on a large horizontal computer display surface (the Active Desk). Thus, the physical input control space and virtual output display space are superimposed. The bricks act as input devices and are tracked by the host computer. From the computer's perspective, the brick devices are tightly coupled to the host computer – capable of constantly receiving brick related information (e.g., position, orientation and selection information) which can be relayed to application programs and the operating system.

The physical bricks allow for direct control of electronic objects by acting as tactile handles for control. These physical artifacts are essentially "graspable functions" – input devices which can be tightly coupled or "attached" to virtual objects for manipulation, or for expressing actions (e.g., to set parameters, or to initiate processes). Figure 1.3 shows an example of a simple *graspable* user interface configuration consisting of two components: (1) a physical object, what we call a "brick" and (2) a virtual object, (in this case a rectangle).

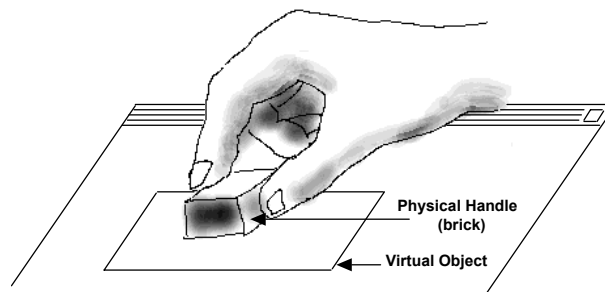


Figure 1.3. A Graspable UI configuration.

One Handle

In the simplest case, we can think of the bricks as handles similar to that of graphical handles in computer drawing programs such as MacDraw (see Figure 1.4). A physical handle (i.e., a brick) can be attached to an object to move or rotate it (see Figure 1.5). Note that the virtual object's center of rotation is at the center of the brick.



Figure 1.4. Traditional MacDraw-like application. Here electronic handles indicate that the square has been selected by the user.



Figure 1.5. Move and rotate virtual object by manipulating physical brick which acts as a handle to virtual objects. Placing a brick on the square selects the object and dragging the brick causes the square to be moved (center of rotation is at the center of the brick).

Grabbing a virtual object assumes that the user already has a brick in hand. One way a user can grab a virtual object is to place the brick directly on top of a virtual object (see Figure 1.6). That is, the brick will sense when it is "on" the desktop surface which will cause an "attachment" action to the virtual object. Sliding the brick on the surface will bring along any virtual objects currently attached to the brick. Raising the brick above the surface releases the virtual object from the physical handle (i.e., brick) and the two components are considered detached (see Figure 1.6).

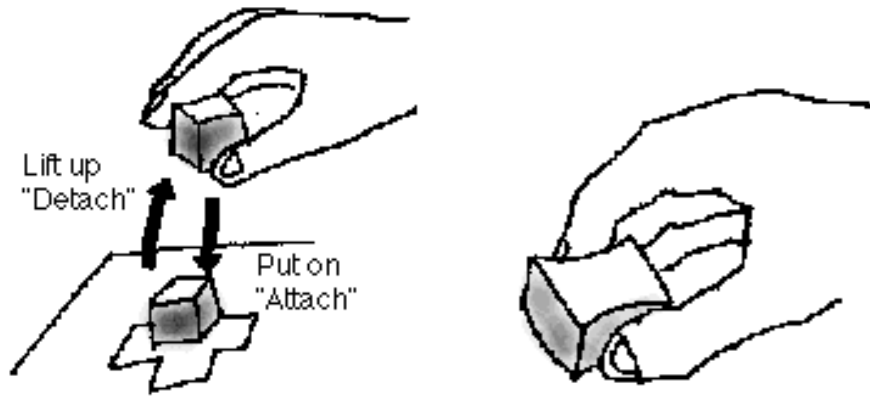


Figure 1.6. (a) Placing a brick on the display surface causes the virtual object beneath it to become attached. Lifting the brick off of the surface detaches the object. Figure 1.6 (b) shows a brick being grasped with natural grab points indicated by the curved brick sides.

A simple example application may be a floor planner (see Figure 1.7). Each piece of furniture has a physical brick attached and the user can arrange the pieces, most likely in a rapid trial-and-error fashion. This design lends itself to two handed interaction and the forming of highly transient groupings by touching and moving multiple bricks at the same time.

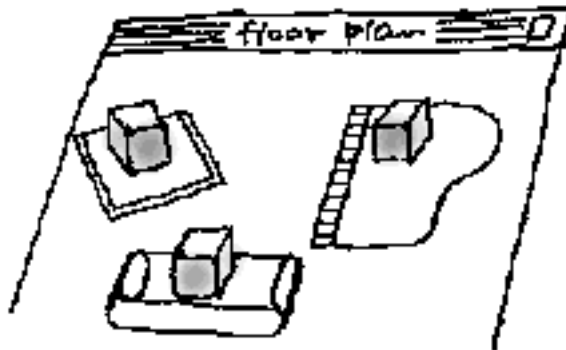


Figure 1.7. Simple floor planner application where electronic objects can have a physical brick handle to allow rapid moving and rotating within the workspace.

Multiple Handles

More sophisticated interaction techniques can be developed if we allow more than one handle (or brick) to be attached to a virtual object. For example, to stretch an electronic square, two physical bricks can be placed on an object. One brick acts like an anchor while the second brick is moved (see Figure 1.8).

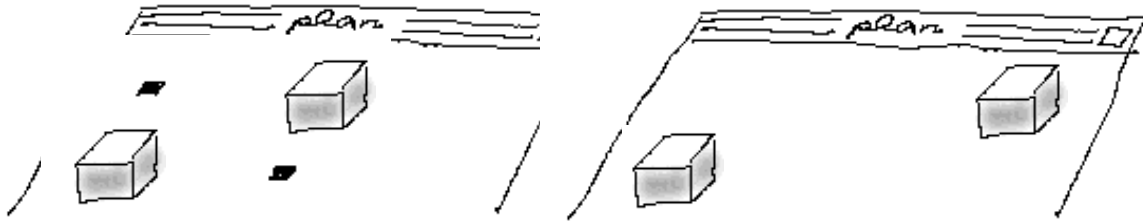


Figure 1.8. To stretch the square, two physical bricks can be used. One brick acts like an anchor while the second brick is moved.

Placing more than one brick on an electronic object gives the user multiple control points to manipulate the object. For example, a spline-curve can have bricks placed on its control points (see Figure 1.9). A more compelling example is using the position and orientation information of the bricks to deform the shape of an object. In Figure 1.10, the user starts off with a rectangle shaped object. By placing a brick at both ends and rotating them at the same time, the user specifies a bending transformation similar to what would happen in the real world if the object were made out of clay. It is difficult to imagine how this action or transformation could be expressed easily using a mouse.

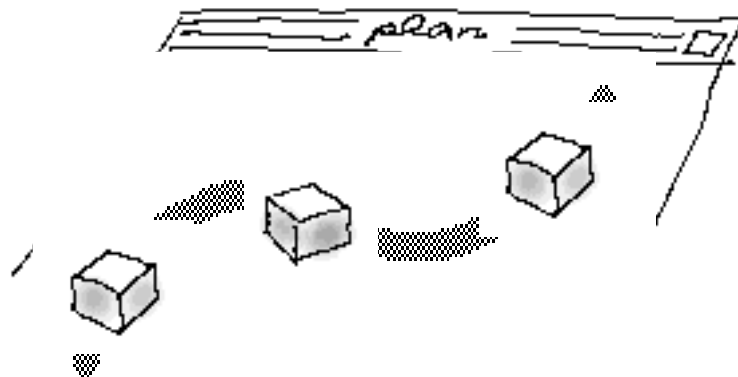


Figure 1.9. Many physical bricks can be used for specifying multiple control points. Here the bricks are used for creating a spline curve.

An extension of this design could allow the physical bricks themselves to be reshaped for a given task. In Figure 1.11 we see two bricks which have been deformed into curved pieces and act like "guard rails." That is, their *side* surface is being used as a constraint to the electronic or virtual object. The two bricks at the ends are used as anchors (their *bottom* surfaces are used).

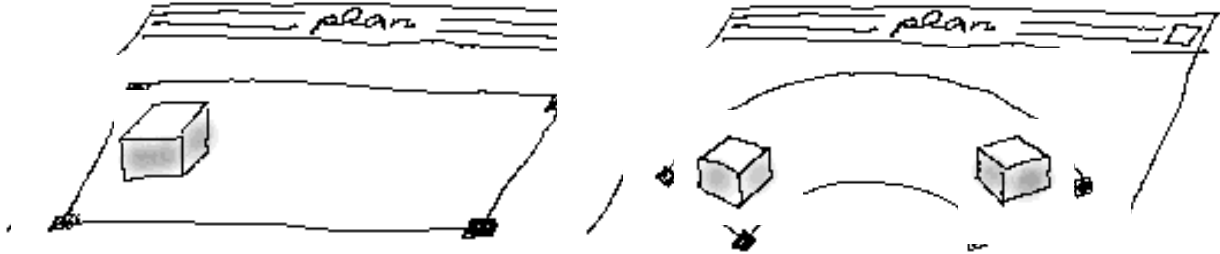


Figure 1.10. Moving and rotating both bricks at the same time causes the electronic object to be transformed as if it were being held in one's hand.

One key idea that this example illustrates is that the bricks offer a significantly rich vocabulary of expression for input devices. Compared to most pointing devices (e.g., the mouse) which only offers an x-y location, the bricks offer multiple x-y locations and orientation information at the same instances in time.

In this characterization, the bricks serve as graspable functions which can be persistently attached to virtual objects. In essence, the Graspable UIs outlined here are a blend of virtual and physical artifacts, each offering affordances in their respective instantiation. In many cases, we wish to offer a seamless blend between the physical and virtual worlds. Finally, the design takes advantage of a space-multiplex instead of a time-multiplex input style.

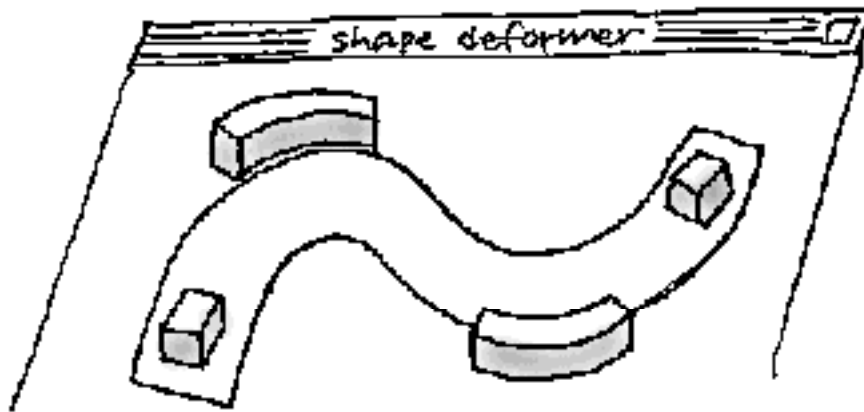


Figure 1.11 Bricks can be reshaped. Two "curve" shaped bricks are shown and act as "guard rails." Their *side* surface is being used as additional constraints to reshape the electronic object.

1.3 Core defining properties

We now more formally present five basic defining properties that embody the Graspable UI concept:

1. Space-multiplex both input and output
2. Allow for a high degree of inter-device concurrency
3. Increase the use of strong specialized input devices
4. Have spatially-aware computational devices
5. Have high spatial reconfigurability of devices and device context

1.3.1 Space-multiplexed input and output

The primary principle behind Graspable UIs is to adopt a space-multiplexed input design. Input devices can be classified as being *space-multiplexed* or *time-multiplexed*. With space-multiplexed input, each function to be controlled has a dedicated transducer, each occupying its own space. For example, an automobile has a brake, clutch, throttle, steering wheel, and gear shift which are distinct, dedicated transducers controlling a single specific task.

In contrast, time-multiplexing input uses one device to control different functions at different points in time. For instance, the mouse uses time-multiplexing as it controls functions as diverse as menu selection, navigation using the scroll widgets, pointing, and activating "buttons."

Furthermore, the Graspable UI design provides for a concurrence between space-multiplexed input and output. Traditional GUIs have an inherent dissonance in that the display output is often space-multiplexed (icons or control widgets occupy their own space and must be made visible to use) while the input is time-multiplexed (i.e., most of our actions are channeled through a single device, a mouse, over time). Therefore, only one user driven, graphical manipulation task can be performed at a time, as they all use the same transducer. The resulting interaction techniques are often sequential in nature and mutually exclusive. Graspable UIs attempt to overcome this.

The space- versus time-multiplex input style classification can also be thought of in terms of algorithm designs which often have a time versus space efficiency tradeoff.

An algorithm that takes less time to execute often requires more space (i.e., memory) to perform the calculations. Conversely, an algorithm may take a much longer amount of time to complete but consumes smaller amounts of space. Grasable UIs shift interactions to a more space-multiplexed design.

Adopting a space-multiplexed input scheme has a number of implications.

Increased use of motor channel. As the visual channel becomes taxed, the space-multiplex input style may offload some of the visual demands onto the underutilized tactile or motor systems. Many sophisticated software packages make intense use of the visual channel to display user interface widgets, state information, and application data. Even more use of the visual channel is used for software packages that operate on 3D data. Here the idea is to transform some of the virtual UI widgets and functionality onto physical widgets. This process frees up some of the valuable screen space, reducing the need to display static UI widgets and instead display more application data. For example, a set of scroll bars are often attached to each GUI window. They are made small to minimize the consumption of screen real estate. However, this very thing makes them all the harder to acquire. Scroll bars are good candidates for transforming into physical widgets.

This point is especially highlighted in applications such as animation and video which are visually demanding tasks. Using conventional GUIs, most user interface widgets, such as transport controls or scroll bars, for example, require the visual channel to operate. This results in contention for the channel between the application and control tasks. Having physical controls dedicated to the function in question potentially afford (but do not guarantee) eyes-free access to the control function, thereby leaving the visual channel free for the primary application task. As always, it is the quality of design of the user interface which first must be satisfied before this potential benefit is realized.

Physical instantiation. Which user interface elements should take on a physical form to allow users to more readily manipulate and interact with them? Many times we think of input devices as physical objects that point to and act on virtual objects. Instead, we can think of the physical objects as a permanent part or feature of the virtual objects. In terms of design, the idea is to choose which features of the hybrid objects should be physical or virtual based on whichever medium is best suited for representing and carrying out the user's task.

In addition, the physical laws and constraints of everyday object interaction will govern the overall interaction behavior between the physical and virtual interface elements. For example, physical laws dictate that two objects cannot occupy the same space or that two marbles cannot easily be stacked on top of each other.

Everyday skills and learned motor behaviors. Using more physical artifacts in the user interface implies that we can tap into our everyday skills of object manipulation and learned motor behaviors. The physical object manipulations are possible by knowledge we have learned through a lifetime of practice. We already know how to manipulate physical objects. Our innate motor abilities, sense of touch and texture discrimination, and our everyday skill in grasping, gesturing and manipulation will all contribute to the performance gains of Graspable UIs. The tradeoff is designing the proper Graspable UI objects and the learning time needed to understand the relationship between the physical manipulation and corresponding virtual action.

Using the motor system has the advantage in that users can become skilled at issuing commands through learned motor behavior [Singer, 1980; Schmidt, 1988]. In our current GUIs, it is very difficult to tap into our spatial memory (or "muscle memory"). For example, it is almost impossible to select from "Save" and "Save As" (i.e., adjoining items) in a pull down menu using a mouse without visual feedback to discriminate between the menu items. Touch typing, in contrast, is a great example of how learned motor behaviors can be effectively used for efficient interaction without having a strong dependency on the visual channel for continuous feedback.

Multiple devices. A space-multiplexed input design implies that there will be more than one input device for users to manipulate. At the extreme, each of these devices should be assigned a permanent graspable function.

Multiple persistent selections. Having multiple devices allows interfaces to have multiple persistent selections. Graspable UIs make a distinction between "attachment" and "selection." In traditional graphical UIs, the selection paradigm dictates that there is typically only one active selection; selection N implicitly causes selection N-1 to be unselected. In contrast, when graspable input devices are attached to virtual objects the association persists across multiple interactions. Selections are then made by making physical contact with the devices (i.e., not having to grab the physical device and re-acquire the virtual object). Therefore, with Graspable UIs we can possibly eliminate many of the redundant selection actions

and make selections easier by replacing the act of precisely positioning a cursor over a small target with the act of grabbing a device.

1.3.2 Concurrency

Having multiple devices available, we can then consider interactions that allow for concurrent access and manipulation of interface components. Moreover, there are different categories of concurrency. For example, functional coupling has two or more devices operated simultaneously to achieve a desired goal. One instance is the mouse and the shift modifier key on the keyboard. When the shift key is down during a mouse press, the selected item is added or removed from the current selection set. In contrast, a physical coupling could exist. That is, when one physical device is manipulated, one or more additional devices are physically affected as a byproduct of the original device manipulation. Both styles of concurrency apply for inter-device and intra-device interactions.

Since we are adopting a space-multiplex design, we are able to develop interaction techniques that can use multiple devices at the same time. This is not possible with a system that has only one device being used in a time-multiplex design; a sequential instead of parallel manipulation style must be employed.

Moreover, we make the distinction of foreground and background concurrency. With foreground concurrency, users are actively manipulating two or more input devices or actuators. Background concurrency deals with the remaining input devices and actuators that are immediately available to access. These devices which are nearby but not "in-hand" can be considered "in-use" as, at the very least, they remind the user as to what functionality is available. Note that users will switch between devices and modes of concurrency (devices in use are being pushed into the background while others are being brought into the foreground).

Using multiple manual devices concurrently suggests that the interactions will involve the use of two hands. We use both of our hands while driving a car, cooking, drawing, sculpting or even playing a piano. With the GUI there have been a limited application of two handed interactions (e.g., keyboard typing). However, there is a rich gamut of two handed interactions that are possible (e.g., two handed discrete actions; one handed discrete with the other hand continuous; two handed continuous, etc.). As well, we can go beyond two handed interactions as driving an automobile involves two hands and sometimes two feet.

1.3.3 Strong specific devices.

When considering a Graspable UI, we must consider the tradeoff of generality vs. specificity. Having a general, all purpose tool allows one to use it for solving many tasks with the tradeoff of not being very efficient; it is convenient, familiar and basically gets the job done. This is true for the mouse which is a very generalized pointing device. For Graspable UIs, we advocate moving in the direction of multiple, specialized physical objects (i.e., tools or physical widgets) for interacting with the computer. This offers more efficiency in that the physical objects are designed to be more specialized and tailored for working on a given task. Yet, the specialized tools lose some of their generality; it may be very difficult to use a tool for a task which it was not designed for. This apparent loss of generality, however, may be overcome by the task specific power a collection of physical tools (or input devices) provides. Stated differently, the value of the Graspable UI may lie mainly in specialized domains such as in animation or computer-aided design, rather than in general purpose computing.

1.3.4 Spatially aware computational devices

The interface elements that do take on a physical form should be spatially aware of their surroundings and be registered with a central processing unit. It is important to note that both *position* and *orientation* (possibly in 3 dimensions) are critical pieces of information to be sensed. Communication to a central processing unit or independent sensors on each device can also determine *proximity* information between devices. As computer tasks become more graphic intensive instead of alpha-numeric intensive, we argue for an increased benefit of having spatially-aware devices as graphical tasks are inherently spatial in nature.

For example, physical devices can act as control points for manipulating a spline-curve (as illustrated earlier in Figure 1.9). These devices are spatially-aware. That is, the devices know their position on a given surface (i.e., digitizing tablet) and the application can query the devices for this spatial information. More intense applications of spatially-aware devices can be defined. For example, the Chameleon system [Fitzmaurice and Buxton, 1994] envisions an application in which a spatially-aware, hand-held computer can be used for diagnostic procedures for a rack of video equipment. Since the components occupy unique regions of space, there is a natural spatial mapping for placing the virtual diagnostic hotspots (i.e., directly over the corresponding physical components.) As the device is positioned over components of the video equipment, diagnostic information and current state

information can be displayed on the hand-held unit. The Chameleon device is spatially-aware of its surroundings.

1.3.5 Spatial reconfigurability of devices

While the placement, orientation or proximity of spatially aware devices are important, these devices operate within a context or situation. That is, the surrounding environment (e.g., walls, everyday objects, people, day, time, etc.) all contribute to the meaning, purpose, and intent of the device. The contextual space a device occupies contributes to its overall function and use. Even when a device is not currently being used (i.e., not held in one's hands) it is still very valuable in that it serves as an external cognitive aid (e.g., reminds the user of particular functions or data) operating within the current context or situation. The ability to rapidly reconfigure and rearrange a set of devices in a workspace is important in that it allows users to customize their space to facilitate task workflows and rapid task switching. Finally, note that the context is intended to be physical context but will undoubtedly include virtual context as well.

Finally, we ask if all five properties are necessary in order to form a Graspable user interface. The first property (space-multiplex input) is essential. The remaining four properties are derived from the first property. Depending on the application and task, it may be possible to neglect some of the remaining four properties but at the risk of designing a weak, clumsy or inefficient user interface.

These defining Graspable UI properties will be examined and discussed throughout this thesis. However, the core property that will be examined in depth is the space-multiplexing for input devices.

1.4 Weak instantiations of Graspable UIs

Having defined what Graspable UIs are, it is valuable to describe a few input interfaces which either do not follow the Graspable UI designs or are weak instantiations of Graspable UIs. Voice input is a great example of not exhibiting Graspable UI properties; there is no physical instantiation (i.e., nothing to grasp), no spatial awareness (except perhaps sophisticated digital signal processing systems with multiple microphones which can detect the 3D location of a sound source) and no contextual awareness. Similarly, eye-tracking input devices do not match the profile of Graspable UIs. That is not to say that these techniques may not be able to work effectively in conjunction with Graspable UIs.

Before considering some more conventional haptic input devices, it is necessary to specify how the device is being used for a given task before we can judge it. For example a trackball could be considered a Graspable UI if our entire task only involved one function or action (i.e., “do it” now). That is, the trackball is acting as a dedicated physical manipulator. However, rarely, if ever, do we build applications that have only one function associated with the input device (the videogame PACman being one of them). Usually, the applications are bristling with functionality. In all of the following examples, we consider the input devices in the working context of applications that have hundreds of functions (e.g., graphics or animation software).

The mouse pointing device in the context of a conventional GUI interface can be considered a weak instantiation of a Graspable UI. At the most basic level, it is a graspable object since it has a physical form. The mouse can be attached to virtual objects but typically at a very transient level (the attachment often persists only as long as a mouse button is held down). In terms of spatial awareness, it has the ability to track its relative position while in motion. This is a very basic spatial awareness. For example, it does not have the ability to detect whether it is operating to the left or right side of the keyboard (i.e., the user is left or right handed). That is, in terms of contextual awareness, there is no physical context registered but instead virtual context exists. The graphical cursor, which is the mouse's representation in the virtual scene, can change shape and color while it is over different UI widgets. As mentioned earlier, the mouse has been designed as a highly generalized pointing device, capable of performing well to adequate in many interaction situations but often lacks the efficiency of more specialized tools. Finally, in our current GUI, the mouse has a time-multiplexed design; only one task can be performed at a time, as they all use the same transducer. Nevertheless, the mouse serves as a valid case study of a primitive Graspable UI and our study can be viewed as an attempt to build on its strength, while avoiding its weakness.

A keyboard does not exhibit enough properties to be considered a Graspable UI. While the keys are physical, they cannot be moved, rearranged or manipulated beyond being pressed downward. There is no corresponding spatial awareness in the virtual scene; the keyboard generates a stream of characters with no associated Euclidean coordinates. The keys are spatially arranged on the keyboard and thus have some physical context (i.e., the return key is to the right of the spacebar). The tasks are certainly space-multiplexed (compared to a single Morse-code telegraph

key which is time-multiplexed). In fact the persistent key layout allows users to develop learned motor behavior (i.e., touch typing). While the keyboard has the property of specialized function and persistence of attachment (i.e., space-multiplexed input), the lack of spatial awareness and the lack of physical rearrangement prevents us from considering the keyboard as a Graspable UI.

1.5 Thesis statement and overview

The thesis is a presentation of a new set of interaction techniques involving input devices. It argues that considerable advantages can be obtained by developing input devices that are specific to the functions they are meant to perform. Although others have generated input devices that work in this fashion, the concept of the devices being function manipulators is new, as is the exploration of this concept. The thesis proposes the existence of a class of dialogue styles, demonstrating their generality, usefulness and application to spatial tasks.

The thesis approaches this exploration via multiple avenues. It explores the literature on the advantages physical environments provide to problem solving and uses this to make a case for developing physically based input devices that are likely to give users the same advantages in interactions with computers (Chapter 2). Its second avenue explores the interface designs that include some instantiation of the Graspable user interface. It does this by listing a set of properties Graspable user interfaces need to possess (Chapter 3). Its third avenue walks the reader through the redesign of a commercial drawing interface showing how the multiple functions needed by the artist using the software can be mapped on to different forms of input devices to achieve a set of space-multiplexed input tools (Chapter 4). A second set of case studies is presented as the fourth avenue, which looks at potential uses of one particularly promising Graspable user interfaces, bricks. A variety of implementations is tried with bricks, all of which have some complications because of the difficulties of adapting existing software and systems to the Graspable user interface concept. This fourth avenue explores what is needed to make Graspable UIs a viable tool (Chapter 5). A fifth avenue examines the claims being made about the efficacy of the proposed new interaction style by two controlled experiments which look at the advantages of two of the five defining properties for Graspable user interfaces (Chapter 6). Finally, Chapter 7 summarizes this dissertation, its contributions and future research.

Chapter 2: Theoretical foundations

In this chapter we review some of the motor and cognitive psychology literature to provide the underlying theoretical support for workable Graspable user interfaces. One of the main themes of this work stems from a belief that we can offload some of our internal cognitive resources into the external world using (1) our motor actions, (2) a physical space and performing spatial arrangements, and (3) physical artifacts for externalizing and embedding rules and information for solving a task.

We start by presenting some recent literature on cognition that gives evidence for the use of motor action specifically performed to reduce internal computation (called epistemic action) rather than to reach an end goal (pragmatic action). The existence of epistemic action suggests that people can use physical objects and their environment to aid their cognition. Therefore, having a greater variety of manipulable objects and physical arrangements may further support the cognitive process while performing a task.

Next we present how the physical space that we work in affects or aids our cognition. Specifically, spatial arrangements can (1) simplify choice, (2) simplify perception, and (3) simplify internal computation. Next, we show how physical artifacts can be used as external representations which can automatically constrain actions and interpretations for users.

A requirement for workable Graspable UIs is understanding the range of ability a user has for grasping and manipulating physical objects. Appendix A gives an overview of the field known as prehension to serve as additional background material. In addition, because we want to develop natural, highly efficient, interaction techniques we wish to involve the use of both hands for our techniques. We briefly review some two handed literature for guiding the development of our designs and interaction techniques.

2.1 Epistemic and Pragmatic action

There is growing evidence that motor activity can be classified as either epistemic or pragmatic action [Kirsh and Maglio, 1994]. Epistemic actions are performed to uncover information that is hidden or hard to compute mentally. The physical actions make internal cognitive computation easier, faster and more reliable. For example, we sometimes use our fingers when we count. Or, novice chess players may physically move a chess piece, temporarily, to its new position to assess the move and possible counter-moves by an opponent.

The epistemic actions can improve cognition by:

- reducing the memory involved in mental computation (space complexity)
- reducing the number of steps in mental computation (time complexity), or
- reducing the probability of error of mental computation (unreliability).

In contrast, pragmatic actions are physical actions whose primary function is to bring the user physically closer to the goal. The distinction of epistemic and pragmatic action may have also been discovered by Gibson [1962] who suggested that hand movements can be classified as “exploratory” and “performatory.”

One may argue that all epistemic actions are inherently pragmatic actions as they involve aiding users in reaching their final goal. In the strictest sense this is true but misses an important distinction. If we view motor action from a purely efficiency point of view, users would think first, arrive at a decision and then perform the minimal motor action for making the external world match their internal state. In practice, this generally does not happen. As in the chess example just mentioned, many novice players move the physical pieces around on the board to candidate positions to assess the move and possible counter-moves by an opponent. From a motor perspective, this is very wasteful. However, from a cognitive perspective, this is quite beneficial.

An experimental study on the game tetris (see Figure 2.1) showed more rigorously the distinction between epistemic and pragmatic actions [Kirsh and Maglio, 1994]. In this experiment, the authors found that subjects would rotate pieces physically by hitting a button (taking less than 150 ms) rather than compute the mental rotation (which they estimate takes much longer, 800 to 1200 ms to rotate a piece 90 degrees).

In addition, they learned that epistemic actions of rotation were used to: (1) reveal new information early in the game (i.e., before a piece is entirely visible on the screen), (2) save mental rotation effort, (3) facilitate retrieval of shapes of pieces from memory, (4) make it easier to identify a piece's type, and (5) simplify the matching process of the falling piece with the contour below.

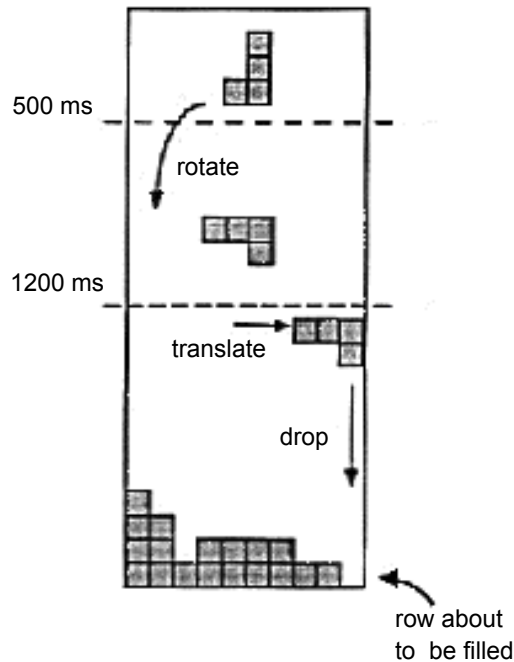


Figure 2.1. Tetris game. The falling pieces enter from the top and are rapidly placed somewhere on the contour below. Users can translate the piece left or right or rotate the piece by 90 degrees. Epistemic rotations sometimes occur before a piece is completely visible to more quickly identify the piece.

Moreover, Kirsh [1995b] describes a *complementary strategy* as “any organizing activity which recruits external elements to reduce cognitive loads. Typical organizing activities include positioning, arranging the position and orientation of nearby objects, writing things down, manipulating counters, rulers or other artifacts that can encode the state of a process or simplify perception (p. 1).” Kirsh has conducted a preliminary "coin-counting" experiment to test these concepts. He asked subjects to sum the value of a collection of coins drawn on a piece of paper. In one condition the subjects can only look at the paper and are prevented from using their hands. In the second condition, the subjects are allowed to use one hand which is often used to point at coins on the paper during the trials. He found improved

task completion times and reduced error rates for the condition which allow the use of hands (i.e., epistemic actions) [Kirsh, 1995b].

This kind of behavior supports a tightly integrated human processing model in which information needed for each step in the process (e.g., iconic buffer, attention, generate, match) can be supplied either by internal cognitive resources or by physically modifying and then perceiving the external environment (see Figure 2.2). That is, internal modules can request motor activity intended to cause changes in the external environment which assist in cognition. *The notion is that it takes less effort to physically modify and re-perceive the external world than it does to compute and retain the new information state internally.* Thus, epistemic actions are specifically targeted at improving one's performance.

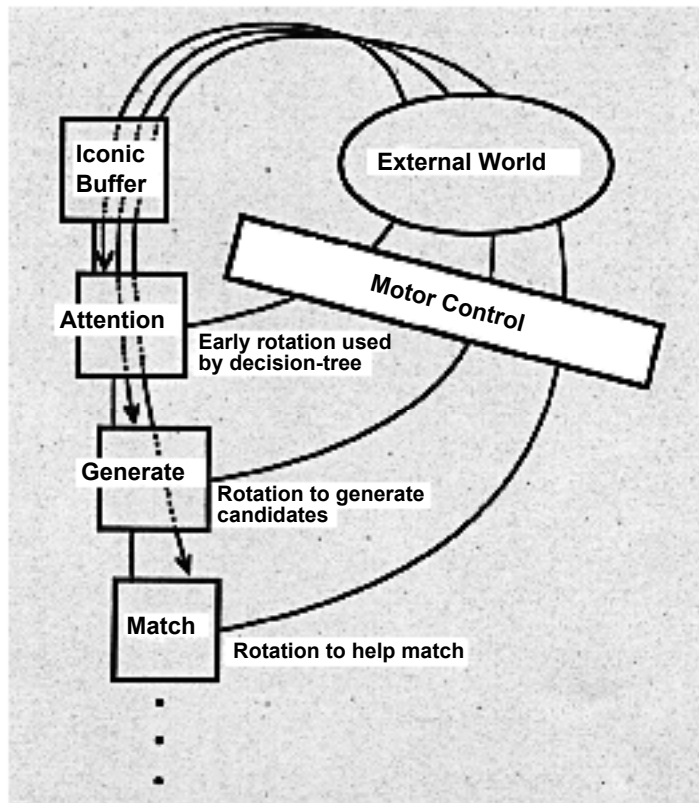


Figure 2.2. Kirsh's processing model [Kirsh and Maglio, 1994]. Here each step in the process can get information either from internal resources or from performing motor actions which affect the external environment and essentially serve as new input.

Interfaces having key components manifested as manipulable physical artifacts may offer more opportunity for epistemic rather than pragmatic actions. This is due to

the potential affordances of the physical interface. We have the potential to rapidly manipulate physical artifacts. The question is does the UI provide us with the affordances to utilize this potential? Underlying all of this is the notion that the problem is hard, yet we handicap ourselves in addressing it by not enabling us to apply skills and strategies in our repertoire.

Said differently, the key issue is the discord between our potential bandwidth (effector) to manipulate with the restricted bandwidth (receptor) of the input transducers. This mismatch is fundamental to the problem.

One key to epistemic action is low-cost manipulation of the external world. There are several ways to reduce the cost of manipulation. One simple way is to increase the number of degrees of freedom of the input channel. However, caution must be used as increasing the degrees of freedom does not guarantee that costs will be decreased.

The existence of epistemic action supports the shift towards Graspable UIs. By using physical objects, users are more able to manipulate and affect their environment. A greater variety of manipulable objects and physical arrangements may more readily support the cognitive process during a task. The amount of effort to use these objects and manipulate the environment must be minimal; this is an essential property in that the amount of effort and attention needed to manipulate the physical objects must be less than the internal cognitive computational effort to make it attractive to use.

One could argue that virtual objects in the computer world (e.g., icons, buttons) can serve as external cognitive artifacts. While this is true, we must take note at the amount of effort needed to access and manipulate these objects. Again, if the amount of effort and attention needed to manipulate these virtual objects is high, it may outweigh the value of using them as external cognitive aids. Nevertheless, external cognitive artifacts can exist in virtual form (i.e., within the computer) or physical form and we argue for Graspable UIs that use physical artifacts.

2.2 Intelligent use of space

A great deal of literature has been written on a human's ability to perform spatial reasoning tasks [Eilan, et al, 1993; Campbell 1993; Cooper and Munger 1993]. Moreover, "how we manage the spatial arrangement of items around us, is not an

afterthought; it is an integral part of the way we think, plan and behave [Kirsh, 1995a, p. 31].” The intelligent use of space has been described by Kirsh [1995a] in which he classifies spatial arrangements that: (1) simplify choice, (2) simplify perception and (3) simplify internal computation.

He states that “experts constantly re-arrange items to make it easy to: (1) track the state of the task, (2) figure out, remember, or notice the properties signaling what to do next, and (3) predict the effects of actions [p. 35].” For example, if a short order cook is making several dishes at the same time, he or she may cluster a collection of plates to match similar orders. The materials for the orders may be clustered together, placing knives, forks and other utensils near ingredients to be used next, essentially marking their place in the procedure. Garnishings may also be used to indicate special orders or variations in an order.

We can exploit the resources of the world to improve execution or to simplify problem solving. Kirsh gives an excellent example of a classic AI planning problem (see Figure 2.3) in which a child is asked to build two towers, one tower reading “SPACE” the other reading “MATTERS.” The blocks start out in a random placement and the following two rules apply: (1) only one block can be moved at a given time, (2) a block cannot be moved if another block is on top of it. Examining the problem, Kirsh observes that it is much easier to solve the problem if the goal towers are stacked horizontally instead of vertically. “On the ground, we can pick up and move a block regardless of whether it is sandwiched between blocks. And if we leave space between blocks we can insert a block without first shifting the others around. Hence, we can save many steps by solving the problem on the ground first (p. 34).”

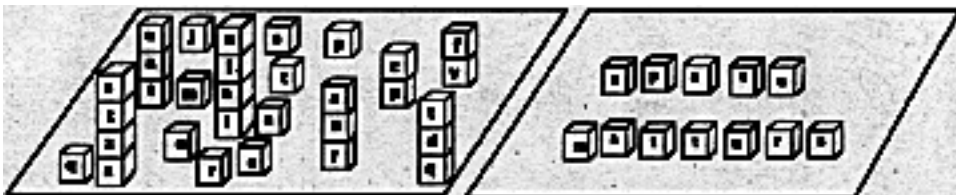


Figure 2.3. Classic AI planning problem in which one must build two towers that spell out SPACE MATTERS. In (a) the traditional approach of solving the problem using vertical stacks versus (b) which solves the problem horizontally. The horizontal approach allows for easy re-orderings without having to balance the blocks.

Kirsh also states that experts do very little planning: “experts find sufficient cues in the situation to trigger a known rule without halting the activity in order to

consciously and analytically take stock of the situation and reason or deliberate about a situation. "(p. 37). In fact, experts often build *environmental damping factors* that serve to decrease the variability of an environment. Kirsh further explains that users "seed" an environment with attention-getting objects or structures. The objects are then used to reduce perceived choice as well as bias the order in which actions are taken.

Affordances [Gibson, 1979] tell you *how* to use something. We often talk about affordances related to objects (e.g., the finger handles on a pair of scissors guide you how to hold and operate them). However, affordances are also applicable to *situations*. The perceived action set at any given moment in time is sensitive to the properties of the situation, specifically spatial arrangement. This action set can be biased by hiding affordances (constraining) or by highlighting affordances (cueing). Arrangements can also highlight the *obvious* things to do or the *opportunistic* things to do. For opportunistic possibility, it is desirable to leave around a certain amount of clutter to increase the chance of getting something for nothing. However, this comes at a potential cost of the clutter getting in the way of the primary task. Finally, space can be set up to provide a temporal order of action.

Spatial arrangements that involve clustering and structuring can simplify perception which can make it easier to (1) keep track of where things are, (2) notice the relevant affordances (3) recognize the availability of actions and (4) monitor the current state. Gestalt theory specifies that there are other factors besides proximity that trigger clustering such as: "how similar the items are (similarity), whether the items move together (common fate), whether they fit into a smooth, continuous line (good continuity), whether they can be seen as a closed form, and whether they stand out together against a background (surroundedness) [Kirsh, p. 57]."

Creative activity makes heavy use of external representations because "in the discovery phase, one wants to note as many possible extensions and variations to one's ideas as possible. This is easier if the representations are externalized (p. 64)." Furthermore, because internal computation involves the generate-and-test phases, the use of physical space and externalized actions may be well suited for this type of exploration.

There are a variety of strategies for the spatial arrangement of control devices in the workspace [Sanders and McCormick, 1987]. Such arrangements include:

- *importance* - arrange items based on the degree at which the activity of using the item is essential to the overall goals of the user. This is very much a subjective determination.
- *frequency-of-use* - arrange items which are more commonly used together near each other. For example, place a stapler near a photocopying machine.
- *functional* - group items based on their functional use (i.e., group all animation controls together or group all file operations together).
- *sequence-of-use* - arrange items based on the pattern of usage such that users can take advantage of these patterns (e.g., place a sequence of items in a row).

Fowler et. al [1968] conducted a study using these 4 styles of arrangements. They designed multiple control and display panel layouts for each of the 4 spatial arrangement strategies. The "sequence of use" layout strategy showed significantly better task completion times for tasks compared to the other three strategies. Layouts based by "function" performed slightly better than both the "importance" and "frequency" layout styles.

We are constantly making intelligent uses of space to aid our cognitive processes. Many of these phenomena are so common to us that they seem obvious or trivial. Yet, these are highly tuned skills that we can take advantage of in human-computer interactions, specifically, for Graspable UIs.

2.3 Things that make us smart

Everyday knowledge, our environment, and the physical characteristics of artifacts can affect the way in which we solve tasks. This is best described in Don Norman's recent book "Things that Make Us Smart" [Norman, 1993]. Specifically, he describes a study conducted by Zhang [1994]. The study was a variation on the classic "Tower's of Hanoi" puzzle. In the study there are three similar puzzles with identical rules but each makes use of different artifacts to solve the puzzle. The first puzzle uses three pegs and three rings, with an initial arrangement as shown in Figure 2.4a. The goal is to reach a final state as shown in Figure 2.4b in which there is one ring per peg and the rings are in descending order by size.

The second and third puzzles are very similar except they use different artifacts (oranges with bowls and coffee cups with plates). For the second puzzle, three different sized oranges are used along with three bowls. The third puzzle uses three plates and three different sized coffee cups. The cups are designed such that the smaller ones fit inside the larger ones. The rules for each move in the three puzzles are as follows:

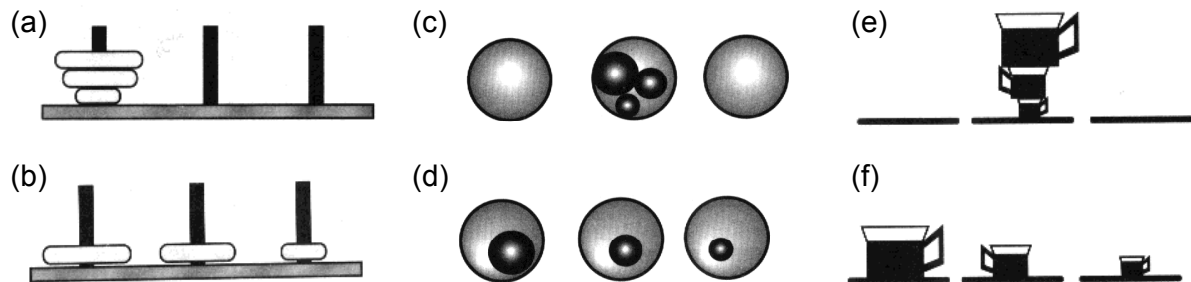


Figure 2.4. Modified Tower of Hanoi puzzles designed by Zhang. Three configurations: rings, oranges and coffee cups. Initial arrangement (a, c, e) and final end state (b, d, f) .

Rule 1: Only one {ring, orange, cup} can be transferred at a time.

Rule 2: A {ring, orange, cup} can only be transferred to a {peg, bowl, plate} on which it will be the largest.

Rule 3: Only the largest {ring, orange, cup} on a {peg, bowl, plate} can be transferred to another {peg, bowl, plate}.

Examining the first puzzle using the rings and pegs, we find that the third rule is redundant for this condition. That is, because the rings are stacked on pegs, the pegs offer physical constraints and force compliance with Rule 3 automatically, assuming the first two rules are followed. So, what happens if we vary the artifacts for the puzzle?

All three puzzles are essentially the same but some are significantly more difficult than others to solve. It was observed that the oranges puzzle took almost 2.5 times as long as the coffee cups puzzle. The oranges puzzle incurred twice as many moves and six times as many errors. These differences are due to the variations in the physical constraints provided in each condition. In the coffee cup condition, Rules 2 and 3 are not necessary since only one cup can fit onto a plate at a given time and the smaller cups cannot be placed on top of larger ones without spilling coffee.

These physical constraints and everyday knowledge aid in solving the problem. The oranges puzzle was more difficult because there were no physical constraints to force compliance with the rules.

External representations add power because the physical structures automatically constrain the actions and interpretations, even though all three rules apply to all the puzzles. Someone programming a computer to solve the task would find all three puzzles to be of equal difficulty and would use the same algorithm to solve all of them. This is because the computer would be unable to take advantage of the physical structures. (Norman, p. 89)

Norman further states that “the more information present in the environment, the less information needs to be maintained within the mind (p. 90).” While conducting the study, he claims that many subjects did not realize that the three puzzles were the same problem. The study “serve as powerful demonstrations of how external representations not only aid in memory and computation but can dramatically affect the way a problem is viewed and the ease with which it can be solved. (Norman, p. 90)”

Norman also generalizes these ideas and explains:

The point is that in the real world, the natural laws of physics allow only the appropriate things to happen. There is no need to compute whether you are walking through a wall: You simply can't do it. In the artificial world of computer simulation, much of the computational effort goes into the part that results from the artificiality of the situation. (p. 150)

The views of Norman are not new, and in fact are embodied in affordance theory (described by Gibson and others) and a new field called Ecological Interface Design.

Ecological Interface Design [Vicente and Rasmussen, 1992] is a relatively new discipline which serves as the intersection of *cognitive engineering* (dealing with problems of measurement and control) and the field of *ecological psychology* (dealing with problems of perception and action). The discipline is based partially on Gibson's affordance theory and visual perception theory [Gibson 1950; 1979]. Three fundamental principles of ecological interface design are:

- Reciprocity of Organism and Environment. We cannot study just the person alone but need to analyze the person and the environment.
- Primacy of Perception. We are much better at perception-action than thinking. Perception is underutilized.
- Start with Analyzing Environment. Identify "what" is going on before "how."

In addition, two applied principles of ecological design are to (1) externalize the constraints that govern a system and (2) support direct perception at the level of the user's discretionary choice.

2.4 Notations that make us smart

We can expand upon Norman's ideas by considering a common concept that "notation is a tool of thought." That is, "thought beget language" and "language beget thought." Notation is tightly intertwined with thought. For example, Figure 2.5 contrasts two long division tasks, one using roman numerals and the other using common decimal notation. Which one is easier to compute?

$$(a) \quad \overline{XXVI} \overline{)CCCLXVII} \qquad (b) \quad \overline{26} \overline{)367}$$

Figure 2.5. Notation as a tool for thought: (a) roman numerals and (b) decimal notation.

As with language, we can argue that *input devices are a notation* (i.e., external representation) or language. Gestures are part of a larger "body language." Thus, body language is a tool of thought and input devices establish through affordances the vocabulary and syntax of that language. Said differently, the use of specific input devices and interaction techniques can regulate, govern or guide the manner in which we use gestures to solve tasks. While developing a gestural vocabulary or input device notation syntax is beyond the scope of this thesis, it is worth of deeper study.

2.5 Sensorimotor integration

Developing visuomotor control requires experience with both the visual and motor systems working together [Brewer, 1993; Schmidt, 1988; Kandel et al., 1993; Warren and Rossano, 1991]. Numerous studies have shown that suppressing the visual system decreases the ability of the motor system [MacKenzie and Iberall, 1994]. One

critical aspect of the control system is an alignment between the visual map and the proprioceptive map. When this mapping cannot occur, performance can degrade.

2.6 Two handed interactions

Having multiple, graspable objects encourages two handed interaction (in Chapter 5 we describe some exploratory studies we conducted which verifies this). We are therefore interested in understanding some theory as to how our two hands interact. Buxton and Myers [1986] as well as others [Bolt and Herranz, 1992] have explored the use of two hands in computer interfaces. More sophisticated two handed interfaces are beginning to emerge such as the Toolglass and Magic lens interface [Bier, et al, 1993]. However, studies have shown that designers must take care in developing interaction techniques that use two hands as poorly designed two handed techniques can be very cumbersome to use [Kabbash, et al, 1994].

Guiard [1987] proposes an interesting framework for bimanual action. He claims that there are three principles that govern the asymmetric behavior of bimanual gestures:

- The non-dominant hand serves as a frame of reference for the dominant hand (e.g., sewing or embroidery, one hand holds the material while the other hand uses a needle).
- The dominant hand is capable of producing finer movements than the non-dominant hand which is capable of coarser movements.
- The non-dominant hand often acts first and is followed by the dominant hand (e.g., the non-dominant hand holds and positions a nail while the dominant hand swings a hammer).

These principles can be applied to designing two handed human-computer interactions. Specifically, we wish to use these principles when developing Graspable UIs when our interactions involve more than one physical object.

2.7 Summary

This chapter has reviewed some of the motor and cognitive psychology literature to provide the underlying theoretical support for workable Graspable user interfaces. First we presented the concept of epistemic motor actions which are specifically performed to reduce internal computation rather than pure pragmatic actions which

are performed only to reach an end goal. The existence of epistemic action suggests that people can use physical objects (i.e., Graspable UIs) and their environment to aid their cognition. Beyond motor activity, people make intelligent use of space in which spatial arrangements of objects serve to simplify choice, perception and internal computation.

The Tower's of Hanoi puzzle presented by Norman and Zhang illustrates the fact that information and rules can be embedded into physical artifacts. Altering the physical artifacts may effect performance and the way in which people solve tasks. Thus, Graspable UIs need to be sensitive to physical form factors. Indeed, input devices can be considered a notation or language where the vocabulary and syntax of the language are established through the affordances of the transducers.

We next review some related research systems and prototypes in Chapter 3. The systems further motivate Graspable UIs as well as exhibit early characteristics and technology in support of Graspable UIs.

Chapter 3: Related research and systems

We now present a collection of research systems, projects and prototypes that exhibit some properties of Graspable UIs. First we summarize the five Graspable UI properties defined in Chapter 1 and present a rating scheme to characterize the related research and systems. Two main research areas of study are then surveyed: computer augmented reality and physical manipulation interfaces.

3.1 Summary of Graspable UI Properties

Figure 3.1 shows the 5 Graspable UI properties as columns while the rows show a level of intensity ranging from low to high.

Generic scale		Graspable UI Properties				
		<i>Space-multiplexing</i>	<i>Concurrency</i>	<i>Physical form</i>	<i>Spatially aware</i>	<i>Spatial reconfigurability</i>
low	○	transient, always reassign, time-multiplexing	1	generic	unaware	permanent location
	◐	3/4 time reassign	occasionally 2			stationary
	◑	1/2 time reassign	2			track
	◒	1/4 time reassign	3			tethered
high	●	permanent never reassign	more than 3	specific	aware	free-ranging (rapid layout)

Figure 3.1. Graspable UI defining properties.

The generic scale to the left of the table (consisting of circles or moons and half-moons) is used to classify the systems. Note that three of the properties (space-multiplex input, concurrency, and spatial reconfigurability) can be defined along a continuous dimension but we have instead defined five discrete units to unify and simplify our comparisons. The remaining two properties (physical form and spatial-awareness) have been defined as binary choices. Many of the dimensions for the five properties can be further refined (i.e., using 10 instead of 5 discrete units). However, the current granularity is sufficient for our initial classification purpose. Further dimension refinement is left as future research.

Given these five Graspable UI properties, we can characterize some common computer interfaces (see Figure 3.2). For the mouse, joystick, and touchscreen devices we assume that they are being used as pointing devices in a standard graphical user interface such as the Macintosh finder or in a specific graphical editor application. The keyboard is mainly used in the context of text entry and issuing hotkey commands. It is interesting to notice that the voice and eye-tracking devices do not exhibit any of the Graspable UI properties.

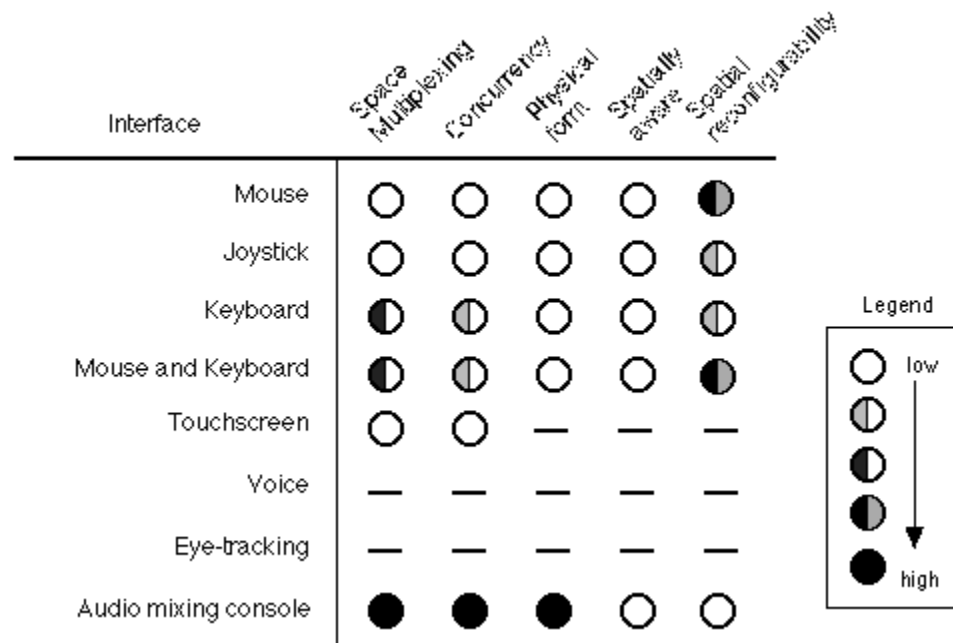


Figure 3.2. Interface systems with Graspable UI design property ratings.

The mouse vs. a fader bank

From Figure 3.2 we notice that the mouse and audio mixing console have quite different Graspable UI property ratings. Consider a detailed example of a sound

graphical equalization application with two different input configurations. The first uses a mouse as an input device the second uses a dedicated linear fader console having a bank of faders or sliders (see Figure 3.3). A typical task may be to adjust the various recording levels for multiple tracks of audio (e.g., percussion, voice, and base tracks) which are merged onto one master track. Both input configurations will ultimately allow the user to solve the tasks at hand but have different affordances.

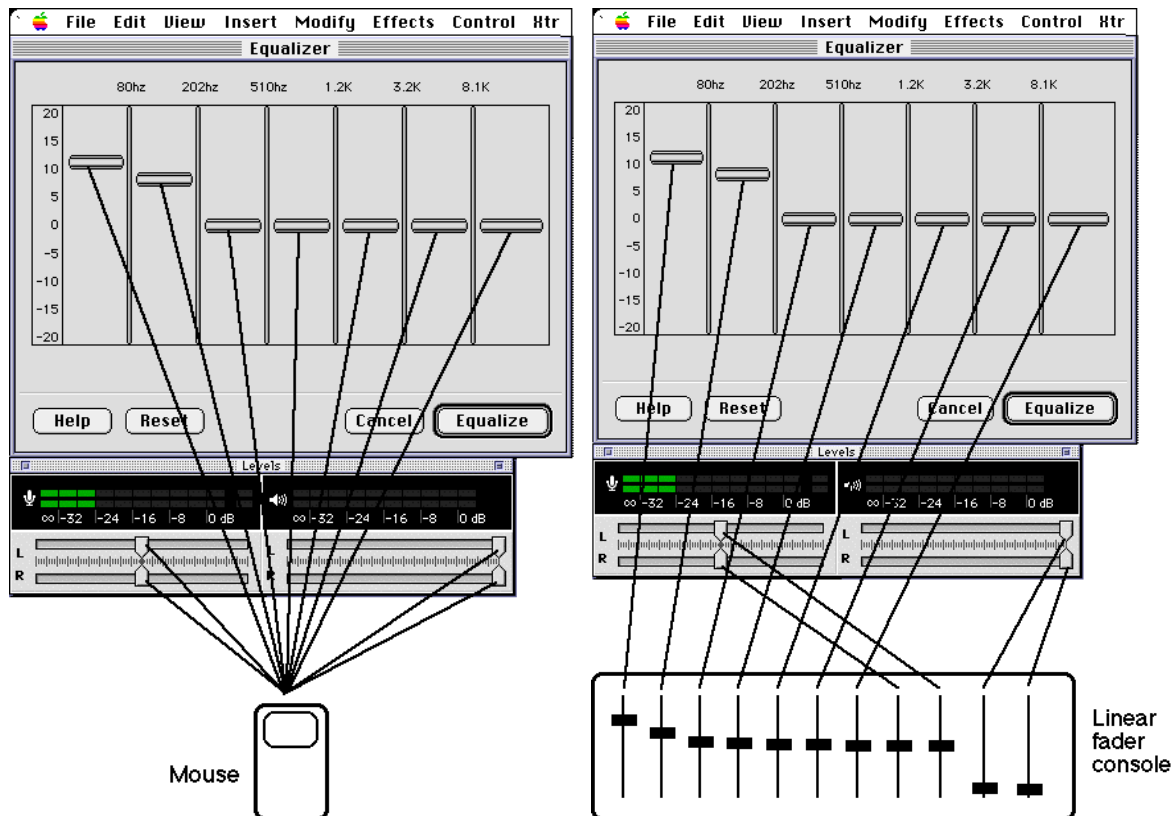


Figure 3.3. Sound graphical equalization application with two input configurations: a mouse and a dedicated linear fader console.

The mouse works as a time-multiplexed input device while the linear fader console is a *space-multiplexed* device. That is, the mouse is a general purpose pointing device which is constantly being attached and detached to logical devices. For the mouse to emulate the linear fader console device, there has to be virtual, graphical interface widgets that corresponded with each of the physical controllers. Moreover, these widgets must be visible in order to be manipulable. In contrast, a linear fader or audio mixing console may have hundreds of physical transducers (e.g., sliders, dials, buttons) each having a *permanently assigned function*. The transducers are, in essence, graspable functions. It is important to note that the linear fader console is not only a

set of input transducers but those same transducers serve as an output display. For example, by examining the position of a set of slider knobs, a user can determine the state of the system. Thus, the system has both space-multiplexed input and output. The audio-mixing console acts as if each graphical widget was *physically instantiated* as a transducer on the fader console device. And on some devices, the sliders are actually motorized so as to give a *dynamic* visual display of state.

Multiple devices are instantiated which allows the user to perform actions not easily done with just the single mouse device. For example, two physical sliders can be associated to two tracks of audio. *Simultaneous adjustments* of the two sliders is possible through the use of both hands. Alternatively, two physical sliders can be *physically coupled* or attached such that when one is moved the other moves as well. This feature can also be implemented quite easily in software where two graphical sliders are logically coupled or attached.

In terms of physical form, the mouse has a generic shape designed to be easily grasped with a precision (i.e., fingertips) or power grip (i.e., palm of hand). In contrast, the physical transducers (e.g., sliders, dials, buttons) of the audio mixing console have a more *specialized form*. For example, a slider is a one-dimensional controller with a fixed range of operation. The physical form suggests and facilitates the functionality it provides.

The physical transducers of the linear fader console cannot be physically moved around. That is, you buy it with a fixed number and layout of transducers. Moreover, if two physical slider knobs are swapped, the system has no way of detecting this. Thus, the transducers have no inherent *spatial awareness*. Moreover, the device does not offer *spatial reconfigurability* due to the fixed transducer layout. In essence, the fader console is a permanent control panel. How could we improve this? Perhaps each transducer could be physically plugged into a peg board. Thus users could design their own control panels for the task at hand. Building in spatial awareness for each controller would mean that the system could track when a controller is moved on the board and if and when it is near other controllers. Furthermore, consider a recording situation in which each instrument or instrument section of an orchestra has a microphone and a recorded audio track. The input transducers (sliders/faders) of a mixing console could be re-arranged in the same spatial layout as the orchestra (e.g., trumpets are next to the percussions, first violinist is in the front, etc.) to organize and facilitate task workflows.

3.2 Computer Augmented Environments

An emerging field known as Computer Augmented Environments [Wellner et al., 1993] reflects a popular trend towards human-computer interaction systems:

Another view of the future of computing is emerging that takes the opposite approach from Virtual Reality. Instead of using computers to enclose people in an artificial world, we can use computers to augment objects in the real world. We can make the environment sensitive with infra-red, optical, sound, video, heat, motion and light detectors, and we can make the environment react to people's needs by updating displays, activating motors, storing data, driving actuators, controls and valves. With see-through displays and projectors, we can create spaces in which everyday objects gain electronic properties without losing their familiar physical properties. Computer Augmented Environments (CAE) merge electronic systems into the physical world instead of attempting to replace it. Our everyday environment is an integral part of these systems; it continues to work as expected, but with new integrated computer functionality [Wellner et al., 1993, p. 26].

Our research into Graspable interfaces adheres to this philosophy. With this definition in mind, we review some systems and projects underway which follow the computer augmented environments philosophy [Rekimoto and Nagao, 1995; Rekimoto 1995; Tani et. al. 1992]. Note that for most of the systems reviewed we start by showing a table containing the associated Graspable UI property ratings.

	Space-mpx	Concur-rency	Physical Form	Spatially aware	Space reconfig
Tabs	○	○	○	●	●
Active badge	●	○	○	●	●

3.2.1 Ubiquitous Computing

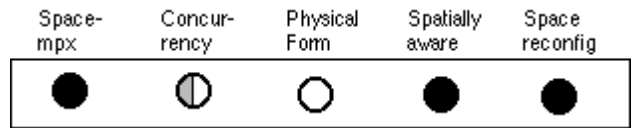
The ubiquitous computing project, initiated by Xerox PARC [Weiser, 1991, 1993], proposes an environment in which people interact with hundreds of nearby wireless, interconnected computers in their everyday environment. The investigation has started by examining three scales of devices: inch (Tab), foot (Pad), yard (Liveboard). The Tab device, modeled after Post-it notes, is a palm-sized computer with a touch sensitive screen which communicates via wireless infra-red to a larger computer infrastructure. Pads, modeled after sheets of paper, are pen-based computers capable of running X-based applications and use radio waves to

communicate to the larger computing infrastructure. The Liveboard, modeled after whiteboards, also can run X-based applications which use wireless pens for drawing on its surface. One of the key ideas here is the relationship between the devices. Applications do not only use one computer but span multiple devices of similar and dissimilar types (e.g., multiple tabs, or applications that use the Liveboard and Tab simultaneously).

Active badges [Want, et al., 1992] are also part of the ubiquitous computing project. These lightweight devices are often attached to people. An infra-red signal is sent from the badge periodically and detected by receivers throughout a building (typically one per room). This allows the computing infrastructure to find people in an environment and bring and configure the computing to them when appropriate.

Buxton et. al. [in press] has extended the idea of ubiquitous computing to include computers (UbiComp) and media spaces (ubiquitous video or UbiVid). In UbiVid there are a wide variety of cameras and monitors in the workspace in different sizes and locations which are sensitive to the relationship between the functions they provide given the space they occupy. The goal is to seamlessly integrate computers, personal spaces and social protocols. In summary, the ubiquitous computing project serves as an example of how computation can be embedded into physical objects and spread throughout a user's environment.

In terms of input style, the Tab device can service many programs and functions and thus has a time-multiplex input scheme. In contrast, since the Active Badge is generally assigned to one object (i.e., a person), it has a dedicated function and thus offers a space-multiplex input scheme. In addition, a user may only be using one active badge, for example, to automatically identify a person and unlock doors in a building. In this application, only one device is being used and there exists no device concurrency. In contrast, a collaborative application that tracks a group of people or the location of many individuals within a building employs a level of device concurrency. It is interesting to note that in this application, a user does not grasp or physically manipulate a set of active badges but instead queries their location. Again, we need to understand the use of a particular input device within the context of an application before we can judge and measure its Graspable UI property ratings.



3.2.2 Digital Desk

The DigitalDesk [Wellner, 1993; Newman and Wellner, 1992] merges our everyday physical desktop with paper documents and electronic documents. A computer display is projected down onto a desk and video cameras pointed at the desk uses image-analysis techniques to sense what the user is doing (see Figure 3.4). The system (1) projects electronic images down onto the desk onto paper documents (2) responds to pen-based interactions as well as with fingers, and (3) can read paper documents placed on the desktop. Using computer vision techniques, the system can also recognize command icons drawn on small pieces of paper. From a Graspable UI perspective, the pen and finger input are similar to that of a touchscreen or digitizing tablet which offer time-multiplexing input. The command icons are more interesting as they are dedicated physical commands which can be spatially arranged and offer space-multiplexing input. Nevertheless, the DigitalDesk is a great example of how well we can merge physical and electronic artifacts, taking advantage of the strengths of both mediums. A related system, the InfoBinder [Sio, 1995], takes advantage of a similar set-up to the DigitalDesk where small physical objects can serve to bind together a physical and virtual component. Each unit, approximately the size of a 2 inch disk, emits a unique infra-red ID to identify itself to the system while being tracked on the desk.

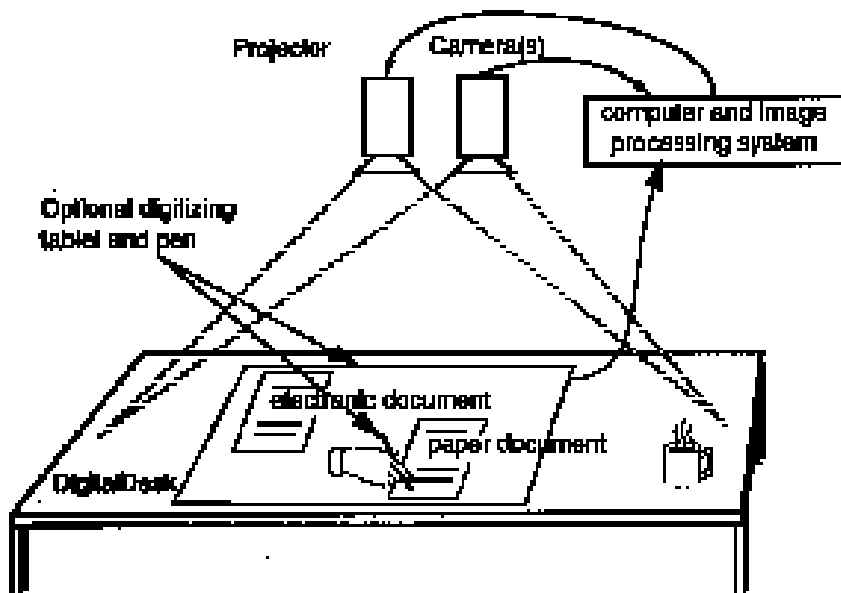
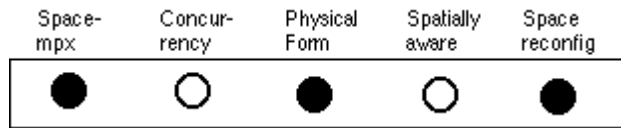
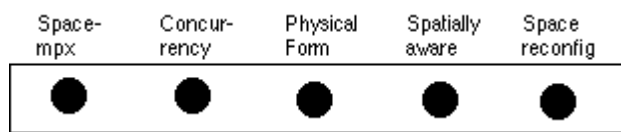


Figure 3.4. DigitalDesk prototype



3.2.3 Mosaic

The Mosaic system [Mackay et al., 1993; Mackay and Pagani, 1994] combines the benefits of paper storyboards and index cards with computer-controlled video. Users manipulate the index cards on the desk and the computer is able to recognize the cards and access their video segment to be played back to the user. The spatial arrangement of the cards can be used to specify the temporal ordering of the video segments. Special buttons and glyphs (e.g., "print" or "play") can be added to the index cards which can also be recognized by the system. Here, once again, the physical artifacts (e.g., index cards) and the dynamic video can be merged, allowing the user to take advantage of physically manipulating the cards representing the video segments while still offering the dynamic function of video access and presentation. Note that although multiple physical cards can be manipulated simultaneously, in the current version of the system, only one card at a time can be "recognized" by the computer vision system. The system cannot currently locate and identify all of the cards on the user's workspace. Instead, the cards must be placed and oriented under a video camera "hot spot" on the desk for recognition. However, it is not difficult to imagine a more sophisticated camera set-up and computer vision techniques which would allow for multiple, orientation-independent, concurrent card recognition on a large workspace.



3.2.4 KARMA

The KARMA system [Feiner et al., 1993] uses a see-through head mounted display to overlay graphics onto physical objects. The prototype accesses an expert system and knowledge base to understand the properties and behaviors of the physical artifacts. One example application is repair work for a laser printer. The overlay graphics provide diagnostic assistance to the operator. In some instances, the graphics allow the user to have "x-ray vision" as internal components of the laser printer can be outlined from the exterior surface. The user's head and components of the laser printer are tracked by multiple 6D devices to provide realistic viewing perspectives when the graphics are overlaid onto the physical objects. A primary issue with this system is the need to accurately align the graphical images to the corresponding physical objects. The KARMA system allows the primary interface to be the physical objects in a user's environment and augments these objects with

virtual information. From a Graspable UI perspective, the KARMA system rates high on all of the properties since the user interface consists of sensing and manipulating real, everyday physical artifacts.

3.3 Physical Manipulation Interfaces

Physical manipulation interfaces emphasize the importance of using real world objects and everyday actions to drive a computer-human interface. The systems below fall into this category as well as using some two handed interaction techniques.

	Space-mpx	Concur-rency	Physical Form	Spatially aware	Space reconfig
Bramble	○	○	○	○	◐
Narrative handles	○	○	○	○	◐

3.3.1 Synthetic physical manipulations

Some complicated virtual user interfaces lack the use of physical objects with their corresponding constraints and affordances which may make many interactions difficult. This is especially true for 3D virtual environments. We present two research systems which attempt to compensate for the lack of physical properties and manipulations within a virtual environment. Both systems use the mouse as an input device.

Bramble -- Differential Manipulation

Virtual objects can take on physical, real-world behavior and follow physical laws by adding constraints to direct manipulation techniques. This has been elegantly shown in Gleicher's work [Gleicher, 1993] in which software constraints have been added to 2D and 3D toolkits to facilitate interaction behavior for users. In his Bramble system, users can attach constraints to objects to guide their behavior in a community of objects. Some of the primitive constraints are go-towards, follow, bound, snap and click.

Narrative Handles

Houde [1992] has explored the concept of using different hand positions as mouse cursors to indicate interaction possibilities (i.e., virtual affordances) for 3D manipulation. Figure 3.5a-d shows the result of this investigation in which four types of hand cursors were defined: (a) ready state, (b) horizontal sliding, (c) grabbing to lift, and (d) two hands pushing in opposite directions for rotation. A bounding box can be generated around objects (Figure 3.5e and 3.5f shows a chair)

with narrative handles attached. These narrative handles consist of hand postures for indicating interaction behaviors. Selecting a narrative handle with the arrow cursor causes the corresponding manipulation. In the absence of the narrative handles, the user clicks on the chair and may be unsure of the resulting action.

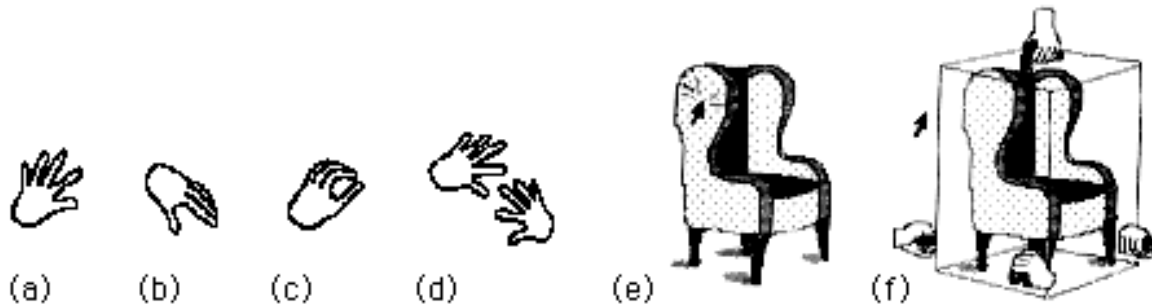
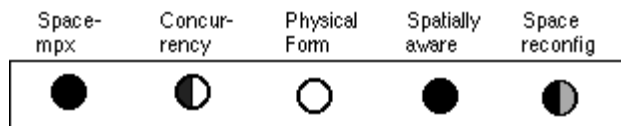


Figure 3.5. A variety of mouse cursors can be used to specify hand postures that correspond to interaction possibilities. A bounding box around the chair object has narrative handles on it along with hand postures for indicating interaction behavior.

Both the Bramble and narrative handles systems attempt to transfer some of the rules, constraints and behaviors of physical object manipulations into the virtual interfaces. This is the opposite approach to Graspable UIs which attempt to leverage off of the physical objects by having systems use both physical and virtual objects for their interaction techniques.



3.3.2 3-Draw

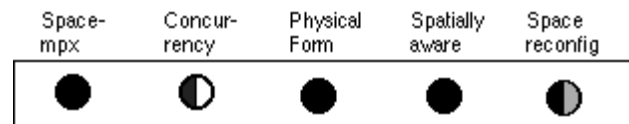
The 3-Draw computer-aided design tool [Sachs et al., 1990] uses a novel two handed input strategy which allows users to easily specify their viewing perspective using one hand and point or draw with a 6D stylus using their dominant hand (see Figure 3.6). It achieves this by first placing a polhemus 6D tracing device on a thin rectangular plate which may be held in the user's non-dominant hand. The plate (approximately the size of the SUN optical mouse pad) acts as a means for specify the user's frame of reference for viewing and manipulating virtual objects. A virtual version of the plate appears on the SGI screen so the user can assimilate the physical and virtual worlds. The stylus, held in the dominant hand, is also tracked in 6D. Consider the following interaction scenario where a user is working in a computer-aided drawing application to edit a car model. The user holds the physical plate in his non-dominant hand with the stylus in his dominant hand. By physically manipulating the plate (i.e., rotating and translating) the user specifies the viewing

perspective of the virtual car model to change in synchronous accordance. The notion is that the user can pretend that the car is sitting on the physical plate. The stylus is used to issue commands or to draw and edit curves on the model.



Figure 3.6. 3-Draw input devices

3-Draw serves as an early example of a Graspable UI for a number of reasons. First, while the plate and stylus are generic physical devices, their functional roles are clearly separated (the plate is responsible for view controls while the stylus is responsible for issuing commands and pointing). This is a space-multiplex design. Secondly, it makes use of concurrency in its interactions (both the plate and stylus can be manipulated at the same time). Finally, both input devices are spatially-aware in that the computer can always sense their physical location.



3.3.3 Passive Interface Props

Following a similar design of the 3-Draw tool, Hinckley has developed the notion of passive real-world interface props for a neurosurgical visualization program (see Figure 3.7) [Hinckley et al., 1994]. Here they want to give the user physical props as a mechanism to manipulate 3D models. Each of the props have a 6D tracker embedded in them. One example they describe provides the user with a doll's head and a small rectangular plate. These two props allow users to select a cutting-plane for a patient's head data (see Figure 3.7). They are striving for interfaces in which the

computer passively observes a natural user dialog in the real world (manipulating physical objects), rather than forcing the user to engage in a contrived dialog in the computer-generated world. The passive interface props differ in the 3-Draw system in that the input devices are truly graspable functions with a space-multiplex input scheme and a very specific physical form.

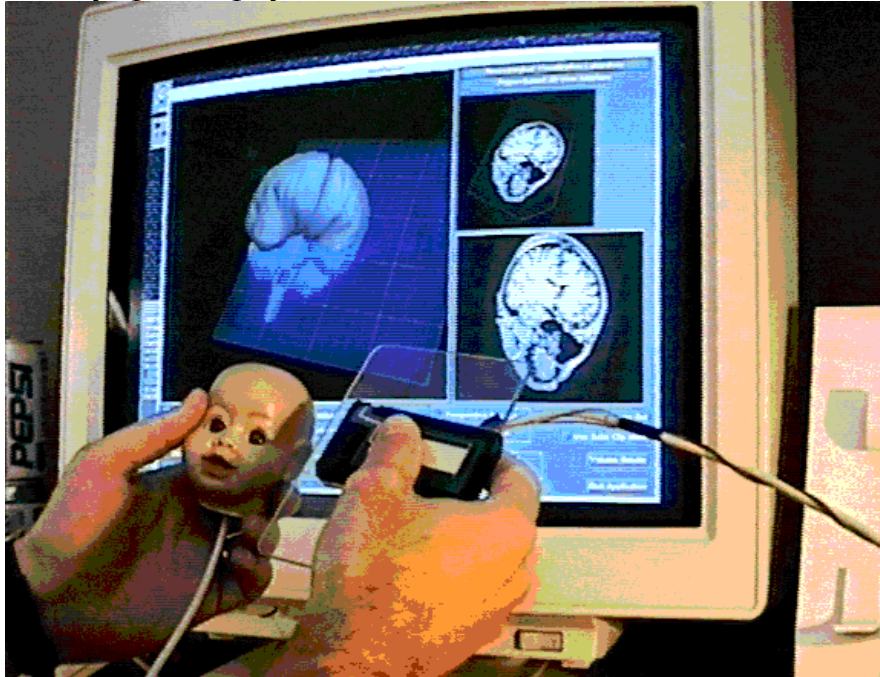
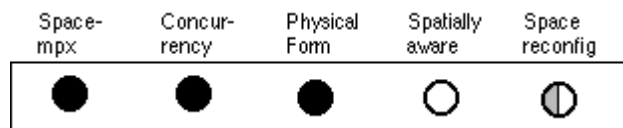


Figure 3.7. Real-world interface props for neurosurgical visualization programs. The user specifies orientation and a cutting plane for a patient's head data using a ball and plate which have 6D trackers embedded in them.

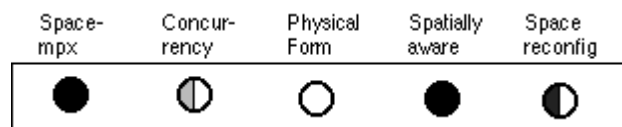


3.3.4 Dinosaur input device

One may argue that the ultimate in physical manipulation interfaces is to build a customized input device that has the same manipulation points as the target virtual object such that all manipulations can be done using the physical input device. This strategy was chosen for creating computer keyframe animations of dinosaur motion for the film Jurassic Park [Knep et. al., 1995]. Here a customized physical skeleton (i.e., armature) was constructed and connected to a graphics workstation to animate an articulated dinosaur figure. The skeleton has sensors attached to each joint; each joint angle is monitored and this data is sent back to the workstation in near real time. This enables animators to manipulate the physical skeleton and have the poses captured by the computer. The authors of the system state that the armature is

precise, fast, compact and easy to use. However, they do point out that some problems do occur. For example, slight discrepancies in the physical skeleton and the virtual skeleton may produce undesirable results (sometimes causing animators to go back to the virtual interface and do some clean-up work). Also, the physical skeleton often cannot support all degrees of joint freedom due to mechanical limitations. In this truly direct manipulation interface, individual joints cannot be easily isolated and separately manipulated without causing surrounding joints to be altered. Nevertheless, the physical manipulations were preferable to the purely virtual manipulations. Finally, this system illustrates the tradeoff between physical and virtual manipulations. Regardless of the medium chosen, the other cannot be ignored.

From a Graspable UI perspective, each joint on the dinosaur can be considered an input device. Thus, the joints offer a space-multiplex input, allow for concurrent manipulation and have a specific physical form. One could argue that the joints are spatially-aware since they collectively describe a spatial pose of the dinosaur. However, consider the fact that the joints are physically strung together and the system has been calibrated for a given joint sequencing. If we swap two similar joints (e.g., right elbow joint with the left elbow joint), the system has no way of detecting this. Moreover, if we swap two dissimilar joints (e.g., right elbow joint with right knee joint), the system will not detect this and even more confusing data will be generated. Thus, spatial reconfigurability of the device is often discouraged due to the extensive recalibration needed.



3.3.5 LegoWall

The LegoWall prototype (developed by Knud Molenbach of Scaitech and LEGO) consists of specially designed blocks that fasten to a wall mounted panel composed of a grid of connectors. The connectors supply power and a means of communication from the bricks to a central processing unit. This central processing unit runs an expert system to help track where the bricks are and what actions are valid (see Figure 3.8).

An example ship scheduling application has been prototyped. The application has objects (e.g., ships) and actions (e.g., print or display schedule). Both the objects and actions are physically instantiated as "bricks." The bricks contain a 64 bit

identification number and come in a variety of sizes and sophistication. Some bricks are very small (e.g., a push button for a scrollbar). Most bricks are designed to be "containers" of information. For example, a container object represents a ship which is traveling between ports. The container is used to access information such as its cargo, crew, schedule, etc. Multiple bricks are instantiated and are permanently assigned a ship or action function. Since the bricks operate on a wall mounted panel composed of a grid of connectors, the bricks and panel collectively define a space-multiplexed input and output system. The wall panel is divided up into spatial regions where a column represents a shipping port. As ships travel to different ports, their corresponding brick is physically moved to the appropriate column. Two or more bricks can be manipulated at the same time.

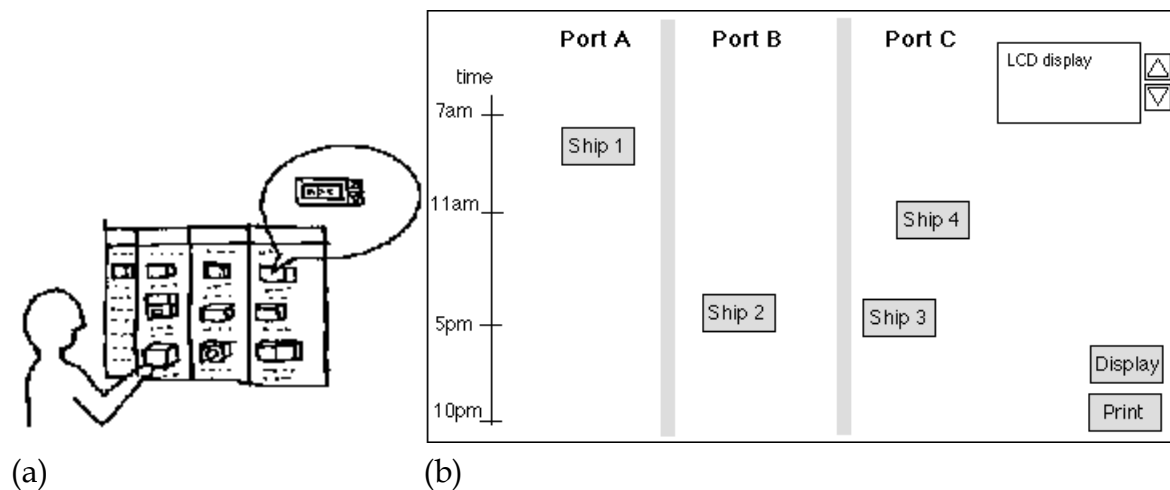


Figure 3.8 LegoWall prototype. Physical bricks can be moved around on a wall mounted panel (a). Ship scheduling application (b).

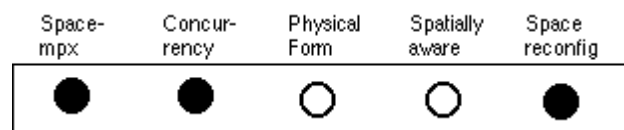
While the LegoWall uses fairly generalized input devices in the shape of a "brick," one could imagine the use of slightly more specialized forms. For example, bricks could have the shape, size and color of model ships. This, potentially, would make it easier to identify individual ships on the board as well as obtain a more detailed gestalt overview of which ships are in which ports.

When the bricks are attached to the grid board, they can be uniquely identified and located on the board. Thus, the bricks are spatially aware devices. Moreover, the proximity of bricks serves to bind an action command operator to operand. For example, placing the "ship" container brick next to the display brick causes the shipping schedule for the given ship to be presented. Positioning the print brick next

to a ship brick and pressing the print button generates a hard copy of that ship's traveling schedule on paper. Because the bricks can be easily moved on or off or within the grid board, the system supports high spatial device reconfigurability.

People can easily point, touch or activate the bricks. Moreover, their everyday skills (touching, pushing, squeezing, moving) and spatial reasoning skills can be used. One of the key ideas is that the user manipulates physical artifacts and the computer monitors these manipulations and interactions and reacts appropriately.

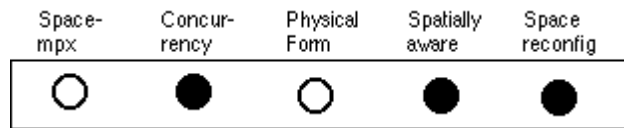
There are several ways of enhancing the current LegoWall design. First, we could project a large display onto the gridboard. This would allow the application to display dynamic output directly on the board. For example, it could select or highlight a set of bricks after a user query (e.g., show me those ships carrying perishable items). Moreover, we could imagine a system that used an alternative input sensing technology that did not require the use of a gridboard. The gridboard requires the discrete and effortful placement of bricks on the board (i.e., users must line up the pegs with the holes in the board). Instead, a smooth surface may be more desirable which allowed the bricks to slide along the surface and afford more rapid placement. These ideas are further explored in the "Bricks" prototype described in Chapter 5.



3.3.6 Behavior Construction Kits

The behavior construction kits being developed at the MIT Media Lab [Resnick, 1993] consist of computerized LEGO pieces with electronic sensors (such as light, temperature, pressure) which can be programmed by a computer (using Lego/Logo) and assembled by users. The idea is to have children construct an assortment of "behaving machines" such as vehicles that move toward the light, or run away when they hear sounds. These LEGO machines can be spread throughout the environment to capture or interact with behaviors of people, animals or other physical objects. In terms of Graspable UI properties, the LEGO pieces offer a space-multiplex input scheme, allow for concurrency and a high degree of spatial reconfigurability. Their physical form is designed to be generic and composable and the pieces are not spatially-aware. Note that a related research system, the Lego Interface Toolkit [Ayers and Zeleznik, 1996], allows users to assemble customized, physical control panels using LEGO components that have been augmented with

sensing devices (e.g., rotation and linear sensors, push buttons). The intent of this physical toolkit is to allow designers to rapidly experiment with developing 3D interaction devices.



3.3.7 Programmable brick

Most recently, Randy Sargent, a member of Resnick's group, has developed a programmable brick [Resnick, 1993]. The brick is a small (approximately 2"x3"x1.5"), battery-powered computer with a wide variety of I/O features (Figure 3.9). Some of the I/O features include a microphone and speaker, infra-red transmitters and receivers, networking capability to connect to host computers, two push buttons and a dial, and a small character-based LCD display (16 characters by 2 lines). It uses a Motorola 6811 microprocessor and contains 128K of non-volatile ROM. These bricks serve as a powerful, portable, computational device which can plug into other lego compatible sensors and motors for detecting and modifying the surrounding environment.

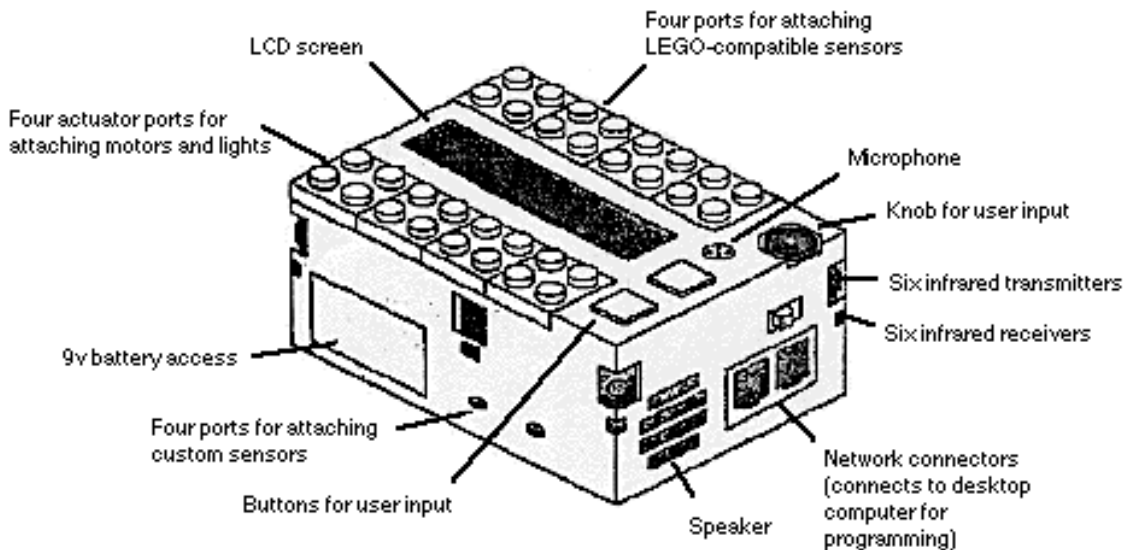
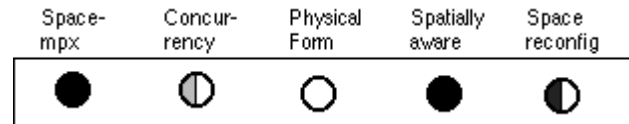


Figure 3.9. Programmable brick consisting of a microprocessor, non-volatile ROM and many multiple I/O ports. Brick measures approximately 2"x3"x1.5".

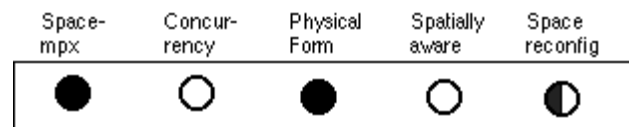
While the programmable brick shares many of the Graspable UI properties as the behavior construction kit, it has significant differences. First, it offers a time-multiplex input scheme since the brick can be used and re-used for a variety of multimedia and computational functionality. The brick is spatially-aware, at a

coarse level, due to its infrared transmitter and receivers; each brick can uniquely identify itself within the range of a given infrared sensor. Finally, while the programmable brick has a very intricate physical form, it is still fairly generic in that the shape, color, and size does not suggest a single function or role.



3.3.8 AlgoBlock

The AlgoBlock system [Suzuki and Kato, 1993] is a set of physical blocks that can be connected to each other to form a program. Each block corresponds to a single Logo-like command in the programming language. Once again, the emphasis is on manipulating physical blocks each with a designated atomic function which can be linked together to compose a more complex program. The system facilitates collaboration by providing simultaneous access and mutual monitoring of each block. The AlgoBlock system shares Graspable UI properties similar to the LegoWall.



3.3.9 Phantom chess

The Phantom electronic chess system dramatically begins to illustrate the concepts of self-propelling bricks, position and motion feedback and system reciprocity. The Phantom system consists of physical chess pieces each having a small magnet embedded within its base. The pieces are placed on a special board that houses a computer controlled mechanical arm underneath the surface. Chess pieces can be grabbed by the computer due to the magnetic attraction and moved around on the playing board by the hidden mechanical arm. The board is also touch sensitive to detect when humans move the chess pieces. This allows a very seamless blend of user control and computer control. At any point, the user can override the computer moves by grabbing a game piece and breaking the magnetic hold. When pieces are "captured" by the computer they are moved off of the main game board and into a designated "parking spot" region situated to the left and right of the board.

The Phantom system has a few interesting features and properties. If the system needs to draw the user's attention to a specific chess piece, the Phantom wiggles or shakes the piece in its current square. This "highlighting" mechanism is very clever and effective. It is used, for example, when the user requests a hint from the computer as to which piece to move next. Another feature the user can request is the

"Valid moves" button. First the user selects a game piece and hits the "Valid move" button. The Phantom will demonstrate what moves are currently valid given the current game board state by moving the piece to each valid final position. During a game, it is often necessary to "jump over" pieces to get to the final square. The system cleverly calculates the path to its target square and shifts those pieces in the way slightly off of their center square to slip the moving piece through the traffic. After the moving piece reaches its final resting square, the Phantom slides any displaced pieces back to its center. In fact, the Phantom keeps a quiet vigil over the pieces and centers them in the square whenever given the opportunity (a neatnick).

In terms of Graspable UI properties, the Phantom offers a space-multiplex input scheme, specialized physical pieces and no concurrency support. The Phantom gives the illusion of having spatial-awareness of the pieces but it actually cannot sense the pieces. Instead, it relies on the pieces starting in their proper positions at the start of a game. It keeps a constant accounting of where each piece should be throughout the game. If a user moves a piece without registering the move with the computer, the Phantom will be unaware of the change. Moreover, game pieces can be swapped (i.e., a pawn for a king) without the Phantom's knowledge. While the pieces may be considered free-ranging, the game board serves as a touch sensitive gridboard where each piece must register its original and final resting position (by applying pressure on these spots).

	Space-mpx	Concur-rency	Physical Form	Spatially aware	Space reconfig
Multidevice tablet	◐	◐	○	●	●
Character devices	●	◐	●	●	●

3.3.10 Wacom Character devices

Wacom Technologies Inc. has explored the concept of having specialized "character devices," what they call electronic stationary, in which devices have a unique shape and a fixed, predefined function associated with it [Fukuzaki, 1993]. The idea is that the form or shape of the device reveals or describes the function it offers. Three character devices were defined: (1) eraser, which functioned to erase electronic ink, (2) ink pot which served to select from a color palette and (3) a file cabinet which brought up a file browser to retrieve and save files (see Figure 3.10).

When the devices are on the tablet, the system is spatially-aware of their location and can identify which device is on the tablet. Note that only one of these character devices can be sensed on the tablet at any given time. However, the devices not on

the tablet do serve a purpose in reminding the user what functionality is available and serve as dedicated graspable functions. In that sense, the devices are being used all of the time. We call this background concurrency. Serving as physical, graspable functions the devices can be easily rearranged in a user's workspace to afford rapid task switching and task workflow. Thus, the system has a high degree of spatial device reconfigurability. The main difference between the character devices and the puck and stylus configuration is that the character devices offer a space-multiplex input design as they have one permanently assigned functional role in the user interface. Moreover, their physical form factor is specialized to suggest and facilitate the functionality it offers.



Figure 3.10 A Wacom tablet with prototype "character devices:" a file cabinet, ink pot and eraser.

Wacom offers digitizing tablets which supports simultaneous use of a stylus and puck device (see Figure 3.11). This Wacom tablet is the first in its class to offer two handed device support. The stylus is pressure sensitive and has a button along the shaft. The puck, has four buttons and a cross-hair pointer. Both the stylus and puck can be sensed and tracked simultaneously. Alias | Wavefront uses this multi-device mode in StudioPaint, a high end drawing program. The puck can be attached to a tool shelf while the stylus can select from the shelf. In a different two-handed mode, the puck can be used to move the drawing canvas around while the stylus remains

in inking mode. Thus, each device is assigned a functional role in the integrated two-handed interaction techniques.



Figure 3.11. Wacom tablet which supports simultaneous use of the stylus and puck devices for two handed interactions.

3.4 Summary

In this chapter we presented a collection of research systems, projects and prototypes that exhibit some properties of Graspable UIs. For every system surveyed, we discussed and rated them using the 5 Graspable UI properties outline in Chapter 1. Figure 3.12 summarizes these ratings.

We survey two main areas: Computer Augmented Environments and physical manipulation interfaces. The augmented environments advocates merging electronic systems into the physical world instead of attempting to replace it (as with immersive virtual reality systems). We reviewed some systems which emphasize blending virtual and physical artifacts into a unified interface (e.g., the Digital Desk, Mosaic, KARMA). Next we described systems that focus on the act of physical manipulations of customized and generic sets of physical artifacts. Many of these systems can be considered early examples of Graspable user interfaces. The most influential and Graspable UI compliant are the Passive Interface Props, LegoWall and the Wacom character devices. These systems have a balance between virtual and physical interface components.

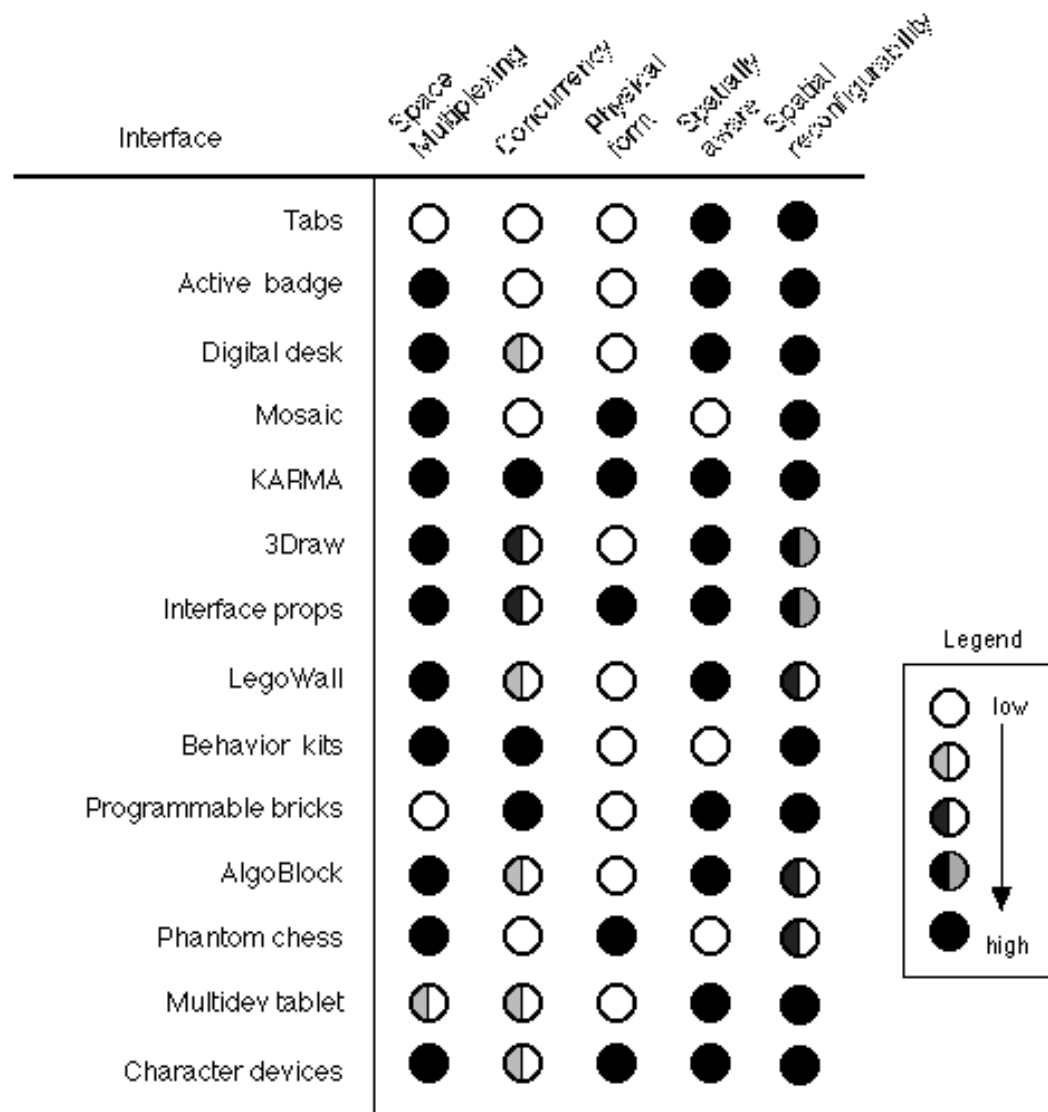


Figure 3.12. Summary of surveyed systems and Graspable UI property ratings.

All of the systems reviewed in this chapter collectively contribute to the refinement and articulation of Graspable user interfaces. In the next chapter (Chapter 4) we go through a detailed example of applying the Graspable UI properties to an existing interface to see the benefits and costs of evolving a GUI to a more Graspable UI.

Chapter 4: A Design Evolution – from GUI to a more Graspable UI

In this chapter we illustrate how the Graspable UI properties are applied to an existing graphical user interface to evolve it into a more Graspable UI. Here we have chosen the context of a commercial software animation program called PowerAnimator(TM) by Alias|Wavefront. Specifically, we chose the task of character keyframe animations. By applying some of the design properties we, in some cases, generate new input devices and interaction techniques.

4.1 Keyframe animation and current GUI design

In character animation, a geometric model of a character is first built. Mathematical expressions which serve as constraints are then added to govern the movement of the character model. To adjust the character, animators can often manipulate the limbs or expressions directly, one at a time. Creating a keyframe animation sequence involves: (1) setting a pose with the character, (2) recording the frame, and (3) advancing to the next unit of time. These three steps are repeated until the sequence is complete. Often this involves going back and adjusting existing keyframes. Another common style is to edit portions of a character (e.g., the head), going through the entire sequence then returning to the beginning and editing another portion (e.g., the arms).

What makes this job challenging is having only one input device (i.e., the mouse) to manipulate each limb or expression of the character. The keyboard is often used to quickly select or cycle through the limbs or expressions which are then manipulated using the mouse.

Four general control categories

A high level task analysis of the animation workflow was conducted. We determined that character keyframe animation tasks have roughly four categories of interaction: (1) selection of objects and commands, (2) 3-D view controls, (3) time

controls and (4) character control. These four control categories are candidates for dedicated devices. It is interesting to note that awareness of the Graspable UI pushes us to search for categories. These categories, in turn, aid us in determining where to deploy devices.

Selection (of objects and commands)

Selection consists of picking command icons, menuing or picking geometric objects which compose the character and the scene (see Figure 4.1). In the current GUI design, selections are made by pointing to an icon or a graphical handle on an object. Selecting a graphical handle is sometimes difficult as a scene may have many handles clustered together. Multiple views of the model (in perspective or orthographic views, or hierarchical component lists) serve to facilitate selecting parts of the model. Keyboard short-cuts (i.e., "hotkeys") also allow the user to issue commands and cycle through selection lists.

3-D View controls

View controls often allow users to manipulate all six degrees of freedom (i.e., tumble, track, and dolly) to change their viewing perspective. The 3D view controls are accessed in a variety of ways including selecting from a set of command icons on a window border or in a tool palette (see Figure 4.1). Since these commands are used so frequently, the view controls are sometimes assigned to a dedicated set of modifier keys on a keyboard. By holding down the keys and using the mouse along with the three mouse buttons, users can activate the necessary view controls.

Time controls

Since animations are heavily time based, VCR-like controls are readily available to go forward or backwards in time with varying time increments. Users click on the corresponding VCR control icon to issue a time control command (see Figure 4.1) or use the timeslider widget. This widget shows a range of keyframes with a graphical bar indicating the current time unit being viewed. Using the mouse, users can drag this bar along the slider to access frames.

Character controls

Finally, character controls are used to set and adjust poses for each keyframe. Sometimes users create customized graphical widgets to facilitate this process. In general, however, users first select a joint on their character and manipulate it directly with the mouse (often specifying a translation or rotation along the X, Y, or

Z directions). The collection of joints and limbs comprise the skeleton of the model which are superimposed (or are "inside") the main volume and surfaces of the character (see Figure 4.1).

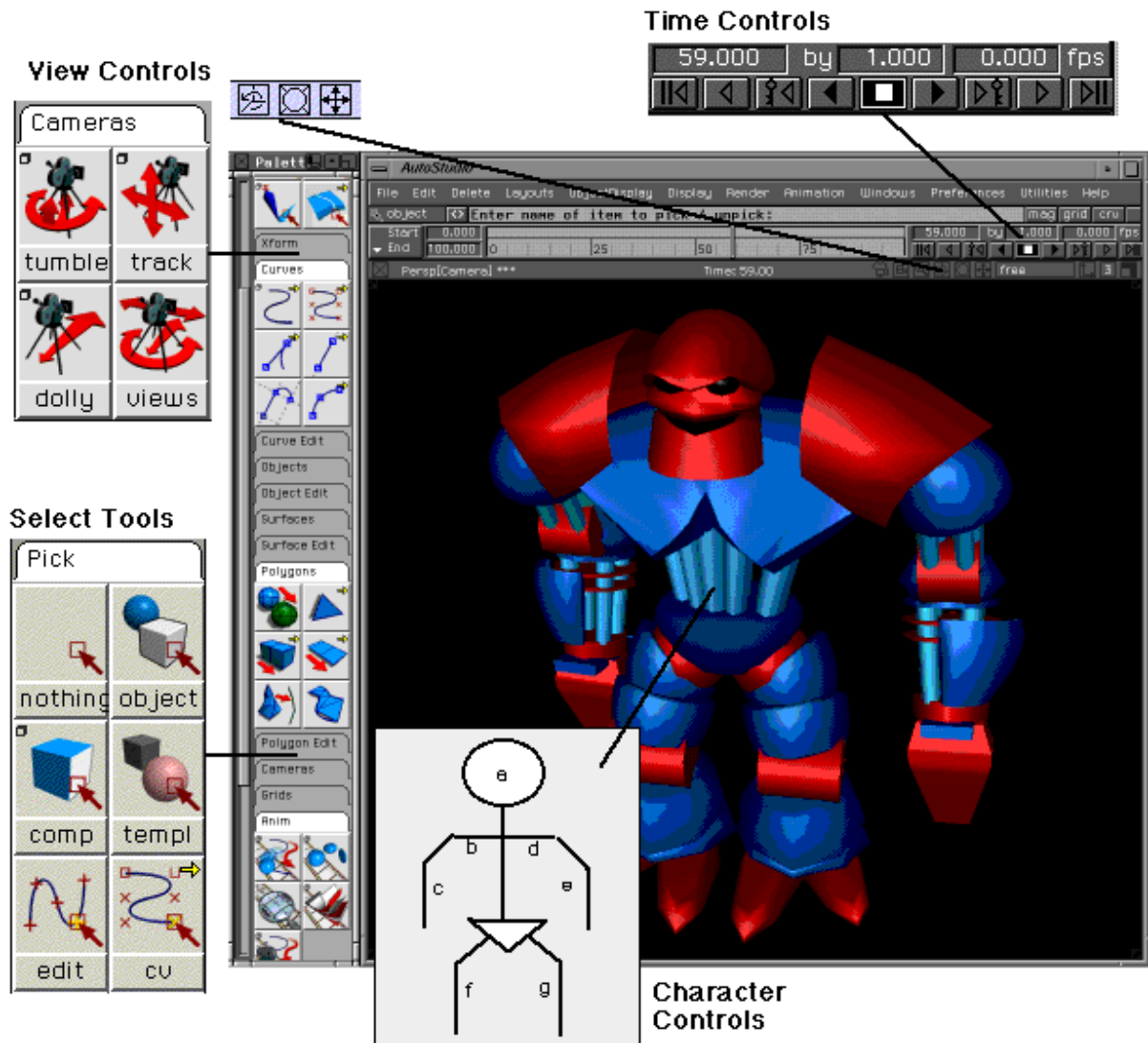


Figure 4.1. Alias|Wavefront's PowerAnimator(TM) package with four main controls for character keyframe animation: time, 3-D view, select and character controls.

4.2 Matching input devices to tasks

Once the control categories are chosen, we must decide which input devices are best suited for the specific tasks. In general, we want to select compatible input devices where the form factor and the manipulation capabilities of the device serve not only to remind the user of the associated functionality but also facilitates the execution of the task. There has been much research on classifying input devices [Card et al.,

1990; Robinett, 1992]. Many input device properties must be considered. While it is beyond the scope of this thesis to give an exhaustive enumeration of input device properties, the following factors illustrate some of the issues to be considered:

- *discrete/continuous action*. Does the task require single discrete actions (e.g., a push-button for issuing a command) or continuous actions (e.g., a slider for selecting a range of values such as audio volume).
- *relative/absolute mapping*. Can the task be performed equally well with input devices that operate using relative or absolute mappings? For example, device clutching is often necessary for relative mappings. Will this interfere with the performance of the task? In addition, absolute device mappings may be more suited for tasks that rely on spatial layouts and arrangements.
- *fixed ranges*. Does the task have natural, pre-defined bounds of operation? For example, a scrollbar in a document has a natural fixed range (i.e., beginning and end of document). Thus a physical linear slider may be suited as its range of operation is physically constrained.
- *number of degrees sensed*. How many degrees of freedom (DoF) does the task require? Does the input device offer extra or too few DoF?
- *device uniqueness*. If a range of devices will be used for a set of tasks, is the device similar to other devices already being used? Using similar devices for related functionality may aid the user to identify a device. However, users must be capable of easily differentiating between similar devices. For example, if a second mouse is added to a system and the two mice sit side by side, users may get confused as to which mouse is attached to which functionality and accidentally acquire the wrong device. This confusion may be diminished by changing the shape, color, or texture of one of the mice or by moving the second mouse to its own space (e.g., the other side of the keyboard).
- *gestural compatibility*. Does the task have gestural requirements in terms of scale of gesture and degree of expression. For example, a gesture can have granularities such as finger, wrist, and arm (i.e., free-hand drawing often benefits by having input devices that support arm-scale gestures such as large digitizing tablets). Degree of gestural expression also has a wide range. For example, a push button does not capture any specifics of *how* the user pressed

the button. In contrast, a stylus on a digitizing tablet can capture a high degree of gestural expression including the user's stroke, the orientation of the stylus as well as the stylus tip pressure.

- *fatigue*. What will be the average time duration of the task? Some input devices are more fatiguing than others. In general, we wish to minimize the onset of fatigue especially for tasks that have fairly long durations and high frequency of use.
- *system pragmatics*. In terms of the pragmatics of adding and maintaining the input device to a system, a myriad of other issues are at play including: system support (e.g., device drivers), power requirements, serial ports, cable requirements, etc.
- *footprint*. How much physical space does the device take-up in the user's workspace?
- *cost*. How expensive is the device?

There are many design tradeoffs when matching and assessing an input device for a given task. Tasks that are more frequently performed, for example, may justify a larger device footprint and higher cost than less frequent tasks. While further analysis is beyond the scope of this thesis; we wanted to provide some design rationale and insight into the process of matching input devices to tasks.

4.3 Stages of Evolution

Stage 0: Status quo

Having defined the basic character animation task and the set of four main controls, we now describe the design stages and the specific input devices used in the evolution of this GUI towards a more Graspable UI. Our starting point reflects the status quo or the current GUI design (see Figure 4.2). That is, the mouse device is commonly used to perform all four categories of tasks (select, view, time, and character controls).

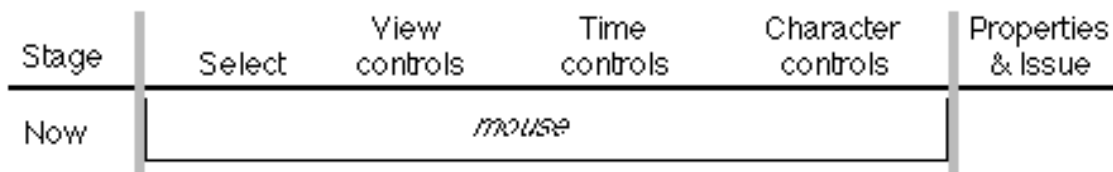


Figure 4.2 Starting design where the mouse is used for all four categories of task.

Stage 1: MIDI sliders for character control

We first apply the space-multiplex input property to the character controls. In our first prototype enhancement, we use the MIDI interface [Chamberlin, 1985] to drive multiple interaction input channels for controlling a character. Each limb or expression of the model can be assigned a generic interaction input channel. This is achieved by simply selecting the graphical object and, using a dialog box, assigning it a channel number and optional scale and offset values. Once assigned, these interaction input channels can be driven by one or more MIDI devices which are routed through the serial port on the SGI workstation. For example, we have a MIDI device which contains 8 physical sliders (see Figure 4.3a). Each of the sliders are assigned to one of the interaction input channels. In our example, a robot model can be controlled using the MIDI sliders (see Figure 4.3b). The slider assignment is done dynamically during run-time by using a "plug-in" architecture supported by PowerAnimator(TM). Note that for convenience, each physical slider is labeled as to its limb or expression association using a pen and masking tape.

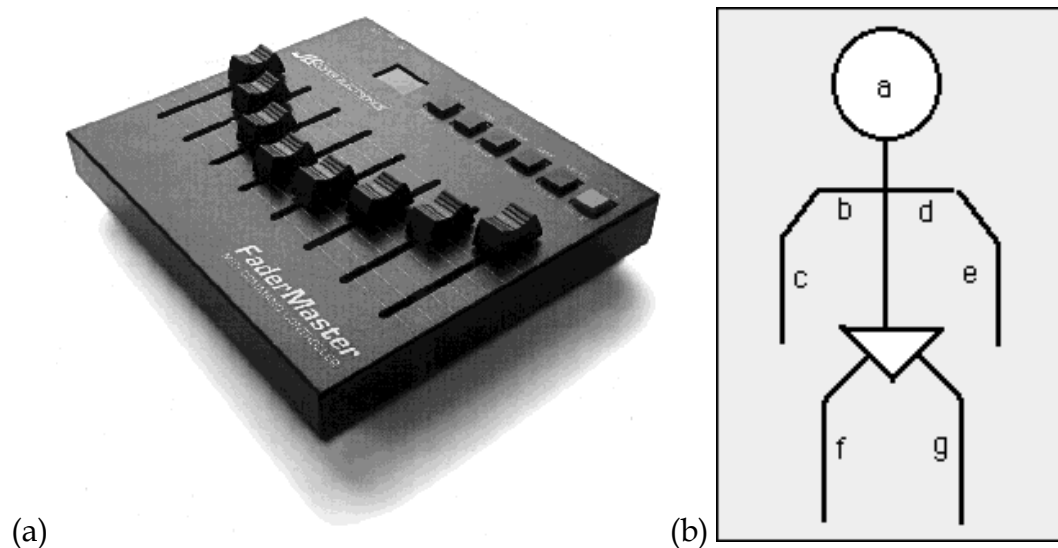


Figure 4.3 MIDI sliders (a) used to control the robot character. Each slider is assigned a limb to control the robot model character (b).

There are a number of advantages to the MIDI approach. First, the MIDI sliders serve as a space-multiplexed input scheme where each slider is a graspable function. That is, each slider has a permanent selection and attachment to the graphical widget used to move a particular limb on the robot model. The net result is the ability to manipulate multiple limbs or expressions simultaneously using the physical MIDI sliders. In contrast, using the traditional GUI approach of using only

a mouse, users have to select and attach to the graphical widget *every* time they wish to perform a manipulation on a new limb. Figure 4.4 shows the first stage in our design evolution.

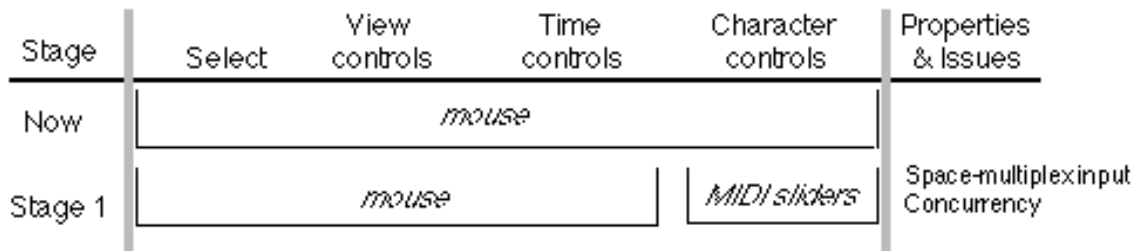


Figure 4.4. First stage in design evolution. Space-multiplex input was applied to character controls. This affords rapid and concurrent limb control.

Stage 2: Space ball for time and view controls

In this stage of evolution we consider using a (quasi) isometric, 6 DoF input device, the space ball, to control both the time and view commands (see Figure 4.5). Here we are again applying the concept of space-multiplexing input by physically instantiated some of the graphical widgets. In terms of physical form, we consider using the spherical shape of the space ball which affords an intuitive navigation metaphor. However, it is difficult to design a set of intuitive time controls based on the spherical device. Figure 4.6 shows the evolution of our design.



Figure 4.5 The space ball six degrees of freedom input device has a generic physical form well suited for view controls but less so for time controls.

Stage	Select	View controls	Time controls	Character controls	Properties & Issues
Now	<i>mouse</i>				
Stage 1	<i>mouse</i>			<i>MIDI sliders</i>	Space-multiplex input Concurrency
Stage 2	<i>mouse</i>	<i>space ball</i>		<i>MIDI sliders</i>	Physical form: generic (time control mappings)

Figure 4.6. Second stage in design evolution.

Stage 3: Space mouse for time and view controls

Instead of using the generalized spherical space ball device, we decided to use the 6 DoF Magellan space mouse (see Figure 4.7a) to operate both the view and time controls. The space mouse is also a (quasi) isometric device but has a more specialized form factor. It is roughly the shape and size of a hockey puck instead of a sphere. The Magellan has a "cap" that the user grasps. The cap rests on a pivot which allows the user to tilt, twist, push, pull and translate the cap. When released, the cap returns back to its initial resting state. Here the cap serves as our graspable functions.

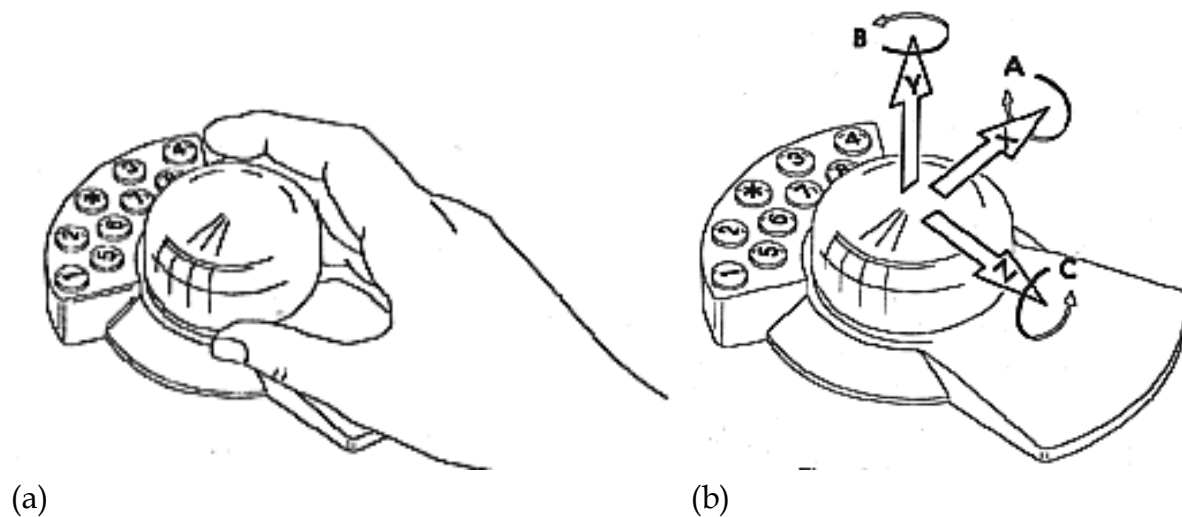


Figure 4.7. A six degrees of freedom input device, (a) the Magellan space mouse and (b) axis labeling for the space mouse.

Six DoF (quasi) isometric input devices such as the Space Ball and Magellan are commonly used to manipulate (translate and rotate) graphical objects in a three dimensional world. Figure 4.7b shows the labeling for each axis and dimension of control that the input device offers (X, Y, Z, A, B, C). Note that positive directions are indicated by the arrows.

Again, the time controls for PowerAnimator(TM) are available through a standard VCR like control panel in the animation package (see Figure 4.8). Once a model and animation is loaded into the program, users click on these controls or the timeslider bar (running across the top of the window) using a mouse to access individual frames or to begin playback.

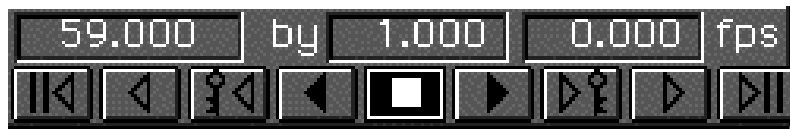


Figure 4.8. VCR-like button control panel for PowerAnimator (from left to right): first frame, retard single frame, previous keyframe, play backwards, stop, play forward, advance to next keyframe, advance single frame and last frame.

As has been stated, the Magellan 6D signals have been typically used to rotate, translate and scale objects in 3D space. The novelty of our design lies in mapping these signals to functions that control the playback and temporal navigation through dynamic (time-based) media such as digital video, audio and, for our design example, animation. The mappings are summarized in Table 4.1.

Dimension	Positive Direction	Negative Direction
X	Last frame	First frame
Y		Stop playback
Z	Mark/set keyframe	Unmark
A	Keyframe retard	Keyframe advance
B	Single/multiple frame retard	Single/multiple frame advance
C	Play backward	Play forward

Table 4.1. Temporal command mappings for 6 DoF space mouse.

These mappings have a strong correlation, or compatibility, with control devices, such as jog and shuttle wheels, typically found in video and audio editing suites. The benefit of the design, therefore, is to enable the 6 DoF technology to support user's existing skills. In short, the new interface will feel familiar to the user, yet the same device can be used in different contexts for other functions, such as the rotate, scale and translate functions which are more commonly associated with the technology. We now discuss the transport control functions summarized in Table 4.1 in more detail.

Single Frame Advance / Retard: A quick twisting of the cap (dimension B in Figure 4.7) to the right/left will advance/decrement by a single time unit (what we call a frame).

Multiple Frame Advance / Retard: Extended or prolonged twists of the cap will result in multiple frame advances/decrements. Sequential frames will be accessed (e.g., 1, 2, 3, 4...) until a threshold time is reached when the frame access jumps from single frame increments to larger units (e.g., 1, 2, 4, 8, 16, ...). This allows for accelerated movement in the time domain.

Single KeyFrame Advance / Retard: Within the animation domain, tilting the cap on the top and bottom (dimension A in Figure 4.7) will advance or retard the current time unit to the next designated "keyframe."

Play Forward / Reverse: Tilting the cap to the right or left (dimension C in Figure 4.7) will cause playback at normal speed (in the forward or backward direction respectively). Rocking the cap back and forth, left and right, enables one to "rock and roll" back and forth smoothly over a particular segment of the data. Animators will find this especially useful when they wish to check how well their current keyframe edits work with the surrounding frames that immediately proceed and follow the current frame.

Stop: Pushing the cap downwards (dimension Y in Figure 4.7) will issue a "stop" playback command.

Go To Beginning / End: To get to the beginning or end frames of a time sequence, users translate the cap to the left or right direction (dimension X in Figure 4.7).

Two Handed Usage

In terms of usage, we expect the 6 DoF input device to be used by the non-dominant hand while a mouse or stylus (or other input device) is being used in the dominant hand. This allows the user to issue time control commands while continuing to work with his dominant hand without switching modes or performing the costly act of traveling to on-screen time-control user interface widgets. Not only is each hand in "home position" for its respective task (typically transport control with the left, selection and marking using the cursor in the right), but these tasks can be performed simultaneously in a two-handed manner. That is, one can select or mark aspects of an animation with the right hand (dominant hand) as its playback is being controlled by the left (non-dominant hand).

Eyes Free "In Hand" Operation

Working with video and animation during playback is a visually demanding task. One's eyes should be concentrating on the data, not some UI widget that enables one to control the data. Yet, the status quo uses graphical icons, representing VCR controls, that not only consume valuable screen real estate, but which also demand distracting visual attention (not to mention additional time) in order to operate. Our technique supports "in hand" immediate eyes free control over the playback. While function keys have been used by others in an attempt to achieve similar purpose, the approach is inferior for at least two reasons: (1) the mapping of keys to function is neither intuitive, obvious nor builds upon the user's existing skills; (2) keys are binary on/off, so this design cannot support speed of playback, for example, being a function of the force of twisting the cap -- something that our implementation does support. Finally, Marking Menus [Kurtenbach, 1993] are another technique that can and has been used for eyes free transport control. The disadvantages of this in comparison with our design are: (1) this is typically done using the dominant right hand, therefore generally not enabling simultaneous marking and selecting, and (2) the technique and marks used are new to the user. They do not build upon existing skills in the same way as our design.

Generality

As an alternative to our approach, one always has the option to interface a "real" jog or shuttle wheel, or a VCR transport control to the computer (see next section). While this will work well, it involves added expense and special purpose hardware. The merit of our approach is that it achieves essentially the same end, using more

general hardware, that is, hardware which can be used for other functions in different contexts, and which is generally supported and available.

This design and time command mapping may benefit users who already work with a jog/shuttle wheel. Specifically, we believe the Magellan 6 DofF input device is well suited for serving as a time controller as the shape of the "cap" is roughly the shape and size of a jog/shuttle wheel. This may suggest functionality to the user and potentially allow for some skill transfer.

One of the difficulties with this design, however, is toggling between the two sets of mappings: time controls and view controls. Users toggle modes by hitting a button on the space mouse (the "*" button). We have found that users will forget what mode they are in and inadvertently issue a command in the wrong mode before realizing their mistake. To compensate for this, we have a mode icon visible on the screen. However, this is not very reliable. A somewhat better solution may be to have the cursor change shape. Nevertheless, this highlights one of the inherent problems of re-using a physical object (input device) for multiple sets of functionality. Here no tactile feedback is possible to let the user know what "mode" they are currently in. Instead they must rely on some visual cue. Figure 4.9 shows the evolution of our design.

Stage	Select	View controls	Time controls	Character controls	Properties & Issues
Now	<i>mouse</i>				
Stage 1	<i>mouse</i>			<i>MIDI sliders</i>	Space-multiplex input Concurrency
Stage 2	<i>mouse</i>	<i>space ball</i>		<i>MIDI sliders</i>	Physical form: generic (time control mappings)
Stage 3	<i>mouse</i>	<i>spacemouse</i>		<i>MIDI sliders</i>	Physical form: specific (view/time mode errors)

Figure 4.9. Third stage in design evolution.

Stage 4: VCR time controls

To alleviate the frequent mode errors in stage 3, we added a second MIDI box (see Figure 4.10) that contains a physical jog/shuttle wheel and a control panel with VCR buttons (e.g., play, stop, record, last/first frame, etc.). With the addition of these physical controls, character animators are able to issue animation and time

commands in a rapid manner without having to use the keyboard or mouse. The physical buttons offer a space-multiplexed design with graspable functions (each button having a persistent attachment to a time function). In this configuration, the space mouse is used exclusively to issue view controls (see Figure 4.11). Here users can arrange their physical devices to accommodate a particular workflow.



Figure 4.10. MIDI jog shuttle wheel and VCR panel buttons

Still, this design can be improved. In many animation applications, there are multiple animation or video sequences to edit. With the current design, users must first select the window in which they wish to issue a time command. Thus, this is a two step process and the time controls do not have a permanent target selection. One obvious solution is to have multiple MIDI VCR control boxes to control multiple sources. This is somewhat wasteful and will clutter up our workspace. In stage 5, we find a better solution.

Stage	Select	View controls	Time controls	Character controls	Properties & Issues
Now	<i>mouse</i>				
Stage 1	<i>mouse</i>			<i>MIDI sliders</i>	Space-multiplex input Concurrency
Stage 2	<i>mouse</i>	<i>space ball</i>		<i>MIDI sliders</i>	Physical form: generic (time control mappings)
Stage 3	<i>mouse</i>	<i>spacemouse</i>		<i>MIDI sliders</i>	Physical form: specific (view/time mode errors)
Stage 4	<i>mouse</i>	<i>space- mouse</i>	<i>VCR cntrls</i>	<i>MIDI sliders</i>	Space-multiplex input (VCR control attachment)

Figure 4.11. Fourth stage in design evolution.

Stage 5: Mobile scrubwheel

The final prototype in stage 5, the mobile scrubwheel, looks at having a more highly specialized physical form to facilitate issuing time commands for multiple, digital video clips or animation sequences. The mobile scrubwheel is a prototype input device developed by Wacom Technologies. We use the device and define interaction techniques to aid the manipulation of temporal digital media. The scrubwheel uses one of the key properties of Graspable UIs to address the mode and attachment problems seen with the jog shuttle wheel. That is, it uses spatial-awareness while preserving the property of skill transfer.

The mobile scrubwheel operates on a Wacom tablet and can sense its position and rotational velocity (see Figure 4.12b). When the mobile scrubwheel is positioned over a window containing temporal media (e.g., video) it can be used both as a cursor as well as for controlling the playback of the media.

The scrubwheel consists of two Wacom sensors and a button on top for user selection. In terms of physical design, it consists of two transparent plastic discs (see Figure 4.12). The outer disk rests on ball bearings housed on the inner disk. By tightening the fastener that holds both disks together, users can vary the tension from locking the disks together to varying degrees of fluid spinning. Ultimately, the transparency could allow users to see "through" the device to the underlying video if the device could operate directly on the computer screen. This, however, is currently not possible.

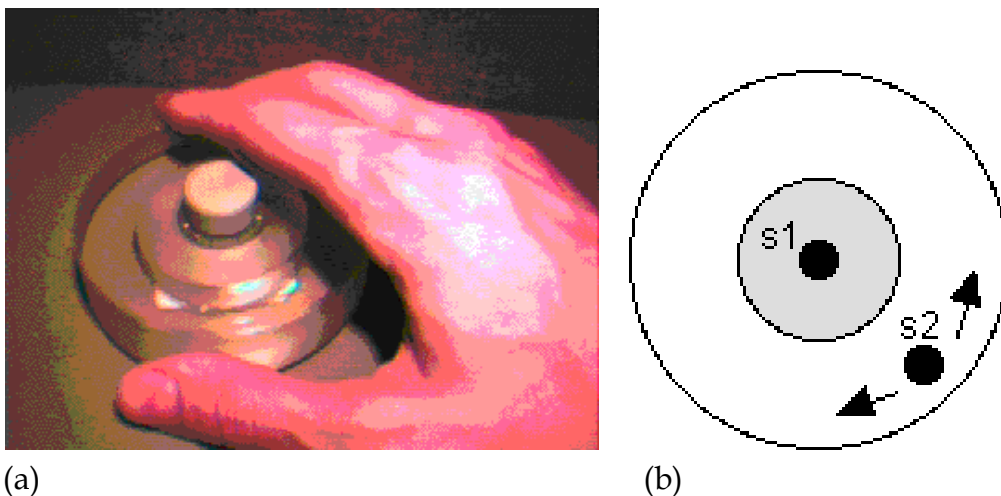


Figure 4.12. Mobile scrubwheel input device (a). Schematic of mobile scrubwheel (b). Sensor s1 specifies location while sensor s2 specifies an angle relative to s1.

Figure 4.13 shows the application design if the scrubwheel operated on the visual surface directly. Here we see three separate windows containing a video clip. Users can position the mobile scrubwheel on top of the window and "scrub" the device which manipulates the underlying video. By scrubbing we mean that the user spins the wheel clockwise (forward) or counterclockwise (reverse) to advance or retard the current video to the next or previous frame. The button is used as a toggle button to stop/resume playback. In our current implementation, the scrubwheel operates on the Wacom tablet (i.e., the input control space and display space are separated).

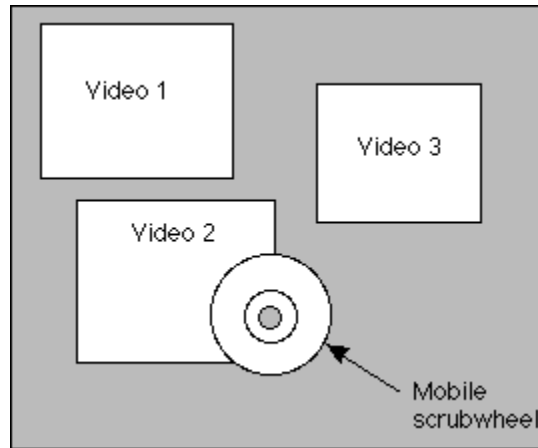


Figure 4.13 Mobile scrubwheel application. The scrubwheel acts on whichever video window it is on top of.

This prototype adheres very strongly to the defining properties for Graspable UIs. Specifically, it physically instantiates part of the user interface (temporal commands) through a customized, specialized physical artifact (scrubwheel), is spatially-aware (its position can be sensed on the Wacom tablet) and is context-sensitive (commands are sent only to the video windows the scrubwheel is on top of).

Nevertheless, we must ask if the same expressiveness could be achieved by using a mouse (to point to the target video) and a stationary scrubwheel (or even a space mouse) to issue commands. This is an area for future research. In any case, we believe the quality of the physical interaction is of paramount importance. The mobile scrubwheel, having a highly specialized physical form, suggests and facilitates the functionality it offers and serves as a very promising example of a Graspable UI. In the future we may wish to explore the idea of using multiple mobile scrubwheels to achieve even better usability. One application may be to use two mobile scrubwheels, one in each hand, to specify "in" and "out" points between two video sources. Again, this is left as future work.

4.4 User Evaluation

Throughout the evolution of the five design stages we elicited user feedback to guide our designs. Since the prototypes were developed at Alias | Wavefront (to gain access to the source code of the PowerAnimator application), we had access to "in-house" expert users of the product. These users have extensive experience creating character keyframe animations (as well as other styles of animation such as motion capture). Many have worked as animators on commercial film productions. The prototypes were informally evaluated by having users spend a small amount of time (e.g., a few minutes) using the system and commenting on their experience. While we were eager to get feedback from any of the expert users, we worked primarily with 3 of the animators.

Our designs were also guided by feedback we received while demonstrating the prototypes at trade shows and conferences. While this feedback was mostly in the form of anecdotal evidence from animators who had a short exposure to the prototypes (e.g., a few minutes), we felt it was very valuable. Many animators and technical directors (professionals who set up animation workstations for animators) described the need to have physical devices to control the user interface and character poses. Much of their rationale stemmed from two beliefs (1) they wanted a way to capture and leverage off of the quality of physical devices and tactile feedback (believing that the mouse and keyboard devices were not sufficient to capture their gestures) and (2) they believed that using physical input devices and artifacts would simplify the interface (e.g., reduce the complexity by reducing the number of functions that are accessible by using only dedicated physical controllers). We were encouraged by how close their requests were aligned with the Graspable UI philosophy.

Stage 1: MIDI sliders for character control

Beyond the parallel activity time-motion gains, users often commented on the "feel" of the physical MIDI sliders and the quality of interaction. This type and quality of interaction cannot be easily achieved using the graphical widgets and a mouse. The tactile feedback gives users additional information. For instance, users can feel when they are at the min or max values as the physical sliders are constrained to operate along a track that has a fixed length. We also had animators try a set of touch-sensitive MIDI sliders instead of physical faders. They much preferred the touch of the physical sliders. Moreover, quite often the graphical widgets or expressions

consume valuable screen space. Using the MIDI approach the graphical widgets are not needed. Finally, because MIDI is a very popular and mature standard, many devices (often very economically priced) are available and can be easily added or chained to other MIDI devices.

Stage 2: Space ball for time and view controls

We did not gather much user feedback from using the Space mouse as the alternative 6 DoF device, the Magellan Space mouse, offered a more viable solution due to the puck-like shape of the device.

Stage 3: Space mouse for time and view controls

With user feedback, the device temporal mappings for the space mouse went through a few refinements (specifically the acceleration algorithm used in the accelerated playback mode). General improvements to the overall robustness in the design also occurred (e.g., allowing for the user to defined sensitivity settings for each control axis). It has been our experience that users who are familiar with the space mouse device quickly learn the new time control mappings. From the small group we have sampled (approximately 15), we estimate that users become very familiar and fluent with issuing commands within minutes. Users tell us that the up-front learning costs are minimal and are easily amortized. The prototype has gone through several implementation iterations and is now part of the standard release for Alias' PowerAnimator(TM) product.

Stage 4: VCR time controls

Users commented that they preferred the use of the dedicated physical control buttons to issue time commands (such as play, fast forward, stop). Early prototypes using the scrub wheel device to advance/retard by single frames was not optimized and would often experience a backlog of events. Users noticed this almost immediately, and commented that they preferred using the mouse and the virtual timebar controller widget. This widget allows users to drag the "current frame" bar indicator along a timeline ribbon, affording rapid access to individual frames. While the backlog of events can easily be fixed, more refinements need to be performed on the mapping between the physical scrub action and the corresponding temporal adjustments. The scrubwheel seems more suited for accessing "nearby" frames (e.g., advancing/retarding up to 10 frames) than for providing random-access to the temporal media due to its linear (i.e., "spinning") nature.

Stage 5: Mobile scrubwheel

For the mobile scrubwheel device users anticipated the functionality associated with the device. This, we believe, is attributed to the shape and manipulation characteristics of the device. Some users would actually use both hands to operate the device: one to hold the center and the other to make it spin. Initially, we thought users would only spin it slowly. Thus, the movie would be stopped and they would use the device only to get to nearby frames. However, to our surprise, while the movie was being played, users would operate the scrubwheel, sometimes spinning it very, very fast. If the movie was playing forward, users could spin the scrubwheel backwards (counterclockwise) at the proper rate to temporarily pause the video or even go backwards a few frames. When the scrubwheel started to slow down, the movie would again proceed forward but at a reduced speed. Normal playback would occur when the scrubwheel stopped spinning. Also, if the movie was playing forward, spinning the scrubwheel forwards (clockwise) would fast forward the video. Users commented that they enjoyed this interaction. The button on the device was not very usable. It was positioned at the outer edge of the scrubwheel so users would often have to hunt to find it. A better design would place the button in the center of the wheel.

All of the 6 stages of design have shown how we can evolve an existing GUI application into a more Graspable UI. Figure 4.14 shows all 5 of the evolutionary design stages that we undertook to make the process of character keyframe animation into a more Graspable UI design.

Stage	Select	View controls	Time controls	Character controls	Properties & Issues
Now	<i>mouse</i>				
Stage 1	<i>mouse</i>			<i>MIDI sliders</i>	Space-multiplex input Concurrency
Stage 2	<i>mouse</i>	<i>space ball</i>		<i>MIDI sliders</i>	Physical form: generic (time control mappings)
Stage 3	<i>mouse</i>	<i>spacemouse</i>		<i>MIDI sliders</i>	Physical form: specific (viewtime mode errors)
Stage 4	<i>mouse</i>	<i>space-mouse</i>	<i>VCR cntrls</i>	<i>MIDI sliders</i>	Space-multiplex input (VCR control attachment)
Stage 5	<i>mouse</i>	<i>space-mouse</i>	<i>mobile scrubwheel</i>	<i>MIDI sliders</i>	Spatially-aware, Spatial reconfigurability

Figure 4.14. All five stages of our design evolution.

Finally note that this process of evolution is not terminal. For example, we can imagine improving the design further by making the MIDI sliders spatially-aware. Having a smaller form factor, the slider unit could operate on top of the Wacom digitizing tablet. Here the tablet can have designated regions marked for specific characters or pieces of geometry. When the slider unit is moved to a region, the sliders are automatically attached to the character's geometry and are ready for manipulation.

4.5 Summary

In this chapter we applied the design properties behind Graspable UIs in the context of a commercial software animation program. The designs at each stage were prototyped and informally evaluated. A byproduct of this design process was the development of novel interaction techniques (e.g., the time control mappings for the space mouse 6 DoF input device and the mobile scrubwheel). Chapter 5 further illustrates the concepts of Graspable UIs by describing a more detailed implementation and case study.

Chapter 5: Bricks

A detailed implementation and case study

This chapter describes a detailed implementation and case study for a specific set of graspable user interfaces which we call "bricks."¹ Here we are investigating a generic dialogue style rather than a single task or application as in Chapter 4. First, a series of exploratory studies was conducted to motivate and investigate some of the brick concepts. Primarily, we wanted to gain insights into the motor-action vocabulary for manipulating hand-scaled input devices on a desktop surface. In addition, we wanted to contrast the differences between physical and virtual object manipulations.

The bricks design explores the use of generic physical objects as handles to virtual objects. In our initial configuration, the physical input control space and virtual display space are superimposed. We argue that the affordances of the physical handles are inherently richer than what virtual handles afford through conventionally direct manipulation techniques. After outlining the basic bricks design, we describe a working prototype system and a sample drawing application called GraspDraw. Both of these efforts aid exploring interaction issues in the context of a simple working application.

Next, we apply the bricks design to a second application, that of curve editing. This prototype builds upon the first bricks prototype and investigates a specific interaction task within the context of a more robust commercial application. We describe a prototype system and the efforts and issues involved in transferring the ideas into a commercial application.

¹ Note that portions of this chapter appear in [Fitzmaurice et al., 1995].

Lastly, we present the “flipbrick” prototype which is a new input device specifically designed to economize the bricks design. That is, we would like to have multiple bricks available but want to minimize the physical clutter and at the same time cluster similar functionality. This prototype is described in terms of using a flipbrick to represent menu choices as well as rapid task switch.

Throughout this case study we set out to gain further design experience with the 5 Graspable UI design properties of (1) space-multiplex input and output, (2) concurrency, (3) physical form (weak general vs. strong specific), (4) spatially-aware devices and (5) spatial device reconfigurability.

5.1 Exploratory studies

The first two studies (LEGO separation task and domino sorting task) examine a user’s range of grabbing and gesturing behavior for tasks that require rapid hand movements and agile finger control for object manipulations. The next two studies (physical manipulation of a stretchable square; comparison using MacDraw application) introduces the concepts of gestural “chunking and phrasing,” issues of interactions [Buxton, 1986] comparing physical and virtual interfaces. The following study investigates V-Blocks, a simple virtual block construction kit application. Here we contrast the differences between physical and virtual manipulations. Next, a curve matching study is described that explores the use of a highly specialized input device. Finally, the last study uses a visual prototyping tool to simulate a range of interaction behaviors to be considered in future, more robust prototypes.

5.1.1 LEGO separation task

In the first exploratory study we asked subjects to perform a simple sorting task as quickly as possible. The basic idea was to get a sense of the performance characteristics and a range of behavior people exhibit while performing a task that warrants rapid hand movements and agile finger control for object manipulation. Four subjects were presented with a large pile of colored LEGO bricks on a table and were asked to separate them into piles by color as quickly as possible (see Figure 5.1).

We observed rapid hand movements and a high degree of parallelism in terms of the use of two hands throughout the task. A very rich gestural vocabulary was exhibited. For instance, a subject's hands and arms would cross during the task. Subjects would sometimes slide instead of pick-up and drop the bricks. Multiple

bricks were moved at the same time. Occasionally a hand was used as a "bulldozer" to form groups or to move a set of bricks at the same time (see Figure 5.2). The task allowed subjects to perform *imprecise* actions and interactions. That is, they could use mostly ballistic actions throughout the task and the system allowed for imprecise and incomplete specifications (e.g., "put this brick in that pile," which does not require a precise (x, y) position specification). Finally, we noticed that users would enlarge their workspace to be roughly the range of their arms' reach.



Figure 5.1. The image shows a subject performing the LEGO separation task.



Figure 5.2. Lego separation task. Separate the lego bricks by color. Here we see a subject using their hand as a "bulldozer" to move a group of bricks.

It is also interesting to note that before the task started, all subjects instantly began “playing” with the LEGO bricks once they were in front of them. That is, they would compose larger structures and design more complex objects. This suggests a few points. First, the bricks afford the act of composing. Secondly, people may enjoy the act of physically manipulating the LEGO bricks. That is, there is an inherent “fun factor” with these particular artifacts.

5.1.2 Domino sorting task

The second exploratory study asked four subjects to place dominos on a sheet of paper in descending sorted order. Here the domino bricks have very similar Graspable UI properties compared to the LEGO bricks except for the obvious shape, color and texture differences. Initially, the dominos were randomly placed on a tabletop and subjects could use the entire work surface. A second condition was run which had the dominos start in a bag. In addition, their tabletop workspace was restricted to the size of a piece of paper (see Figure 5.3). The notion behind restricting the workspace was to see if subjects exhibited different motor behaviors and placement strategies when faced with more workspace constraints.

Once again this sorting task revealed unique interaction properties. Tactile feedback was often used to grab dominos while visually attending to other tasks. The non-dominant hand was often used to reposition and align the dominos into their final resting place while, in parallel, the dominant hand was used to retrieve new dominos.

One of the most useful observations was the confirmation that subjects seemed to inherently know the geometric properties of the bricks and made use of this everyday knowledge in their interactions without prompting. For example, if 5 bricks are side-by-side in a row, subjects knew that applying simultaneous pressure to the left-most and right-most end bricks will cause the entire row of bricks to be moved (see Figure 5.3b).

Finally, in the restricted workspace domino condition we observed one subject taking advantage of the “stackability” of the dominos and occasionally piled similar dominos on top of others to conserve space (see Figure 5.3c). Also, sometimes a subject would use their non-dominant hand as a “clipboard” or temporary buffer while they plan or manipulate other dominos (see Figure 5.3d).

Both the LEGO and domino studies confirm our belief that people will automatically take advantage of their everyday skills and knowledge about physical objects to efficiently manipulate and interact with them. These types of grasping and gesturing behaviors (e.g., “bulldozing,” stacking, composing, sliding, squeezing) should be supported with Graspable UIs that employ similar size and style input devices.

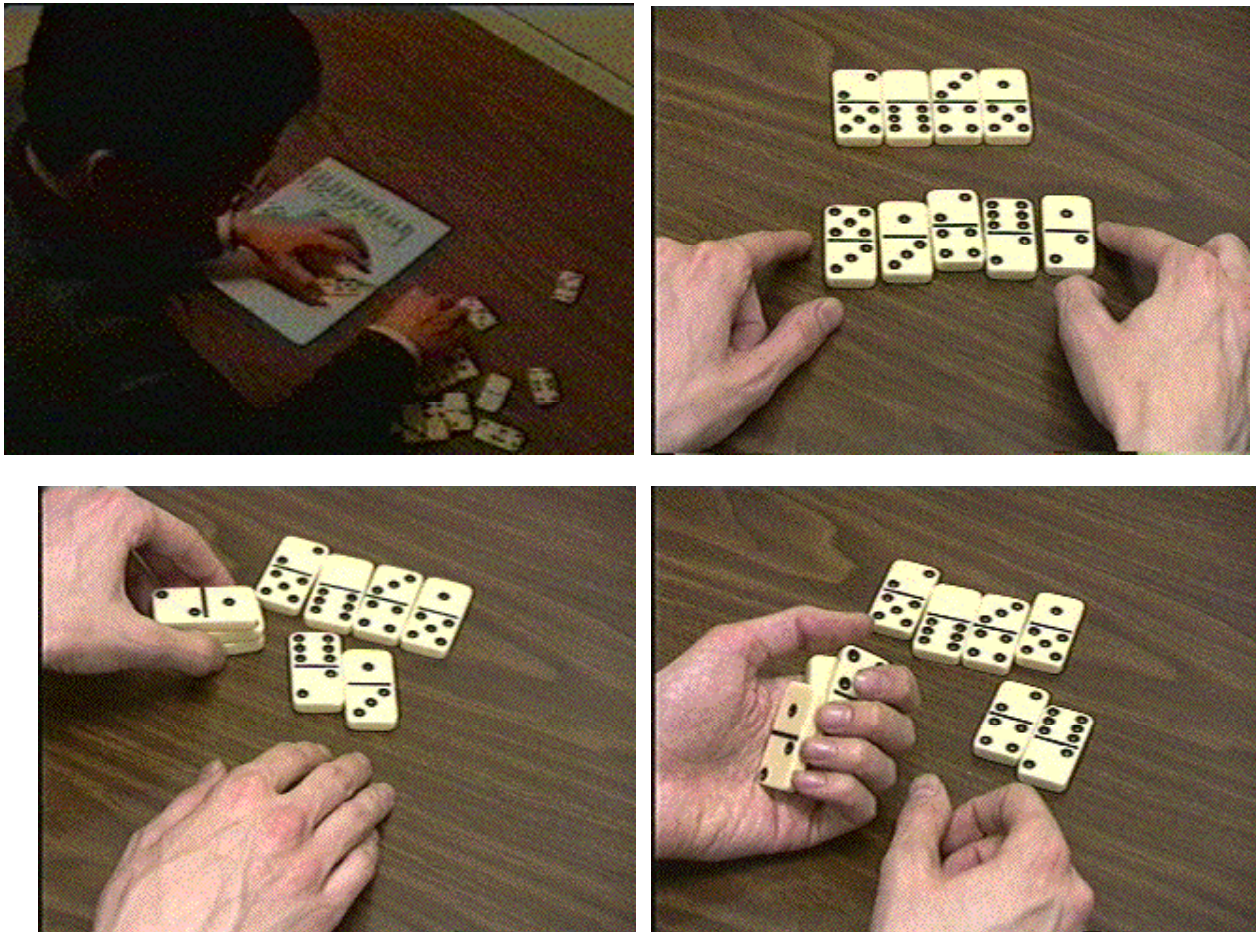


Figure 5.3. (a) domino sorting task. (b) moving a whole row of dominos by applying pressure to the two end dominos (c) stacking dominos to conserve workspace, (d) using the non-dominant hand as a clipboard.

5.1.3 Physical manipulation of a stretchable square

The next three studies (physical manipulation of a stretchable square; comparison using MacDraw application; and V-Blocks: physical & virtual manipulations) are designed to examine the differences between physical and virtual manipulations and the gestural and conceptual “chunking and phrasing” issues of interactions.

Chunking and phrasing are important issues to consider when designing input devices and interaction techniques. The idea of chunking and phrasing [Card, Moran and Newell, 1983] originates in the area of human memory and cognition. While the concept is still being actively researched, it postulates that our memory and cognition uses chunks which are atomic units of operation. These chunks help us cognitively organize our methods of operation. Chunks can be grouped and sequenced into larger units called phrases. A phrase occurs when we need to access our short or long term memory. For example, telephone numbers are likely to be cognitively organized into one phrase consisting of three chunks (e.g., 617-555-4779 instead of, for instance 5 chunks: 61-75-55-47-79). We argue that motor activity also can be thought of as using chunking and phrasing. That is, we want to extend it to gestures and interface design. Phrasing can aid the “ebb and flow of tension in a dialog. It lets us know when a concept is beginning, and when it ends. It tells us when to be attentive, and when to relax [Buxton, 1986].” Many interaction techniques are based on this flow of tension. For example, users click and hold a mouse button to bring up a pop-up menu, drag the mouse while the button is still held down (with tension) and finally release the button to make and execute a menu selection. A key issue in designing interaction techniques is to have the human chunks and phrases match the task at hand. As we shall see in this exploratory study, as well as in our experiments (Chapter 6), a significant performance improvement can be achieved when the user interface is chunked and phrased at the proper granularity.

In this study we wanted to understand the nature of any chunking and phrasing differences between manipulating physical versus virtual objects. A physical "stretchable square" was constructed out of foam core. This square looks like a tray with a one inch rim around each side. Users could expand or collapse the length of the square (see Figure 5.4). We displayed an end position, orientation and scale factor for the physical square and asked subjects to manipulate the square to match the final target as quickly as possible. A variety of cases were tested involving one, two or all three transformation operations (translate, scale, and rotate).

We found that each of the four subjects had a different style of grasping the stretchable square for position and orientation tasks. This served to remind us that physical objects often have a wide variety of ways to grasp and to manipulate them even given natural grasp points. In addition, subjects did not hesitate and were not confounded by trying to plan a grasp strategy. One subject used his dominant hand

to perform the primary manipulation and the non-dominant hand as a breaking mechanism and for finer control.

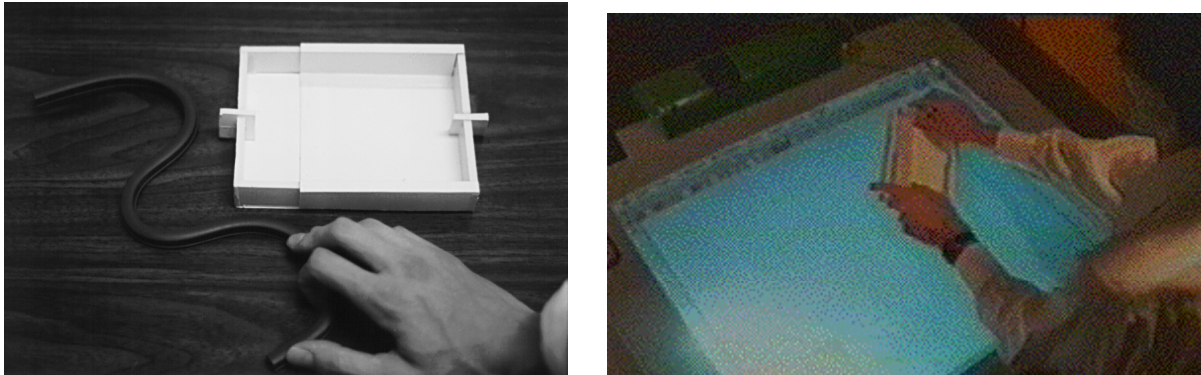


Figure 5.4. (a) Flexible curve and stretchable square, (b) stretchable square in action on the Active Desk.

Perhaps the most salient observation is that users performed the three operations (translation, rotation and scaling) in parallel. That is, as the subjects were translating the square towards its final position, they would also rotate and scale the square at the same time. These atomic operations are combined and chunked together.

5.1.4 Comparison Using MacDraw Application

The same matching tasks were then done using virtual objects and a stylus on the Active Desk. Using the MacDraw II^a program, subjects were asked to move a virtual object on top of a target virtual object matching position, orientation and scale factors (see Figure 5.5).

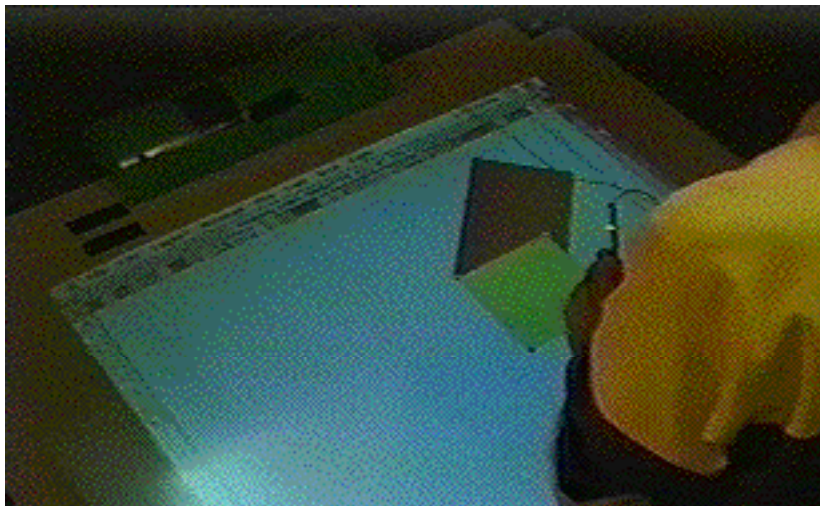


Figure 5.5 Comparable task using MacDraw on the Active desk

We observed that even when we factor out the time needed to switch in and out of rotation mode in MacDraw, task completion time was about an order of magnitude longer than the physical manipulation using the stretchable square. We noticed a "zoom-in" effect to reach the desired end target goal. For example, subjects would first move the object on top of the target. Then they would rotate the object, but often be unable to plan ahead and realize that the center of rotation will cause the object to be displaced. Thus, they often had to perform another translation operation. They would repeat this process until satisfied with a final match.

The MacDraw user interface, and many other interfaces, forces the subject to perform the operations in a strictly sequential manner. While we can become very adept at performing a series of atomic operations in sequence, the interface constrains user interaction behavior. In effect, the interface makes it hard for novices to become experts by not allowing users to exhibit more natural and efficient expressions of specifying atomic operations in parallel. That is, the traditional combination of user interface and input devices do not allow users to chunk and phrase the operations at their desired granularity.

5.1.5 V-Blocks: Physical vs. Virtual Manipulation

Next we compared manipulating physical vs. logical LEGO bricks. Brian Paul at the University of Wisconsin developed Virtual Blocks (V-Blocks), a 3D building blocks simulator in which users create and manipulate 3D structures using virtual LEGO pieces (see Figure 5.6). Users select from a catalog of brick, plate or roof pieces of varying sizes and colors from a dialog box. New pieces can be added to the 3D scene or existing pieces can be manipulated (e.g., rotated or translated). Manipulations are constrained such that the rotations are on 90 degree units and the translations can occur only in orthogonal directions from a given mouse down point. Thus, a sequence of drag, release, drag movements with the mouse are usually needed to position a piece. The constrained manipulations are a bit awkward at first but ultimately aid the overall efficiency of the interactions. Users also have the option of selecting multiple discontinuous pieces and forming groups, as well as deleting, locking, and recoloring pieces.

An informal study was conducted comparing the V-Blocks interface to the physical LEGO bricks in a simple model construction task. Using the physical bricks, we found that users inherently used two hands. This is not possible in V-Blocks which has been designed for one-handed interactions. The second hand was often used to

manipulate the model, to troll for new pieces, or to assemble pieces. With the physical bricks, users get to see the entire range of pieces to choose from since the bricks are often all in a visible pile. Thus, the selection process is much quicker. The fidelity of the interaction in the physical condition is much higher than the virtual condition. In the physical condition, everyday physical constraints are automatically enforced (i.e., you can't place a brick inside another brick). Moreover, the alignment of two bricks often relies more on the tactile than visual perception.

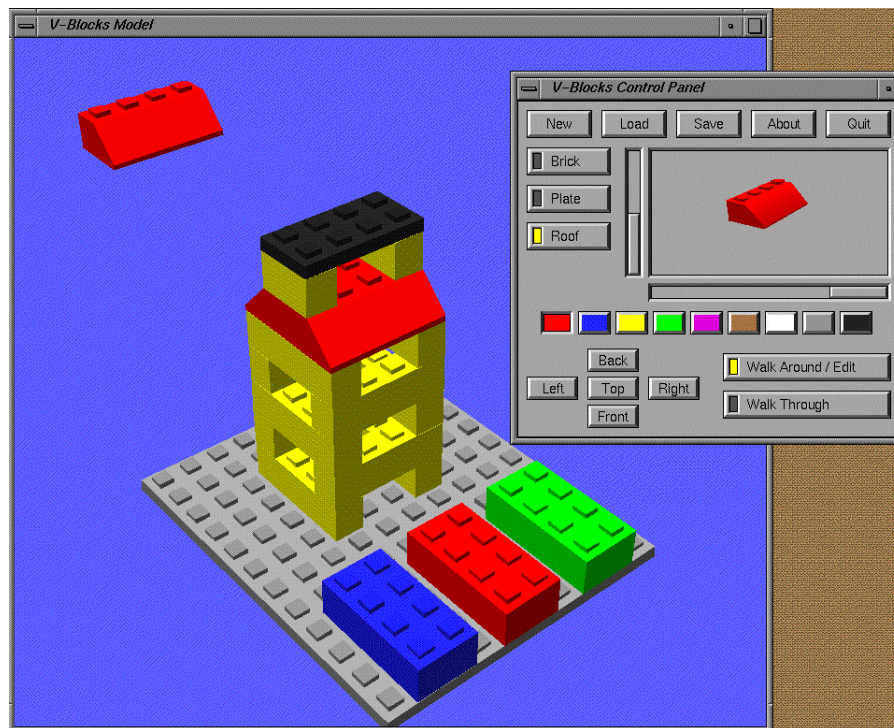


Figure 5.6. Virtual Blocks 3D simulator in which users manipulate virtual LEGO bricks to create models.

In the virtual interface, users are forced to use only their visual perception for placement feedback. However, there are some advantages in the virtual world environment. You can do things that are impossible to do in the physical world. For example, in the virtual interface one can recolor bricks in-place without moving any of the surrounding pieces, or select a set of bricks (even located in different regions) to be manipulated at the same time. Nevertheless, we believe these features would not be performed frequently enough to outweigh the cost of virtual manipulations.

One of the most noticeable differences between the V-Blocks interface and the physical LEGO pieces is the significant learning curve the V-Blocks interface requires users to go through to become adept at navigating the scene as well as selecting and manipulating virtual bricks. In contrast, users are already familiar with

manipulating the LEGO bricks. Therefore, we again argue that human-computer dialogs that use physical objects (i.e., graspable functions) and physical manipulations have the potential to be very efficient.

5.1.6 Curve Matching

Continuing to explore our skills at physical manipulations, we asked four subjects to use a flexible curve (see Figure 5.4a) to match a target shape. The flexible curve is often used in the context of graphic design and has been used in other contexts such as statistics to define a "best fit" curve through a set of points. It consists of a malleable metal surrounded by soft plastic in the shape of a long (18 inch) rod. The inner metal allows the curve to hold its shape once deformed.

We found that users quickly learned and explored the physical properties of the flexible curve and exhibited very expert performance in under a minute. All ten fingers were often used to impart forces and counterforces onto the curve. The palm of the hand was also used to preserve portions of the shape during the curve matching task.

We observed that some subjects would contort their hands and arms before making contact with the flexible curve in anticipation of their interactions. This posturing is a preconceived grasp and manipulation strategy which will allow the user to reach the final target curve shape in one gestural chunk or action. Often the arms, hands and fingers must start in a specific, sometime uncomfortable position.

The flexible curve serves as a highly specialized input device and users take advantage of its unique shape and manipulation properties to facilitate solving the task. It is difficult to imagine how this style of interaction could be expressed easily using a mouse. Here both the minimal learning and the expressive power of the input devices are at play.

5.1.7 Mock-up and simulations

As a final exploration, we mocked-up some sample brick interactions (see Figure 5.7) using a prototyping tool (Macromind Director) and acted them out on the Active Desk. By using a few LEGO bricks as props and creating some basic animations using the prototyping tool, we could quickly visualize what the interactions would look and feel. These sample interactions were video taped and edited. We were able to mock-up many of the primary ideas such as: attaching and detaching bricks from virtual objects; translation and rotation operations using one

brick; using two bricks each attached to separate virtual objects, and finally two bricks attached to a single virtual object to specify stretching and simple deformations.

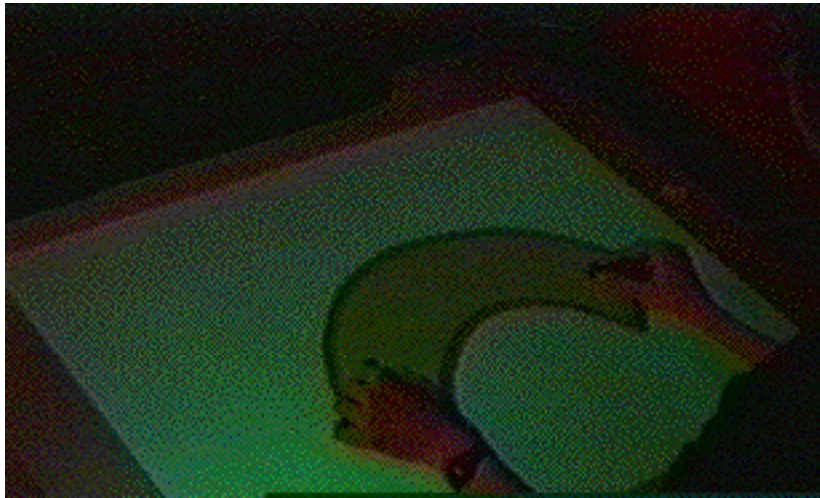


Figure 5.7. Macromind Director simulation of sample brick interactions. Here two bricks are being simultaneously sensed (both translation and orientation) to cause the rectangle to bend.

5.2 Prototype 1: Bricks for drawing

5.2.1 Implementation

After the mock-ups, and design scenarios, we built the bricks prototype to further investigate the Graspable UI concepts. The prototype consists of the Active Desk, a SGI Indigo2 and two Ascension Bird receivers (see Figure 5.8).

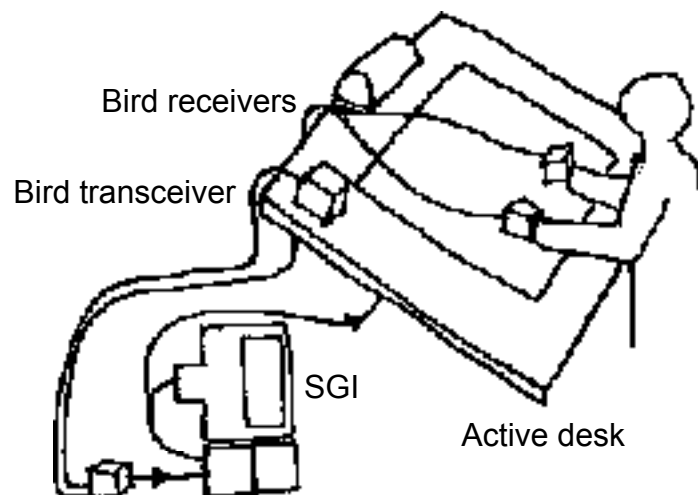


Figure 5.8. Graspable Object prototype environment consisting of Active Desk, SGI workstation and two 6D Bird receivers.

The Active Desk, modeled after a drafting table, has an overall desktop surface dimension roughly 4.5' by 3.0' on a slight 30 degree angle. The projected computer screen inset has a dimension roughly 3' by 2' (see Figure 5.9). A Scriptel transparent digitizing tablet lays on top of the surface and a stylus device may be used for input. The LCD projection display only has a 640x480 resolution so the SGI screen is down converted to an NTSC signal and sent to the LCD display.

To prototype the graspable objects (bricks), we use the Ascension Flock of Birds^a 6D input devices to simulate the graspable objects. That is, each receiver is a small 1 inch cube that constantly sends positional (x, y, and z) and orientation information to the SGI workstation. We currently have a two receiver system, which simulates two active bricks that operate on top of the Active Desk. More receivers can be added to the system but the wires attached to the receivers hinder interactions. Nevertheless, the two receivers offer us an initial means of exploring the design space in a more formal manner.

5.2.2 GraspDraw application

A simple drawing application, GraspDraw, was developed using the bricks prototype to test out some of the interaction techniques mocked-up in the earlier example. The application lets users create objects such as lines, circles, rectangles and triangles (see Figure 5.9). Once created, the objects can be moved, rotated and scaled. GraspDraw is written in C using the GL library on an SGI Indigo2.

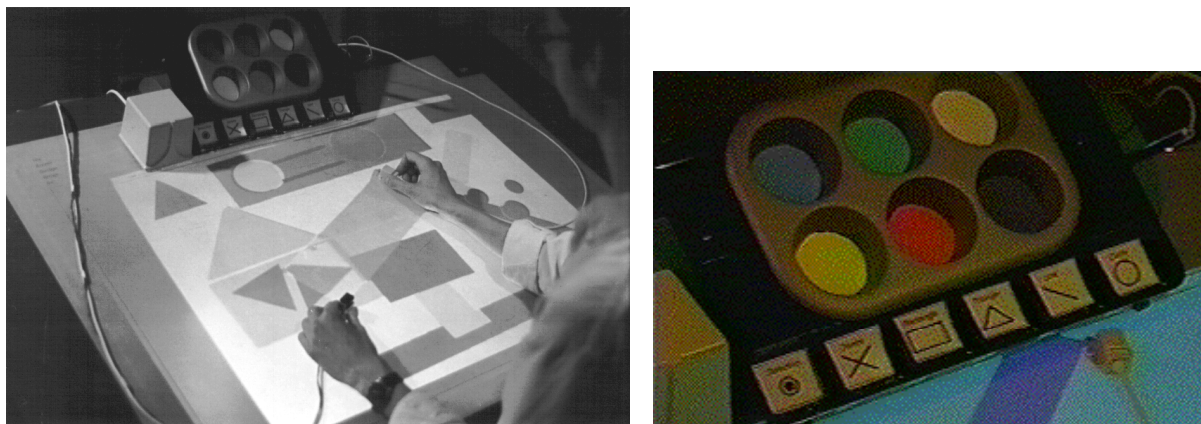


Figure 5.9. (a) GraspDraw application running on the ActiveDesk. (b) close-up of the physical tool tray

The two Bird receivers act like bricks and can be used simultaneously to perform operations in parallel. One of the bricks has a push button attached to it to register additional user input. This button is primarily used for creating new objects. Grasps

(i.e., attaching the brick to a virtual object) are registered when a brick is near or directly on the desktop surface. To release a grasp, the user lifts the brick off of the desktop (about 2 cm).

To select the current tool (select, delete, rectangle, triangle, line, circle) and current draw color, we use a physical tray and an ink-well metaphor (see Figure 5.9b). Users dunk a brick in a compartment in the tray to select a particular tool. A soft audio beep is heard to act as feedback for switching tools. Once a tool is selected, a prototype shape or tool icon is attached to the brick. The shape or icon is drawn in a semi-transparent layer so that users may see through the tool.

We have experimented with an alternative technique for users to select the current tool (square, circle, line, triangle, select). The technique takes advantage of the Z axis by having the virtual tools stacked on top of each other with each tool on a different layer or height. Raising or lowering the brick allows users to select a tool. While this approach seems more efficient, there are a number of interaction difficulties which arise. First, selecting a tool layer is challenging since our design required the user to click the brick button to lock in a layer and bring it down to the work surface. Secondly, there was no easy way to see all of the tool layers at all times. While one could come up with some designs for supporting this, we were restricted by the limited resolution of the active desk. Finally, we felt it was too fatiguing to cycle through the tools (browsing then selecting) by raising one's hand. The physical tool tray has advantages in that the user always knows what functions are available (predictableness), learns the approximate gesture needed to get to the tool, and can use the physical constraints of the tool compartments to make a coarse, imprecise, ballistic gesture to activate the tool.

The concept of an *anchor* and *actuator* have been defined in interactions that involve two or more bricks (see Figure 5.10). An anchor serves as the origin of an interaction operation. Anchors often specify an orientation value as well as a positional value. Actuators only specify positional values and operate within a frame of reference defined by an anchor. For example, performing a stretching operation on a virtual object involves using two bricks one as an anchor and the other as an actuator. The *first* brick attached to the virtual object acts as an anchor. The object can be moved or rotated. When the *second* brick is attached, it serves as an actuator. Position information is registered relative to the anchor brick. If the first anchor brick is released, the actuator brick is promoted to the role of an anchor.

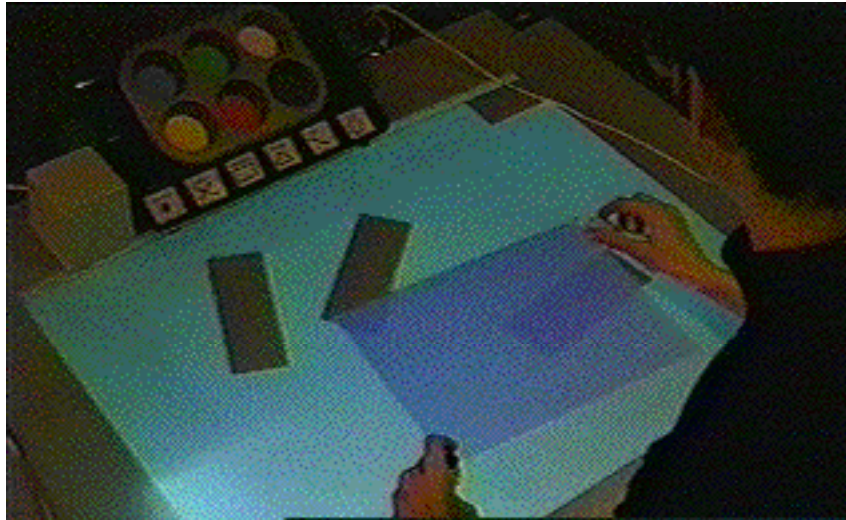


Figure 5.10. Two bricks are used to simultaneously translate, scale and rotate the rectangle. The first brick serves as an “anchor” while the second brick serves as an “actuator.”

The prototype allows users to manipulate two virtual objects at the same time by attaching bricks to each one. This provides a natural level of expressiveness in terms of improved throughput (moving 2 objects at the same time) as well as aiding certain alignment tasks. Beyond the anchor/actuator roles that the bricks take on during manipulations, we explored the issue of assigning permanent or transient functionality to the bricks. Because we could only have two bricks (due to technology constraints), we were forced to go with the transient functionality. However, it is interesting to speculate on the value of having all bricks behaving the same (which can be used interchangeably) or having bricks designated a permanent interaction role (i.e. a specific tool or function).

Following Guiard’s bimanual principles [Guiard, 1987], we originally designed all of the object creation and manipulation interactions as having the non-dominant hand serve as the “frame of reference” for the interaction. This worked well for the rectangle tool but less so for the circle and line tools (see Figure 5.11 and 5.12). Essentially, the visual attention and perception factors dominated the interaction technique. For example, we wanted to avoid interactions that may become visually obscured due to the hands getting in the way. Thus, an interaction technique for creating a circle would have had the non-dominant hand specify the center of the circle while the dominant hand specifies the radius (see Figure 5.11b). This design forces part of the circle to be obscured by the non-dominant hand. Instead, we have designed an interaction that creates a circle by having the two bricks serve as the

diameter for the circle (see Figure 5.11c). The frame of reference for the interaction is not specified by the non-dominant hand but instead by the slope of the line created by the position of both hands (see Figure 5.12). In general, we believe that the *physical triangle of interaction* (defined by the position of the two hands on the Active desk and the user's body) should contain the majority of visual feedback that the interaction technique employs.

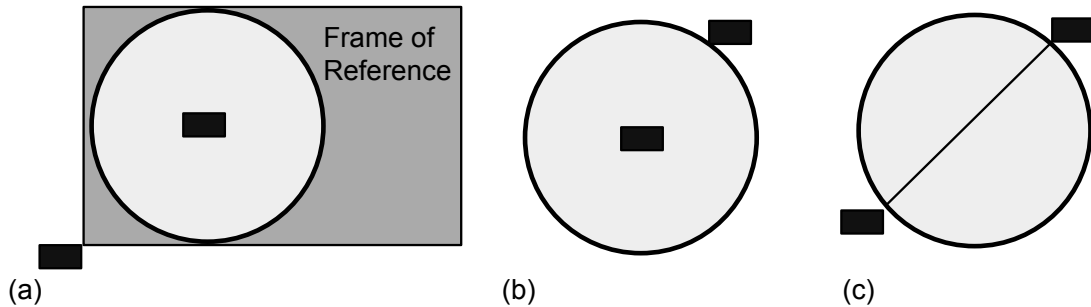


Figure 5.11. Creating a circle with two bricks. In (a) we see the original design with the non-dominant hand brick defining the frame of reference for the interaction. In (b) portions of the circle are obscured by the hands. The chosen technique (c) uses the slope of the line between the two bricks as the diameter for the circle. Note, the left black rectangle in each figure is the non-dominant hand brick.

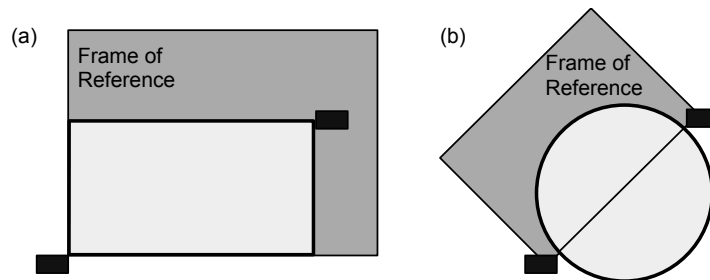


Figure 5.12 Frame of reference for two handed interactions: (a) non-dominant hand defines frame of reference for rectangle; (b) slope of the line created by the position of both hands defines the frame of reference for the circle.

User Evaluation

There were many interesting interaction issues that emerged due to the development of the GraspDraw prototype application. We report on feedback obtained by informal user testing of approximately 30 users (20 who were unfamiliar with the research objectives and 10 people within our research lab). Most of the users worked with the prototype to generate simple drawings. Only 5 of these users had formal training in graphic design or art. The sessions would last between 5 to 30 minutes.

All of the users who worked with the interface performed parallel operations (e.g., translate, rotate and scale) at a very early stage of using the application. Within a few minutes of using the application, users became very adept at making drawings and manipulating virtual objects.

Furthermore, users did encounter some difficulties while using the GraspDraw prototype. First, some users commented on the fact that the bricks were tethered, which hindered some of their interactions. Moreover, depending on the angle of the desktop, physical artifacts will tend to slide off which is an important design issue if we are relying on spatial arrangements and persistence. Secondly, some users had difficulty knowing whether they were in “select” mode or “delete” mode. While we did provide for a different cursor shape to indicate mode, the resolution of the desk made this difficult to see. Indeed, one has to wonder whether the bricks should have any cursor or virtual representation on the screen.

Thirdly, we noticed that many users had trouble when it came to performing precise alignments between objects as well as creating very small objects (i.e., when both bricks were right next to each other). Some of this difficulty could be attributed to technology: the low resolution and distorted imaging of the monitor on the Active desk (caused by the NTSC signal conversion) as well as some “jitter” introduced by the flock of Birds input devices. However, we believe that this is not the only source of the problems. When holding a brick and operating directly on the image, the physical shape of the brick can obscure part of the underlying data that users wish to manipulate. Moreover, the user’s hands, arms and other physical artifacts on the desktop can also obscure part of the application data. Another interesting effect of working on a large scale drafting table size image is that it encourages (and sometimes requires) a different scale of gestures compared to, say, a mouse. Reaching to the top of the table to get to the physical tool tray proved difficult for some users. Beyond the scale of gesturing, the large image makes it more effortful to quickly scan the entire image. All of these factors must be considered when constructing Graspable UIs where the input control space is superimposed on the output display space.

5.3 Prototype 2: Bricks for curve editing

The second application builds upon the previous bricks prototype and investigates a specific interaction task, curve editing, within the context of a more robust

commercial application. We first describe a prototype system and then, the efforts involved in transferring the ideas into a commercial application.

In our context we define the task of curve editing as one of interactively adjusting the contour or shape of an existing curve. We concentrate on editing curves to create models or motion paths for character animations. From a user's perspective, the adjustments include moving, stretching, bending, sharpening and smoothing a curve or a portion of a curve. From the computer's perspective, the curve is defined mathematically by a set of points or control vertices (CVs) in Euclidean space. The number of CVs and their placement in space define the shape of a curve. Many computer aided drawing packages force users to have a sophisticated understanding of the underlying mathematical representation of the curves in order for them to get the desired shape.

5.3.1 Implementation

In our first curve editing prototype, we used two Ascension flock of bird receivers as bricks for editing hermite curves. The hermite curves [Foley, et. al., 1995] define a parametric curve by the use of two vectors at the head and tail of the curve. Each of these vectors consists to two control vertices (see Figure 5.13). To change the shape of the curve, one or both vectors can be adjusted. Each endpoint vector has 4 degrees of freedom to manipulate: x position, y position, vector angle, vector length.

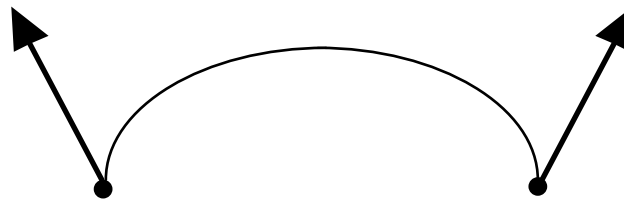


Figure 5.13. Hermite curve. Each endpoint vector has 4 degrees of freedom to manipulate: x position, y position, vector angle, vector length.

For our prototype, we did not use the Active desk but instead used a standard SGI computer monitor and had the bird receivers operate in front of the monitor on a horizontal desktop surface. Our simple hermite curve editor senses both the position and orientation of the bricks and directly maps these values to the hermite curve vectors. This allows for great flexibility and expressiveness for editing a curve as both endpoints can be positioned and oriented at the same time.

Our intention is it to mimic the "flexible curve" interaction style we observed in the exploratory study where each brick corresponds to a small segment on the physical curve. Rapid attach/detach actions would allow the user to make many quick position and rotation adjustments along the curve.

From an implementation perspective, the z dimension is used to attach/detach from the curve. In this configuration, it is difficult to assign one of the remaining 6 degrees of freedom of the bird receiver to the role of modifying the hermite vector scale values. It is an unnatural mapping. However, we believe the rapid position and rotation of many points along a heavily segmented curve may minimize the need to scale the hermite vectors. Alternatively, we can use 2 bird receivers to edit one endpoint vector. In this situation the second bird is used to specify the vector scale by its relative distance from the first receiver.

User Evaluation

We tested the prototype on 5 users, 4 of whom were very familiar with curve editing and developing user interfaces for curve manipulations. All of the users found the prototype interesting but wondered if the design would work in a more complicated environment. As the prototype only allowed for the manipulation of one hermite curve segment, we wanted to see how well it would work with multiple hermite curve segments and more robust curve representations.

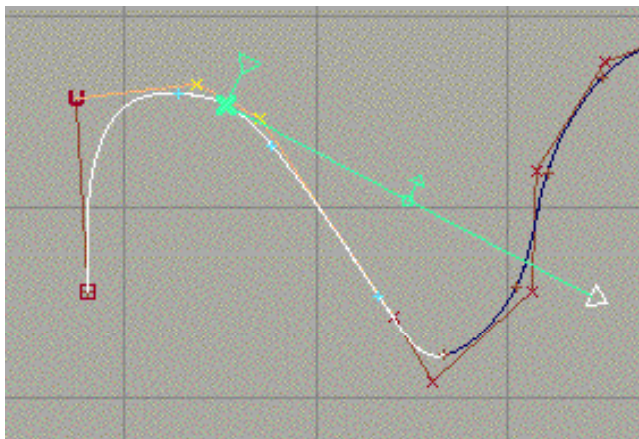
5.3.2 Development in larger application context

We implemented a critical mass of the Graspable UI into a modified version of Alias Studio^a, a high-end 3D modeling and animation program for SGI machines. Specifically, we explored how multiple bricks could be used to aid curve editing tasks. Developing within Studio gave us access to a very sophisticated mathematics library used for representing and manipulating curves.

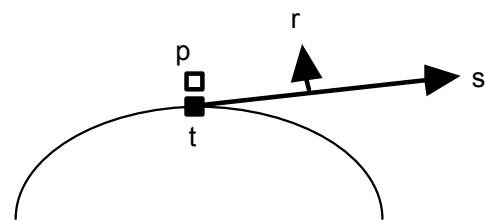
Within Alias Studio^a, curves can be edited in a variety of ways. One of the more popular direct manipulation technique is to use the "curve editor widget." Once a curve is selected, the manipulation widget attaches to the curve. To affect a region of the curve, the widget is dragged to an "edit point" using a sub-component on the widget (see Figure 5.14b part p). The widget glides along the curve as it is being moved so all a user has to do is move the mouse in the left and right direction to get the widget to traverse the curve. Once in position the user can move, rotate or scale the current point on the curve using separate sub-components on the widget (see

Figure 5.14b part t, r and s). The manipulation of the curve widget does not only affect the current edit point along the curve but a curve region or segment. Typically this is 2 or 3 CVs before and after the current widget position. Depending on how the curve was created (and consequently the placement and density of CVs) manipulating the widget can have quite a different affect on a curve. The move operation translates the affected CVs in the same direction that the curve widget is being dragged. Rotating the widget causes the CVs to rotate in the same direction causing the curve to twist. Lastly, scaling the widget will adjust the density of the affected CVs (bringing them closer to the curve widget or pushing them away). Finally note that the widget provides for orthogonal control so only one factor (translate, rotate, scale, and widget position) can be adjusted at any given time.

In our implementation two Ascension flock of bird receivers have been integrated into the Studio program. We were unable to easily implement the hermite style of curve editing where both bricks can be used to modify two points (i.e., both position and orientation) along the curve simultaneously. This is due to two basic reasons. First the application has been built with the assumption that there is only one “pointing device” driving the application. Introducing the second brick as a primary pointing device was infeasible. Secondly, the underlying mathematical NURB (non-uniform rational b-spline) representation and associated libraries could not handle this style of interaction.



(a)



(b)

Figure 5.14. Alias Studio curve editor widget. Figure 5.14a shows the widget within Studio while figure 5.14b labels subcomponents of the widget. There are 4 modes of editing a user can select with the curve editor widget: (t) translate point on curve, (r) rotate point on curve, (s) scale point on curve, (p) reposition widget on curve.

Therefore, we had to settle for our secondary design. One brick is used to specify the point on the curve to be edited as well as the new translation and rotation values. The second brick is used to specify a new scale value if needed by the relative offset from the first brick. Thus, both bricks can be used to simultaneously edit the position, orientation and scale factor for points along a curve. In this approach, the same “curve editor widget” is used so the application visually looks the same. However, one brick is attached to one endpoint (see Figure 5.14b part t) which serves as the primary pointer and the second brick is attached to the other endpoint of the widget (see Figure 5.14b part s).

User Evaluation

We tested the prototype on approximately 7 users, all of whom were very familiar with curve editing and developing user interfaces for curve manipulations. Our informal user study consisted of demonstrating the interaction techniques to the subject and then having the subject “play” with it for a few minutes (between 2 to 15). We asked subjects to describe what they liked and disliked about the interface. All of the subjects had a strong preference for the ability to position and rotate points along the curve. Users felt more awkward using the second brick as the means of specifying a scale factor. This may be attributed to the fact that minimal visual feedback was provided for the position of the second brick. Moreover, the scale factor seems to be a more unintuitive attribute for novices to understand. In essence, the task relies heavily on the ability for the user to predict a response (i.e., how will the curve change) given a change in the widget (or input device) which has no physical analogy. Even providing a very tight feedback loop will not solve the interaction problems if the users are not able to discern the underlying model.

Ultimately, we would like a design which matched more closely to what our original curve matching exploratory studies exhibited. In our envisioned system, users would be able to position one or two bricks on the curve and directly manipulate the curve. One brick would serve as a clamp (i.e., the ability to freeze portions of the curve) while the second brick is used to move or bend portions of the curve. Again, this style of interaction was exhibited in the curve matching study. We have not implemented this approach within Alias Studio due to the mathematical complexity and architectural constraint of supporting multiple simultaneous input devices.

In essence this effort showed us how challenging it is to deviate from the standard input device and event processing model in commercial applications. Moreover, this

difficulty goes beyond commercial applications as many toolkits cannot handle these requirements as well.

In terms of Graspable UI concepts we found that users were very excited about the ability to use the bricks to simultaneously position and rotate points along the curve. We observed them quickly selecting points and making quick adjustments and repeating this process. However, they were not able to always get their desired shape due to the underlying mathematical representation and the awkward access to the “scale” factors for the curve points. We found that the standard “curve editor widget” was very unnatural in that it always forced the user to affect only one dimension at a time (e.g., position or rotate) and thus users were unable to chunk and phrase their interactions at their desired granularity. In addition, we realized that some of the curve interaction techniques (e.g., scale) were very unnatural and hard to predict outcomes when using the widget. Perhaps this was due to the fact that these operations did not correspond to any physical analogy or physical metaphor. Our attempt to provide physical objects to overcome this failed as the physical objects suggested one style of interaction but the underlying software could not support this.

5.4 Prototype 3: FlipBricks

The third prototype, called “flipbricks” is a new input device specifically designed to economize the bricks design. That is, we would like to have multiple bricks available but want to minimize the physical clutter and at the same time cluster similar functionality. Specifically, we consider this design in terms of using a flipbrick to represent menu choices as well as rapid task switching.

A number of techniques can be developed when using a brick to make menu selections. One option could design a pop-up menu in which the user slides the brick up or down to choose menu selections. When pressure is released on the brick, the current menu item is selected (see Figure 5.15b). A second option uses the brick to specify an angle for selecting items in a pie menu (see Figure 5.15c). While these are interesting and valid designs, they do not take advantage of the full properties of the physical objects as the flipbricks do.

The flipbrick is different than the previous styles of manipulations described so far. With the flipbrick we assign a menu selection on each face of the brick (see Figure 5.15a). Whichever side is facing up defines the menu selection. This design takes

advantage of our fine finger manipulation skills and our object manipulation knowledge. Similar operations can be placed on adjacent sides so that they can be issued in sequence. For example the Copy and Paste operations could have adjoining faces (see Figure 5.16). The shape of the brick may dictate how the menu selection items should be assigned to each face. That is, neutral or common selections should be placed on faces of the brick that have the most surface area and object stability (e.g., top and bottom face of a domino shaped brick instead of any of the side edges).

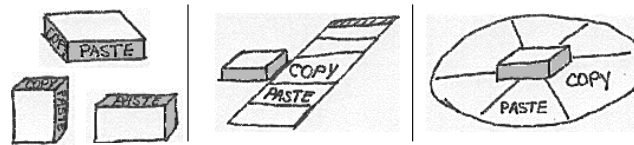


Figure 5.15. Menu design options. (a) Flip brick menus, (b) Pop-up menus, (c) Pie menus.

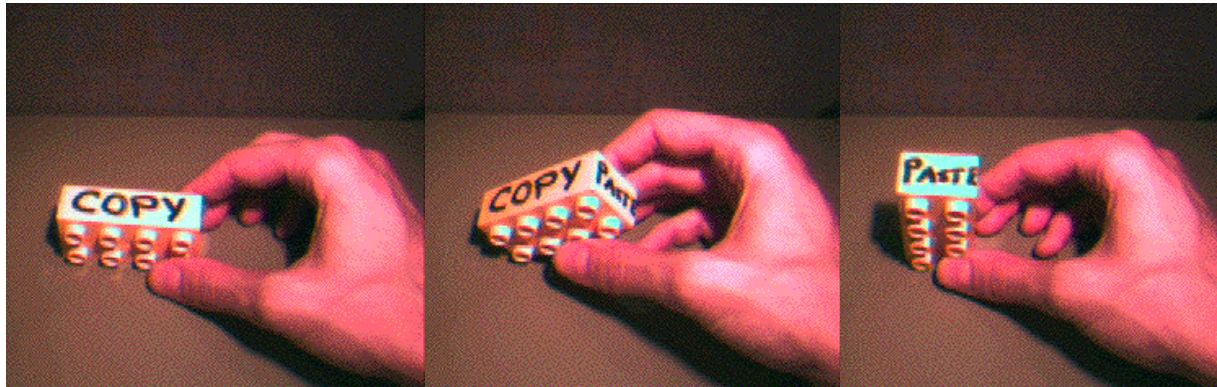


Figure 5.16. Proposed layout and manipulation sequence for flipbricks that have related command functionality. User starts in (a) issuing the “Copy” command then transitions to the “Paste” command by flipping the brick (b-c).

The flipbrick design has the advantage of clustering similar functions to the same physical object and potentially reducing the number of physical objects needed in a workspace at a given time.

An initial simple prototype of the flipbrick design was built using the Ascension Flock of Birds^a. Admittedly, it was very clumsy as the tethered receiver made it very difficult to “flip” the receiver. Nevertheless, the flipping action was associated with two features: (1) selecting the current drawing tool and (2) selecting drawing modes (i.e., constraints on/off, snapping on/off, etc.).

A more robust design was built using the Wacom tablet and sensors (see Figure 5.17). Three of the faces of the brick have sensors embedded in them plus two buttons. The two largest faces and one of the edges of the brick have sensors built into them. With the sensors, we are able to tell which face of the brick is facing up and whether a user is pressing one of the two “dimple” buttons. Users can distinguish the current mode (i.e., which face is up) by tactile sensing (one side is covered in felt material). In general, using a variety of different textures is a simple technique for determining sides tactily. Other tactile cues, such as asymmetrical shapes could further disambiguate the orientation of the brick.



Figure 5.17 A customized, wireless “flipbrick” that operates on a Wacom tablet.

One proposed use of flipbricks allows the user to operate the flipbrick as a simple one button mouse, flipping the brick to rapidly switch between functions or application programs. Continued exploration on this design is still needed but it appears to be promising. One unusual issue that has arisen so far is the soft “thumping” sound that is generated each time the brick is flipped. This sound may become annoying to users but may be minimized by careful selection of material used for the flipbrick and tablet surface.

Indeed, there are many open research questions to be answered about flipbricks. For example, in terms of menuing, what shape works best in terms of number of sides (e.g., a pyramid shape or an octagon shaped rod)? How many sides should be used? How does the user learn what is on each face edge? Can multiple flipbricks be used at the same time?

User Evaluation

We tested the flipbrick prototype on approximately 15 users. Our informal test consisted of showing them the flipbrick device and how it works within an application (e.g., to flip between 2 different tool palettes). All of the users understood the concept in less than a minute. Many commented on the need for having a way of knowing which side is up using a tactile cue such as texture or size.

From our initial feedback and designs, we believe this approach to be worthy of deeper investigation. Specifically, the flipbricks offer a design solution that can reduce the physical clutter of having only one brick per command and, at the same time, offers a way of clustering similar functionality. In some sense, the flipbrick goes beyond offering space-multiplexed input in that it also offers modes and state information not only by its spatial presence and location but also by the physical act of manipulating the object. We call this "manipulation multiplexing."

5.5 Summary

This chapter described a detailed case study for a specific set of Graspable user interfaces known as "bricks." Specifically, a set of exploratory studies were conducted followed by the development of three prototypes (GraspDraw, curve editing and flipbricks). Through this investigation we have discovered some of the design challenges posed by implementing Graspable user interfaces. These challenges and lessons learned include:

- Current software systems are hard to adapt. That is, many assumptions have been built into software toolkits to handle only a single pointing device and a single stream of input.
- Tethered devices can get in the way during use. For example, having more than two bird receivers on the Active Desk is infeasible due to the high potential for wires becoming entangled. While tethered devices have the benefit of not easily being removed from a workspace, the devices can often impede a user's natural gesturing style.
- Hands can get in the way. For example, when operating on the Active desk, the hands and arms can obscure portions of the computer display. As well, during interactions, a person's hands can bump into and displace physical objects which are being used for solving the current task.

- It can be difficult to map physical tools to computer functions especially when the functions are abstract.
- Very little software has been written to handle some of the physical motions observed in the exploratory studies.
- The cost of building interfaces that capture physical motions is quite difficult. That is, inexpensive recognition of multiple (i.e., more than two), spatial-aware objects with accurate and precise sensing resolution is not commercially available yet. However, we believe tablet technology holds great potential.

Finally, note that Appendix B presents a series of design variations for Bricks to illustrate a variety of interaction styles. The design variations include using bricks without virtual context, 3D applications and transitioning between physical and virtual interactions. All of the studies and prototypes served to gain further design experience with the 5 Graspable UI design properties. The next chapter formally evaluates the core Graspable UI design properties of space-multiplex input and physical form factors.

Chapter 6: Empirical Evaluations on Space-Multiplexing Input for Graspable UIs

This chapter describes two experiments that empirically investigate the property of space-multiplexing input for Graspable UIs. We conduct these experiments to investigate the claims that the proposed Graspable UI properties provide faster input and less errors compared to conventional GUI styles of input for a set of spatial tasks.

The first experiment focuses on manipulation issues for tasks when users already have input devices acquired in their hands. Here we compare three space-multiplexed conditions with a time-multiplexed condition. We predict that, in general, the space-multiplexed conditions will out-perform the time-multiplexed condition. In addition, we want to determine if and how the "physical form" of the input devices (ranging from generic to specific form factors) can influence performance. Finally, we want to understand how manipulation performance varies through time (e.g., learning) and as the task becomes more difficult (i.e., the number of degrees for manipulation increases).

The second experiment again focuses on the issue of space-multiplexed versus time-multiplexed input but examines the inter-device transaction phase of interactions and tests the utility of having generic vs. specialized form factors for input devices. That is, the experiment is designed to study the relative costs of acquiring physical devices (in the space-multiplex conditions) versus acquiring virtual controllers (in the time- multiplex condition). One possible advantage is the presence of visual and tactile mnemonics for the specialized, space-multiplexed input devices.

6.1 Experiment 1: Manipulating physical/logical devices

6.1.1 Design

The focus of this experiment is to examine time- vs. space-multiplex manipulation issues for tasks when users already have input devices acquired in their hands. For the experiment we varied the physical form of each input device configuration and the degree of task difficulty and asked subjects to match a target rectangle shape on the computer screen as quickly as possible. This is essentially the same task prototyped and discussed in Chapter 5.1.3. We defined one time-multiplexed and three space-multiplexed input device configurations. Note that the remaining Graspable UI properties were fixed: maximum concurrency, spatially-aware devices and high spatial reconfigurability of devices (e.g., free-ranging). All input configurations operated on a digitizer tablet. The first condition (time-multiplex) used a stylus device and most closely reflects the traditional GUI design. The next three conditions were space-multiplexed (see Figure 6.1). Two bricks in the shape of a round dial and a square block served as the second device configuration. The bricks (discussed in detail in Chapter 5) offer a strong compliance to the Graspable UI philosophy. A stretchable ruler and stretchable square are customized input devices and were constructed for this experiment serving as the third and fourth input device configurations. Both the stretchable ruler and square are more specialized devices which more closely match the properties of the task due to their shape and manipulability.

In defining the target stimuli we varied the difficulty or dimensionality of the task. A single dimension target would require the subject to alter only one parameter (i.e., only translate, rotate or scale) to match the target. These single dimension stimuli were considered the easiest target set. Two dimension targets would require the subject to alter two out of the three parameters (e.g., translate, rotate but not scale) to match the target. Finally some targets varied all three parameters. These served as the most difficult targets to acquire. Said slightly differently, we claim that the level of task difficulty corresponds directly with the task dimensionality. Finally, note that we considered translation as a single one dimensional parameter even though it is often described in terms of X and Y positions. That is, a straight line can be drawn from the initial subject's rectangle to the target rectangle; translation along this line is a one dimensional transformation from the subject's perspective. We had the subjects perform three repeated blocks of a set of trials for each input device configuration to measure learning effects on the task.

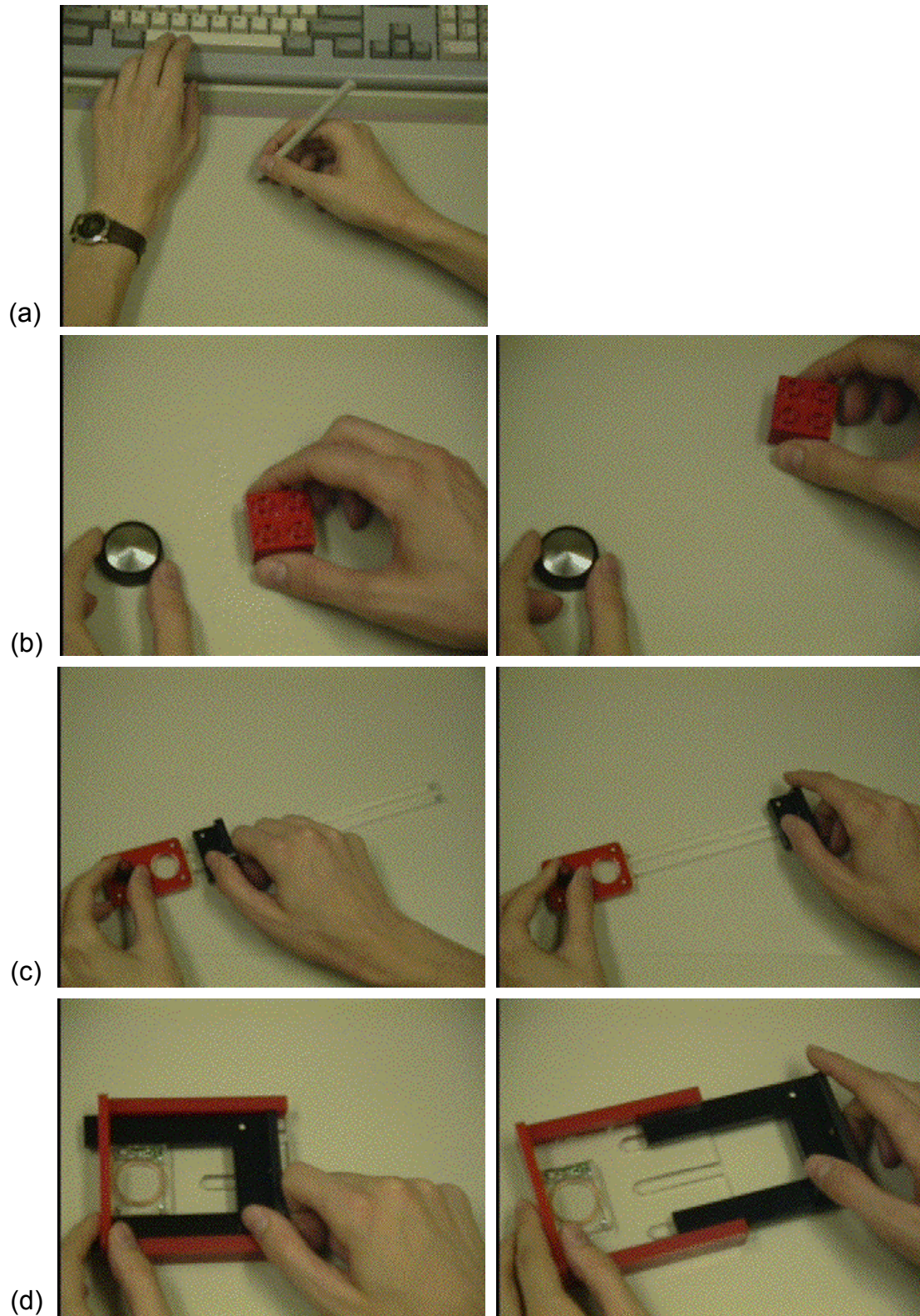


Figure 6.1 Four input device configurations. Configuration (a) serves as the traditional time-multiplexed GUI condition using a stylus while the bricks (b) and stretchable ruler (c) and square (d) serve as space-multiplex conditions. All devices operate on the Wacom digitizer tablet.

6.1.2 Hypotheses

Hypothesis 1a. *Space-multiplex performs better than time-multiplex*

We hypothesize that the space-multiplex input configurations will result in an overall superior task completion time performance over the time-multiplex stylus configuration. This is based on the belief that the space-multiplex condition minimizes interaction modes. With the time-multiplex condition, subjects must plan when to switch among tools to achieve the end goal. That is, their interactions are regulated by a set of atomic virtual functions which they must switch between. In contrast, the space-multiplex condition allows for more functionality to be active all of the time (e.g., translate, rotate, and scale operations can be active simultaneously). Instead of the interactions being regulated by virtual functions, the physical properties of the graspable functions (i.e., devices) offer physical laws and regulations that we are already very familiar with. Moreover, we argue that the space-multiplex conditions allow for designing interactions which are more compatible with how subjects' naturally chunk and phrase their actions. That is, for the task subjects probably articulate a high level goal such as "match my rectangle with the computer target." What follows for the time-multiplex situation is a more complex set of subtasks (e.g., separate, multiple sequences of translate, rotate, and scale combinations) compared to the space-multiplexed one which allows the expression of this goal at the interaction level.

Hypothesis 1b: *As task dimensionality increases, space-multiplex performs better than time-multiplex*

As the dimensionality of the task increases (i.e., the tasks become more difficult), performance will degrade in the time-multiplex condition. This is due to the fact that the time-multiplex condition (e.g., stylus) often only allow one dimensional transformations at any given time. Thus, parallel transformations are not possible and potentially degrade performance. Nevertheless, this time-multiplex design should be superior over the space-multiplex conditions for single dimension tasks because it is designed to have constrained or isolated single dimensional transformations. For example, it will be difficult to keep the stretchable square device perfectly horizontal (i.e., no rotation changes) during translations. We predict that the space-multiplex input conditions will also degrade as dimensionality of the task increase but to a much lesser degree. The degradation is primarily attributed to the increase in task difficulty.

Hypothesis 2. *In space-multiplex conditions, specialized devices perform better than generic*

We predict a higher degree of coordination for the stretchable ruler and square over the brick conditions. This is primarily due to the fact that the bricks can operate independently (i.e., the two sensors are housed in two separate physical objects) while the ruler and square have inherent coordination built into the physical design of the artifact (i.e., the two sensors are housed in the same physical artifact). Said slightly differently, we hypothesize that motor limb coordination will be facilitated by the specialized devices which have the sensors housed in the same physical artifact. This improved limb coordination will, in turn, result in superior task completion performance times.

We believe that motor limb coordination is improved because the connection between the two devices gives a physical relationship between the two dimensions being adjusted that parallels the virtual relationship. Just as the additional physical constraints in the tower of Hanoi/oranges/tea cups task helped the user with mental problem solving, the physical constraints in the ruler and stretchable square help the users physically maintain these relationships that exists between the dimensions of the virtual and real rectangle being drawn.

Hypothesis 3. *Space-multiplex input is easier to learn*

We expect a learning effect to be present in both the time-multiplex condition and the space-multiplex input conditions. However, we expect the space-multiplex task to be easier to learn than the time-multiplex one. This is due to the fact that there is more cognitive management to learn in the time-multiplex case. With the space-multiplex condition, all functionality is available and loaded into the physical input devices. Manipulation of physical devices is a well established, finely tuned learned motor behavior. In contrast, the time-multiplex condition requires extra cognitive management and planning as all functionality is not available all of the time. Subjects must decide which function to make active and decide when to switch to a different function to best achieve solving the task at hand. Thus, we predict that subjects will take longer to become proficient in the time-multiplex condition.

6.1.3 Method

Subjects

Twelve subjects participated in the experiment. Only one was left handed. All subjects except one had minimal exposure to operating the tablet device (10 have

used the tablet a few times, one never and one makes use a few times a month). All subjects were university students (mostly graduate students) and were naive to the purpose and predictions of the experiment.

Equipment

The task was performed on a Silicon Graphics Indy workstation computer using a Wacom tablet (see Figure 6.2). The first time-multiplex input device configuration used a pressure sensitive stylus. The remaining three space-multiplexed input conditions used customized devices with Wacom sensors built into the physical housing. The Wacom device operated in "multimode" which can simultaneously sense a stylus and puck device. We removed the stylus and puck sensors and placed them in our customized input devices (see Figure 6.1). Therefore, all three input device configurations reported a stream of X and Y tablet positions for both sensors. Also note that the sensors are small, wireless and batteryless which allowed us to build input devices without having wired tethers and yet still providing accurate and efficient sampling. All input devices operated in absolute position mode.

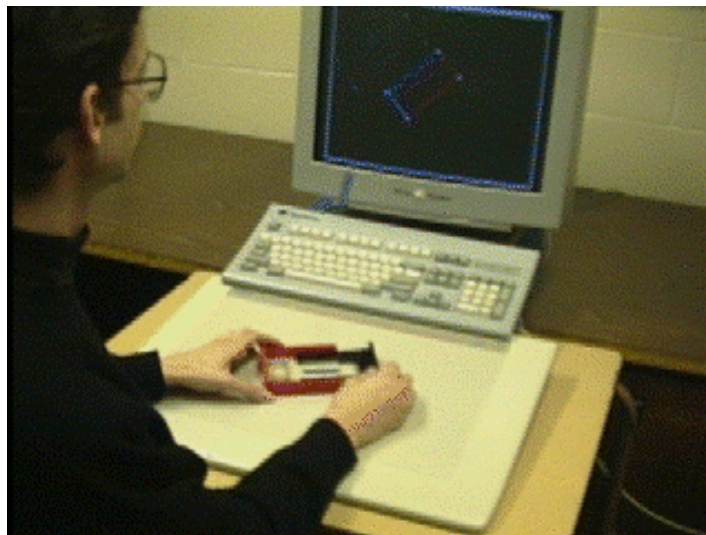


Figure 6.2. Experimental setup consisting of an SGI workstation, Wacom tablet, keyboard and graspable objects (stretchable square shown here).

The customized input devices used for the experiment have the following physical shape, dimensionality and manipulation range. The brick input device condition consists of two physical objects: a "dial" and "block." The "dial" shaped knob is approximately 1.25 inches in diameter and 0.5 inches height and colored black. The block is a square red LEGO brick having 1.25 inch width and length with a height of approximately 0.75 inches. Both objects have felt on the bottom surface for a

consistent smooth feel. The stretchable ruler measures 11 inches long with a thin knob at one end (for the non-dominant hand) and a slider on a track that extends to the opposite end. The ruler is approximately 1.5 inches wide. The puck sensor is housed in the knob end while the pen sensor is housed in the physical slider. The stretchable square has a more compact design in that its length dimension ranges from 4.25 inches to 8 inches. It has a constant width of 3.25 inches. The puck sensor is at the left edge while the pen sensor is at the right edge. Both the stretchable ruler and square were milled out of hard plastic.

Task

Subjects used each of the four input device configurations to match a series of target rectangle shapes as quickly as possible. The task reflects the use of three common operations (translate, rotate and scale) performed in many graphics applications. A total of 18 matching tasks were presented randomly to the subject for each block and input condition. Six were of dimensionality one, six had dimensionality 2 and six had dimensionality 3. Table 6.1 shows the 18 stimuli along with its corresponding dimensionality (where T=translate, R=rotate, and S=scale).

Stimuli	Dimension 1	Stimuli	Dimension 2	Stimuli	Dimension 3
1	T	7	TR	13	TRS
2	T	8	TR	14	TRS
3	R	9	RS	15	TRS
4	R	10	RS	16	TRS
5	S	11	TS	17	TRS
6	S	12	TS	18	TRS

Table 6.1 Eighteen task stimuli and corresponding dimensionality.

For each task, the subject is asked to align their rectangle shape with the target rectangle stimuli. Alignment consists of properly matching all four corners of the rectangle as well as color matching the edges. That is, each rectangle was drawn with two edges colored red and the other two edges colored blue (see Figure 6.3). This ensured that the rectangle was properly oriented (instead of having a 180 degree mismatch). When a corner matched (i.e., less than 5 pixels away from the target corner), the system would instantly highlight the corner handle (i.e., turn it from gray to yellow) to indicate a match. If all four corners matched for a given threshold time period (0.75 seconds), the trial was considered completed. Note that

subjects could not advance to the next trial until they successfully completed the current trial, thus trial errors were not possible.

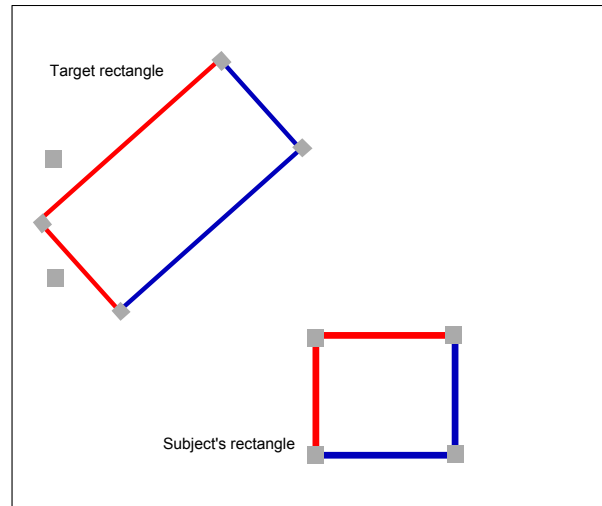


Figure 6.3. Sample task trial with target rectangle stimuli and user's rectangle.

For the bricks, stretchable ruler and square, the subject specifies the translation, rotation and scale factors by physically manipulating the input devices. That is, the current translation is determined by the average (X, Y) values for both sensors (pen and puck). The rotation is determined by the current slope of the line formed by both of the sensors while the scale factor is calculated as the relative distance between the two sensors. Therefore, all three dimensions can be manipulated simultaneously if the subject desires.

In contrast, the time-multiplex stylus input condition requires the subject to toggle between scale and rotate mode while solving the task. The current mode is indicated by the shape of the cursor (a cross for rotate mode and a normal arrow cursor for scale mode). Depending on the current mode, selecting a corner handle will cause the rectangle to scale (grow or shrink) or cause the rectangle to rotate (with the center of rotation being the center of the rectangle). Subjects toggle modes by hitting the spacebar on the keyboard which was positioned at the top of the Wacom tablet. This was achieved using their non-dominant hand. Thus, this condition employed the use of both hands from the subject. Moving the rectangle is achieved by selecting any interior region of the rectangle or edge and "dragging" it. This style of interaction is very similar to Macintosh graphics applications.

Finally, before each trial begins, the input devices must be placed in their "home positions" which are indicated by one or two rectangles on the screen. After a threshold time (1.5 seconds), a target rectangle shape appears and the trial begins.

Design and procedure

All twelve subjects used the four input device configurations (stylus, bricks, ruler and square). The trials were blocked into three sets of 18 stimuli. The stimuli were randomized for the three blocks and then this ordering was consistently presented across all four input device configurations. Subjects were assigned the sequence of input device conditions based on a Latin-square counterbalancing scheme to minimize ordering effects. Thus, this is a four factor $4 \times 3 \times 3 \times 6$ (Device \times Blocks \times Dimensions \times Trials), within subjects, repeated measures design.

For each new input device condition, subjects were given a minimal number of practice trials (up to 18) to acquaint themselves with the device or interaction techniques lasting no more than 5 minutes. After the experiment, the subjects were presented a questionnaire to obtain their subjective preference for each condition as well as to note any other issues they may have had.

In summary, each subject performed 54 trials on each of the four input device conditions resulting in a total of 216 scores per subject. Thus, the twelve subjects collectively generated a total of 2,592 data points.

6.1.4 Results and discussion

The main dependent variable of interest is task completion times for each trial. The task completion time is defined as the time from the initial stimulus presentation to the matching of the target rectangle by the subject. An analysis of variance (ANOVA) was conducted on the data (see Appendix D). Note that for the analysis reported here, subjects were grouped to factor in ordering (degrees of freedom equals (subjects - 1) - (orderings -1) which, in this experiment is (12-1) - (4-1) = 8). Given the hypotheses, we now present our findings.

Hypothesis 1a. *Space-multiplex performs better than time-multiplex*

As we predicted, there was a significant performance difference between input device conditions ($F(3,24) = 98.0, p < .001$). Specifically, the space-multiplex input configurations have an overall lower mean task completion time compared to the time-multiplex (stylus) configuration. A significant difference was found when a

pairwise means comparison was conducted between the conditions (stylus and bricks: $F(1,24) = 206.0$, $p < .001$; stylus and ruler: $F(1,24) = 186.2$, $p < .001$ and stylus and square $F(1,24) = 195.0$, $p < .001$). No difference was found when a pairwise means comparison was conducted between the space-multiplex conditions. These results suggest that for this class of task, the more generic brick devices can perform almost equivalently compared to the highly specialized stretchable ruler and square devices. However, we believe that the physical constraints that the stretchable square and ruler offer for the task would make them more advantageous. Our findings did not show this but perhaps with more trials or a more difficult task these differences could be observed. Figure 6.4 shows the mean response time values with 95% confidence error bars.

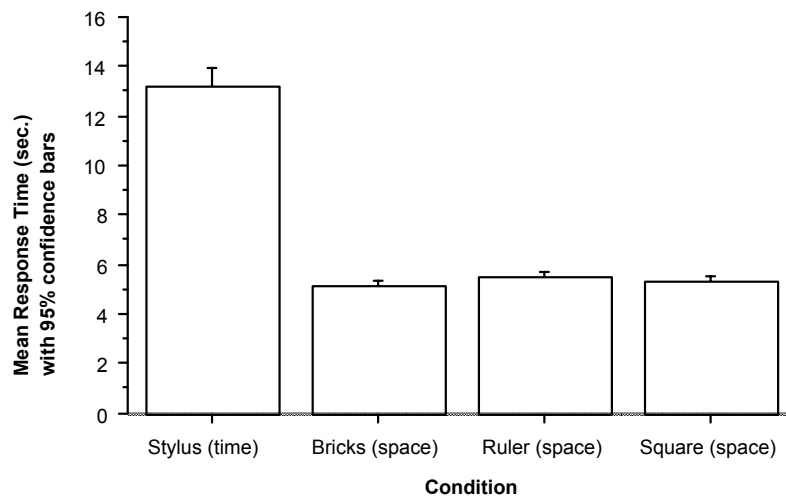


Figure 6.4. Mean response time as a function of input device configuration.

Hypothesis 1b: *As task dimensionality increases, space-multiplex performs better than time-multiplex*

We found that task dimensionality has an effect on task completion time ($F(2,16) = 88.3$, $p < .001$). In addition, we found a significant interaction between input condition and task dimensionality ($F(6,48) = 57.0$, $p < .001$). Figure 6.5 shows the combined results of task completion time separated by input condition and task dimensionality. By examining Figure 6.5, we first observe that in the stylus, time-multiplex condition, task completion times increase as task dimensionality increases. This, however, is not true for the space-multiplex conditions which have a much lower increase in task completion time as the dimensionality increases. Very little

performance difference exists among the space-multiplex conditions for any given task dimensionality.

Against our intuition, the overall task completion times for the one dimensional tasks in the stylus, time-multiplex condition were significantly longer than the space-multiplex conditions. Since the stylus provides single dimensional transformations, we believed it would be superior to the space-multiplex conditions which are much more difficult to operate along a constrained single dimension. One explanation for this result could be that subjects sometimes did not realize that they only needed to do a single transformation. For example, it was observed that for a task that required only a single scale operation (i.e., two corners are matched at the start of the trial), subjects would sometimes translate their rectangle to match the other corners then perform the stretch action. Clearly this is not the most efficient way of completing the task. Similarly, the one dimensional rotation stimuli gave subjects trouble. This may suggest that our perceptual systems are not well suited for detecting and performing mental rotations, but this analysis is beyond the scope of this study. Nevertheless, when we examine learning effects we show that the stylus, time-multiplex performance improves over time and become roughly equivalent to the space-multiplex conditions for one dimensional tasks (see Figure 6.7).

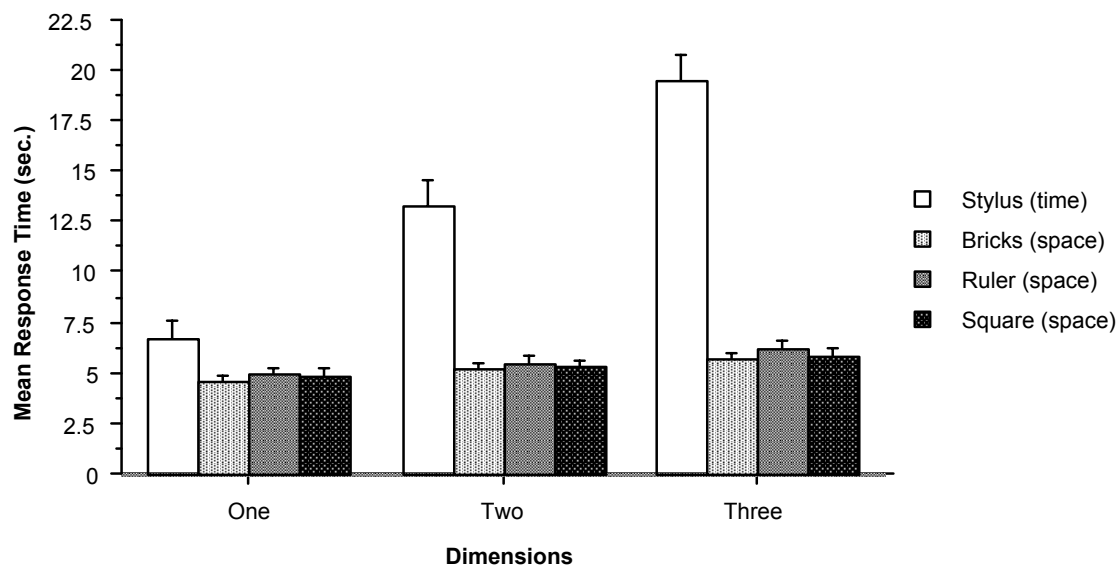


Figure 6.5 Mean response time as a function of input and task dimensionality.

Hypothesis 2. *In space-multiplex conditions, specialized devices perform better than generic*

The specialized space-multiplex conditions (ruler and square) performed statistically equivalent to the generic space-multiplex condition (bricks). See Figure 6.5. This may be attributed to the fact that our task could be easily accomplished with the use of visual feedback instead of tactile feedback or relying on the physical constraints of the specialized input devices. One could argue that once the devices are acquired, all of the space-multiplex conditions are the same from a motor-control perspective. That is, it does not matter what physical objects you use in each hand (i.e., a brick, dial, puck) for this task, the gross motor movements will be the same. While we believe that there exist differences between the specialized and generic devices, our task was not sensitive to detect this (floor effect).

A more detailed coordination analysis also verified that there was approximately the same degree of motor limb coordination using the bricks, stretchable square and ruler devices (see Appendix C). However, a more complex task (e.g., using more degrees of freedom) or a less visually dominant task (e.g., not having such a closed feedback loop) could yield different results as we place more demands on the motor channel and less on the visual channel.

Nevertheless, we believe that the specific physical form factors for the graspable functions can be used to suggest and facilitate the functionality they offer. Experiment 2 further explores these issues and the utility of specialized vs. generic devices.

Hypothesis 3. *Space-multiplex conditions easier to learn*

Within each input device condition we separated the trials into three consecutive blocks. Response times were calculated based on these blocks across all input device configurations. A learning effect is present across blocks ($F(2,16) = 24.8, p < .001$). In addition, a significant interaction effect exists between input device configuration and blocks ($F(6,48) = 13.0, p < .001$). However, when we examine this more closely we find that this difference is mostly attributed to the stylus, time-multiplex condition (see Figure 6.6). The slope of the stylus, time-multiplex condition is very different compared to the three space-multiplex conditions. With the space-multiplex conditions, we observe that learning has almost leveled off over the three blocks. One explanation is that the majority of learning happens very quickly in the space-multiplex conditions (perhaps in the first few practice trials). We argue that this rapid learning indicates that subjects are very familiar with manipulating these

input devices and are operating them based on their everyday knowledge, skills and a lifetime of learned motor behaviors.

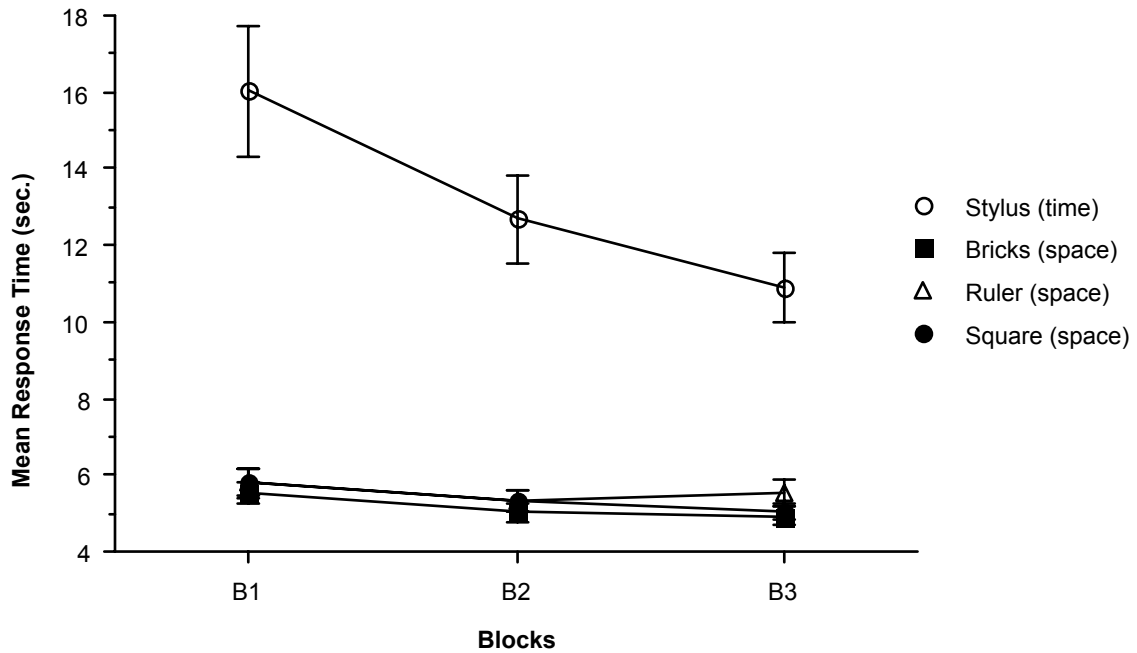


Figure 6.6. Mean response time by blocks and input device configurations.

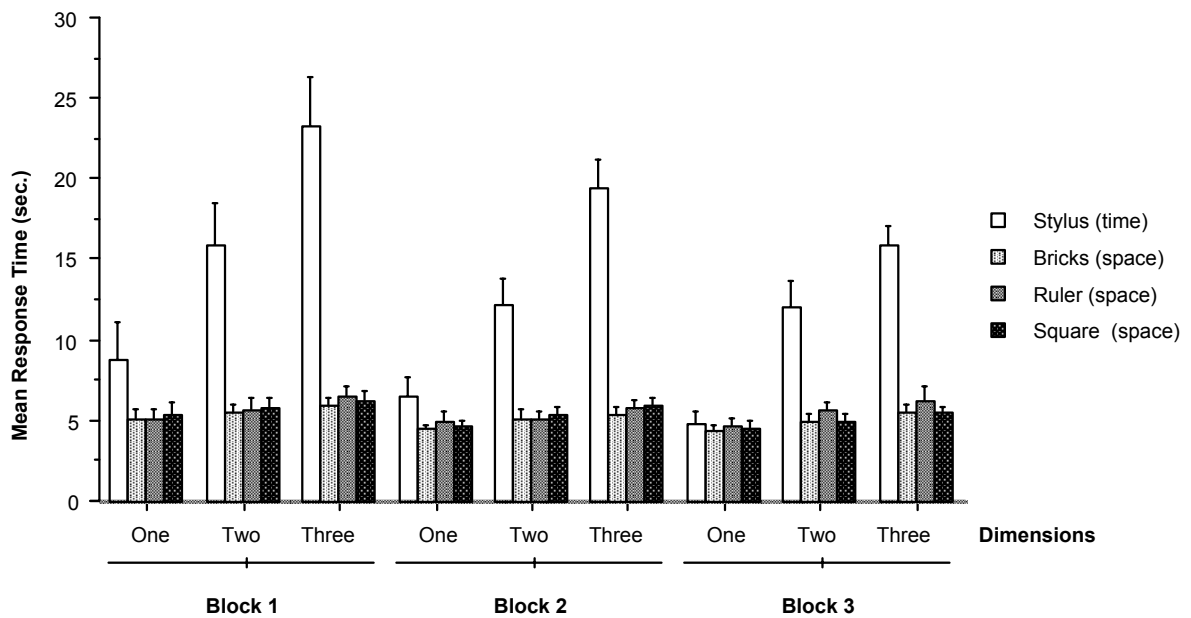


Figure 6.7. Mean response time by blocks, task dimensionality and input device configuration.

If we further decompose the learning effect into task dimensions within blocks and input condition, we can examine how task dimensionality effects learning (Figure

6.7). This again supports our hypothesis that the stylus, time-multiplex condition takes longer to become proficient compared to the space-multiplex conditions. Most notably, in Figure 6.7 task completion time decreases for the stylus, time-multiplex condition across blocks for one dimensional tasks. In fact, for the one dimension task during the third block of trials, subjects perform equally as well in the time- and space-multiplex conditions. The Power Law of Practice [Welford, 1968; Card, Moran and Newell, 1983] predicts that the time $T(n)$ to perform a task on the n th trial follows a negative exponential (see equations 6.1 and 6.2).

$$T(n) = T(1) * n^{-a} \quad (6.1)$$

$$\log T(n) = C - a * \log n \quad (\text{typical values for } a \text{ range between } 0.2 - 0.6) \quad (6.2)$$

In the 2 dimensional tasks we can see that it will take many more time-multiplex trials before reaching the same skilled performance as the space-multiplex conditions. Moreover, Figure 6.7 shows that still more trials will be needed for the 3 dimensional time-multiplexed tasks to reach equivalent space-multiplex performance. In general, as the task dimensionality increases, it will take progressively longer to reach equivalent space-multiplex performance.

After the experiment, subjects were asked to quantify their preferences for each of the input device configurations. They were asked to rate the physical comfort of each device (extreme discomfort to extreme comfort) as well as the ease in which they could solve the task (very difficult to very easy). A continuous scale from -2 to +2 was used for both ratings. Figures 6.8 and 6.9 show the results. On average, the stylus, time-multiplex condition was considered more uncomfortable and more difficult to use to solve the tasks. While the bricks appear to have the highest comfort factor, a pairwise means comparison shows only a significant difference between the stylus and bricks (student-t(11) = -2.51, $p < .02$) for physical comfort. As for ease of use, the stylus, time-multiplex condition was viewed as significantly more difficult than the space-multiplex input conditions (stylus and bricks: student-t(11) = -7.39, $p < .0001$; stylus and ruler: student-t(11) = -6.07, $p < .0001$; stylus and square: student-t(11) = -7.44, $p < .0001$). A pairwise means comparison indicates no significant difference between the bricks, ruler and square.

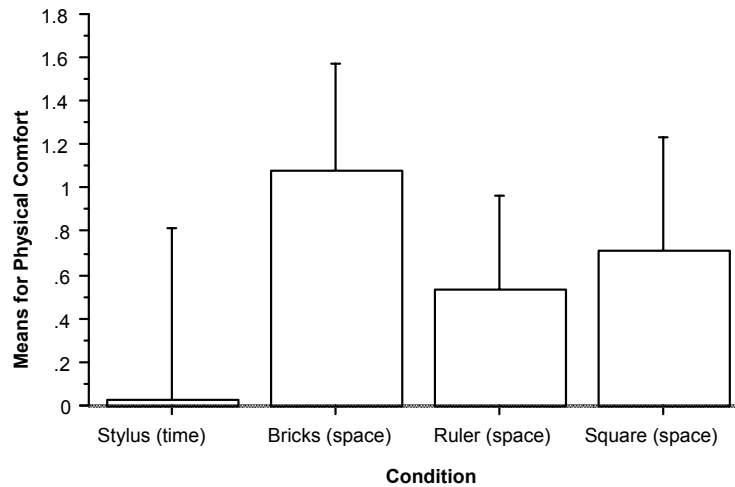


Figure 6.8. Physical comfort subjective rating (from -2 to +2) for each input device.

Subjects were then asked to rank order the overall preferences for each input device configuration. The bricks and stretchable square tied as the most preferred condition (ranking = 1.6). The stretchable ruler was next preferred (ranking = 2.6) followed by the stylus condition (ranking 3.9).

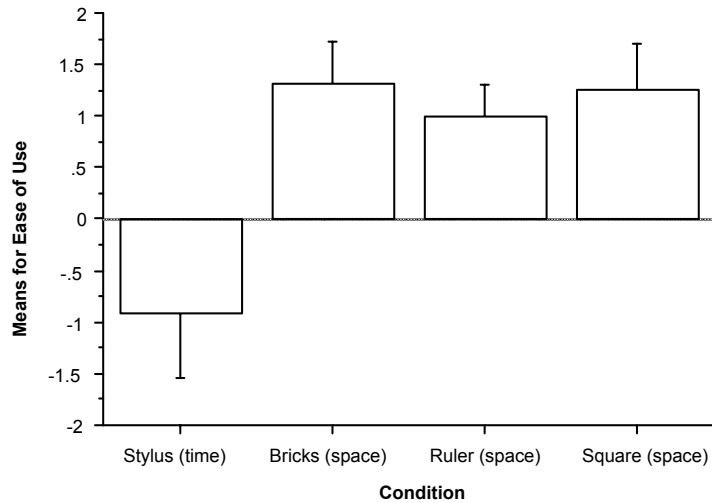


Figure 6.9. Ease of use subjective rating (ranging from -2 to +2) for each input device.

In Appendix C we present additional analysis of the data for this experiment called "coordination" analysis. This analysis offers more insight into the manipulation styles by more closely examining and comparing the manipulation efficiency used for the input conditions.

6.2 Experiment 2: Acquiring physical/logical devices

In this second experiment we again focus on the issue of space-multiplexed versus time-multiplexed input but examine the inter-device transaction phase of interactions. That is, the experiment is designed to study the relative costs of acquiring physical devices (in the space-multiplex conditions) versus acquiring virtual logical controllers (in the time-multiplex condition). We predict that the space-multiplex conditions will out perform the time-multiplex conditions due to the persistence of attachment between the physical device and logical controller. The act of selecting a widget or tool is made by physically acquiring an input device in the space-multiplex condition instead of selecting a logical tool handle in the time-multiplex condition. Moreover, we investigate the utility of specialized physical form factors versus generic form factors for input devices. Specialized input devices should out perform generic input devices in that the specialized forms suggest and facilitate their designated functionality. Said slightly differently, the specialized input devices can offer *tactile mnemonics*.

6.2.1 Design

This experiment varies the input style (from space-multiplexed to time-multiplexed) and the physical form factor of the input devices (generic to specific) and asks subjects to continuously track four randomly moving targets on the computer screen (see Figure 6.10). The four targets can be considered four user interface widgets which a user manipulates during a compound task or workflow. Two of the targets (mobile scrubwheel and flipbrick) require position and rotation adjustments while the other two targets (stretchable square and ruler) require position, rotation and scale adjustments. The continuous pursuit tracking task was chosen to emphasize the inter-device transaction phase, not the manipulation phase (as was explored in Experiment 1). That is, we are interested in studying the switching costs of the interaction. Condition 1 and 2 consists of space-multiplexed input while condition 3 consists of time-multiplexed input. With the space-multiplexed conditions, the physical input devices are permanently assigned and attached to a virtual, logical widget. Thus, to manipulate an on-screen widget, the subject directly manipulates the physical device. In contrast, the time-multiplex condition uses only one set of input devices which must be attached and detached to each logical widget before it is manipulated. Thus, subjects never need to release the physical input devices in the time-multiplex condition. Condition 1 uses specialized input devices (the mobile scrubwheel, flipbrick, stretchable square and ruler) while condition 2 uses a generic

puck and brick pair for each logical widget (thus a total of 4 pucks and 4 bricks are used). Here we are testing the utility of input devices having a specialized form which suggests and facilitates their designated functionality.

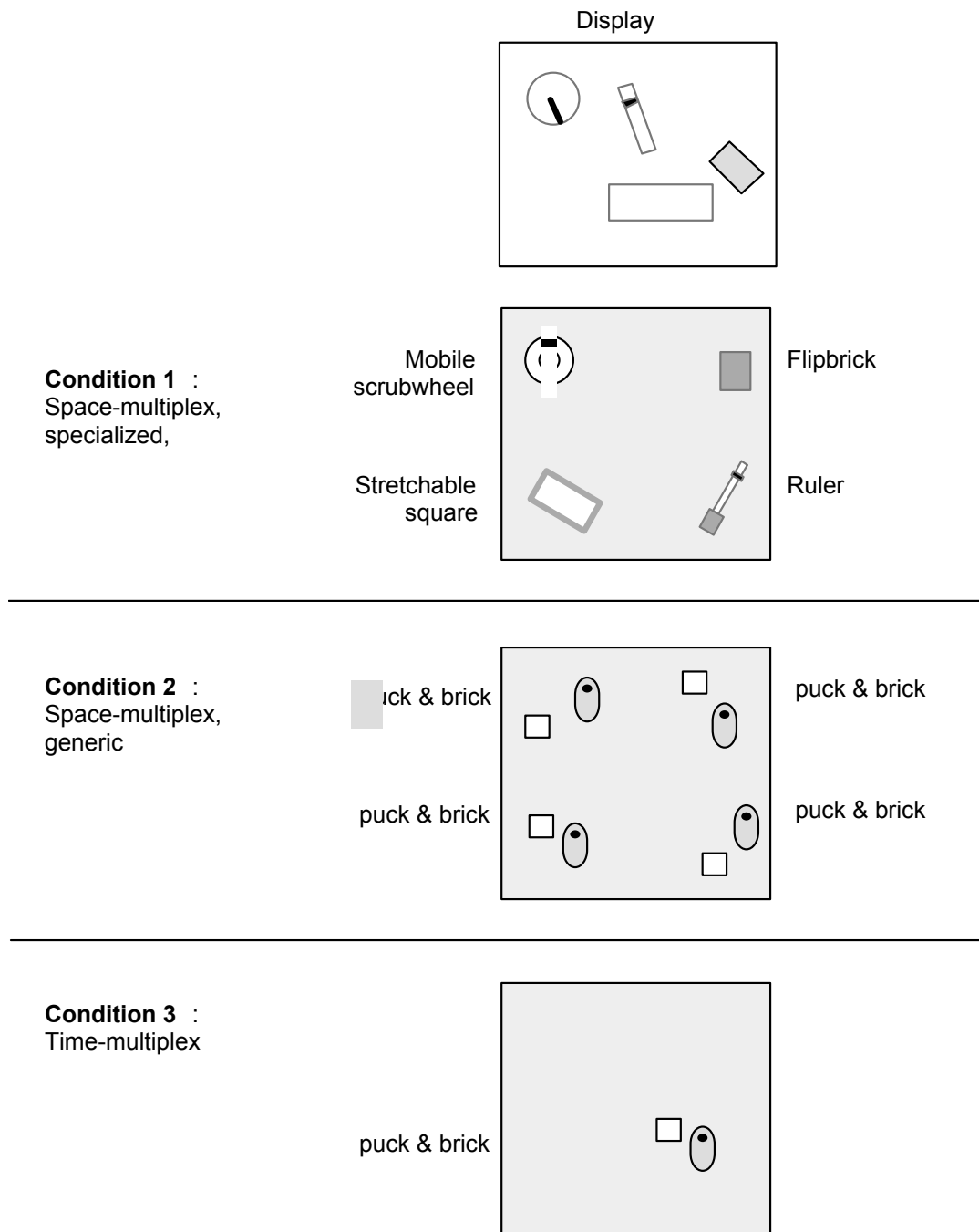


Figure 6.10. Three experimental conditions.

The three remaining Graspable UI properties did not vary throughout the experiment: maximum concurrency, spatially-aware devices, and high spatial

reconfigurability. All of the input devices operate on the Wacom tablets and thus are spatially-aware and offer highly spatial reconfigurability (i.e., free ranging). Subjects were encouraged to use as much concurrency as possible. Finally, the multiple target tracking task was designed as a two handed task. In summary this experiment is a three factor $3 \times 4 \times 6$ (MultiplexCondition \times Device \times Trials), within subjects, repeated measures, Latin-square design.

6.2.2 Hypotheses

Hypothesis 1. *Subjects perform better with space-multiplex than time-multiplex input conditions*

We predict that subjects will have superior performance for the space-multiplexed conditions over the time-multiplexed input condition. This is primarily due to the persistence of attachment between the physical input devices and the assigned virtual, logical widgets. We speculate that the physical input devices are easier to acquire than the corresponding virtual handles in the time-multiplex condition. Moreover, the space-multiplex conditions offer a greater potential for concurrent access and manipulation of virtual widgets by providing continuous access to the physical handles.

Note that hypothesis 1 is consistent with experiment 1. Hypothesis 2 is intended to tease out results that did not emerge in the first experiment. Specifically, within a space-multiplex input design, do the specialized physical form factors affect performance?

Hypothesis 2. *In space-multiplex conditions, subjects perform better with specialized than generic devices.*

Within the space-multiplex conditions, we predict that the specialized input devices will allow for superior task performance compared to the generic devices. Again, the specialized form factor should serve to remind the subject what virtual widget is attached to the device as well as facilitate the manipulation of the widget.

6.2.3 Method

Subjects

Twelve right-handed subjects participated in the experiment. All subjects except two had minimal exposure to operating a tablet device. Ten of the subjects were staff from Alias|Wavefront with significant computer experience. Two of the subjects

were graduates students from the university. Finally, all subjects were naive to the purpose and predictions of the experiment.

Equipment

The task was performed on a Silicon Graphics Indigo2 workstation computer using four 12"x12" Wacom tablets arranged in a 2x2 grid for the space-multiplex conditions and a single 18"x25" Wacom tablet for the time-multiplex condition (see Figure 6.11a-c). A SpecialiX serial expander was used to attach the four Wacom tablets simultaneously to the computer and all accessed the same X11 device driver. The program was written in C using a mixed-model of OpenGL (a graphics library) and X11 (for window and event-based input handling). The 2x2 grid of Wacom tablets was necessary due to the fact that the tablets can only support two sensors on them while operating in "multimode." Ideally, we would have run all conditions of the experiment on one large Wacom tablet if it could support multiple sensors (e.g., 8 or more). Each of the tablets map onto a full screen dimension. All input devices operated in absolute mode. Thus, moving a device to the bottom left of a tablet would have the corresponding effect of moving the virtual widget to the bottom left of the computer screen.

Four specialized input devices were used in the space-multiplex, specialized devices condition consisting of the stretchable square, ruler, flipbrick and mobile scrubwheel (see Figure 6.11a). Both the stretchable square and ruler were used in experiment 1. While the flipbrick can sense which side it is on, we ignored this and sense only position and orientation. The mobile scrubwheel senses both position and orientation (see Section 4.3.1). The devices were assigned to the same tablets in the 2x2 grid of tablets for all subjects (scrubwheel top left, flipbrick top right, ruler bottom left, and stretchable square bottom right tablet).

Four pairs of a brick and puck were used in the space-multiplexed, generic devices condition. The puck is a standard 4 button Wacom digitizing puck. The brick was a LEGO brick measuring 1.25 inches in width and length and having a height of approximately 0.75 inches. Inside the brick was a Wacom stylus sensor which is small, wireless and batteryless providing as accurate position information as a regular stylus device. Note that both the pucks and bricks have felt on the bottom surface for a consistent smooth feel. Each of the four tablets were labeled using a graphic picture to indicate the virtual widget which was permanently attached to the brick and puck pairing (see Figure 6.11b).

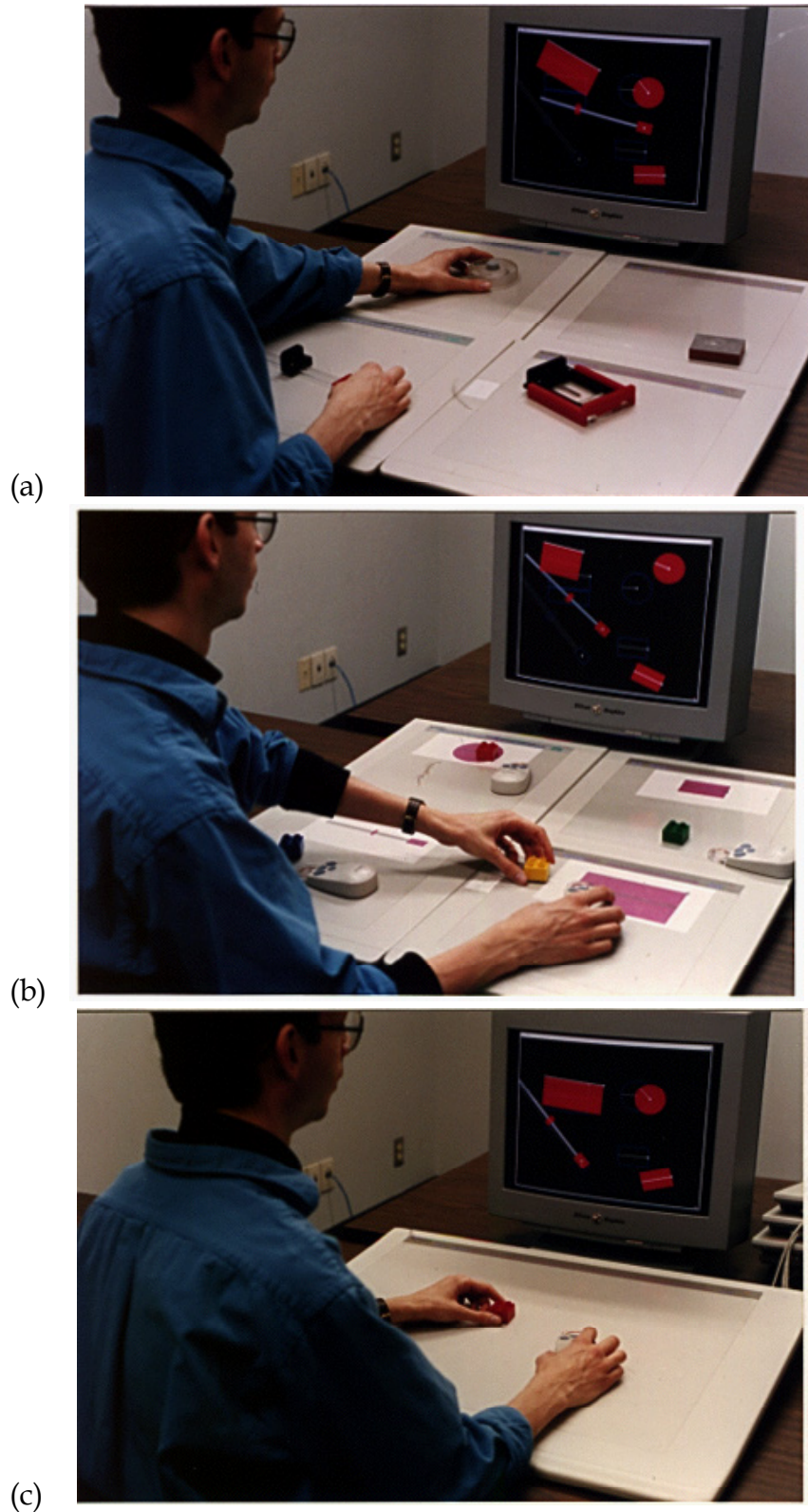


Figure 6.11. Experimental equipment set-up for the three conditions consisted of (a) space-multiplex with 4 specialized devices using 4 tablets, (b) space-multiplex with 4 puck and brick pairs of generic devices using 4 tablets and (c) time-multiplex with one puck and brick devices operating on a large tablet.

The time-multiplex condition used one puck and brick device on a single 18"x25" Wacom tablet (see Figure 6.11c).

Task

Subjects used the three input conditions on a multi target tracking task. A trial consisted of a 90 second pursuit tracking session. Six trials were conducted for each of the three input conditions for a total of 18 trials. Before the trial begins, subjects must align their 4 widgets on top of the 4 computer targets. When the trial begins, the 4 computer targets begin to move on their pseudo-random track. Each target position is updated approximately every 1/20th of a second having a total of 1800 tracking steps. The targets can make up to 4 adjustments (x, y, rotate, scale) per update. However, to minimize a jittering effect, a direction and a minimum duration were chosen to have a target adjust along one dimension for a period of time before possibly switching to a new direction. The duration was approximately 0.5 seconds. In addition, periodically (approximately every 4 seconds), one target would "dart off" (i.e., make much larger incremental adjustments). Thus, the targets have a non-uniform adjustment. This design encourages the subject to service the dominant deviants in order to achieve the best score as opposed to randomly servicing each widget or sequencing through each widget regardless of assessing the scene. A total of six pseudo-random tracks were pre-computed for each of the four computer targets. The ordering of the tracks were randomly shuffled for each condition. Thus, all subjects experienced the same 6 tracks a total of three times (once per input condition).

In terms of visual representations, the computer targets were drawn in a blue outline while the user's widgets were drawn in a solid, transparent red color (see Figure 6.12). The transparency was used to allow for computer and user target overlaps. Transparency was achieved using alpha-blending with a value of 0.60. The shape of the targets roughly matched the shape of the specialized input devices (stretchable square, ruler, flipbrick and mobile scrubwheel).

At the end of each trial, subjects were presented with a score of their trial. The score represents the average root-mean-square (RMS) Euclidean distance off-target for all four targets (along all dimensions: translation, rotation and scale).

For the space-multiplex conditions subjects could move their targets by physically acquiring the associated input device(s) and manipulating the device(s).

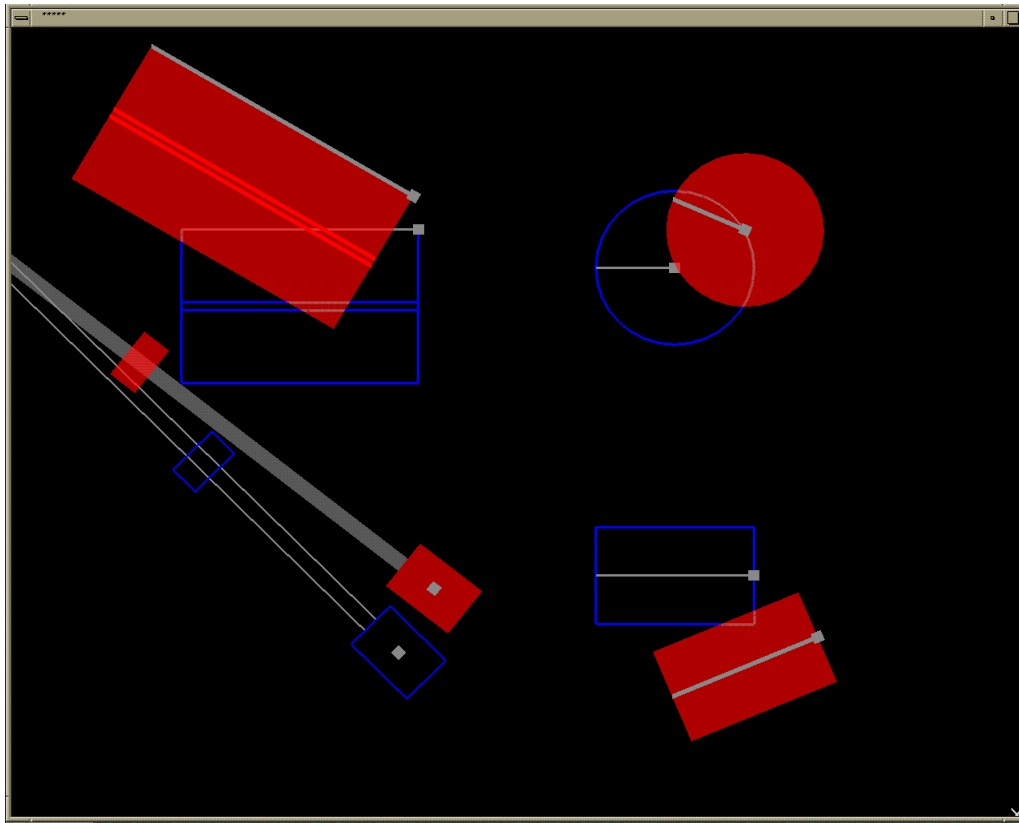


Figure 6.12. Snapshot of multi target tracking task. Computer targets are outlined in blue while the user's targets are in transparent red color.

For the time-multiplex condition two graphical cursors are visible on the screen. The puck (used in the dominant, right hand) is represented by an "arrow" cursor while the brick is represented by a "cross" cursor. Before manipulating a user widget, the subject first must acquire the widget by moving towards the widget's selection "handle" and selecting it with the puck cursor. This is achieved by pressing and holding any one of the four puck buttons. Once pressed, the user's widget becomes attached to the puck and automatically attached to the brick device. Subjects manipulate the widget and once the puck button is released, the widget is detached. Note that the selection handles appear as rectangle on the widget approximately 15 pixels wide.

Design and procedure

All twelve subjects used the three input conditions: space-multiplex, specialized devices (SpaceS), space-multiplex generic devices (SpaceG), and time-multiplex (Time). Six trials lasting 90 seconds were conducted in each of the three input conditions. A total of six, 90 second, multi-target, pseudo-random tracking path

stimuli were predefined. The ordering of the stimuli were randomly shuffled for each condition. Thus, all subjects experienced the same 6 track stimuli a total of three times (once per input condition). Subjects were assigned the sequence of input device conditions based on a Latin-square counterbalancing scheme to minimize ordering effects. For each new input device condition, subjects were given a maximum of one 90 second trial to acquaint themselves with the device and interaction technique. After the experiment, subjects were presented a questionnaire to obtain their subjective preference for each condition and to elicit comments about their experience.

All tablet motion and button events were logged with timestamps. The computer and user targets were logged on every update (approximately every 1/20th of a second). In summary, each subject performed 6 trials on each of the three input conditions resulting in a total of 18 scores per subject. Thus, the twelve subjects collectively generated a total of 216 data points.

Experimental biasing. The technology constraint of using four tablets biases the conditions in favor of the time-multiplex conditions. The 2x2 grid of Wacom tablets was necessary due to the fact that the tablets can only support two sensors on them while operating in "multimode." Ideally, we would have run the experiment on one large Wacom tablet if it could support multiple sensors (e.g., 8 or more). With the time-multiplex condition, a stronger stimulus-response (SR) compatibility exists with the input control space and the computer display space. That is, subjects move their devices and limbs in the direction they wish to acquire or manipulate a widget. In contrast, the 2x2 grid of tablets has a stimulus-response incompatibility. First, the input devices always remained on their designated tablet. In order for subjects to manipulate a virtual, logical widget, they must remember or visually search the 2x2 grid of tablets to acquire the proper physical input device. For example, the ruler logical widget may currently be in the top right of the computer display. However, the physical ruler device is located on the bottom left tablet. We believe this mismatch places an extra cognitive burden on the subject. In addition, the space-multiplex conditions were susceptible to infrequent system lags due to the multiple tablet configuration. The lag would manifest as moving a physical device but not seeing an immediate update of the users' target (up to a 1 second delay but often much less). In pilot studies, the lag was only observable in the space-multiplex, specialized device condition which generates more tablet data due to the inherent concurrency of having two sensors built into one physical device. Again, this lag

phenomena was very infrequent and biases in favor of the time-multiplex control conditions. We predict that the phenomena we wish to detect is strong enough to overcome these effects.

6.2.4 Results and discussion

Traditional tracking experiments define the tracking error at any moment as the distance between the center point of the user and computer targets. This is not sufficient for our tracking experiment that varies multiple dimensions and has multiple targets. An overall single measure of the tracking quality is necessary for feedback to the subject as well as for manageable data analysis [Zhai, 1996]. Thus, we have defined a single main dependent variable of interest, the "score," to reflect the overall tracking error of the user's 4 targets from the computer's 4 targets. Specifically, the score is defined in equations 6.1-6.8 as the root-mean-square (RMS) Euclidean distance off-target for all four targets along all three dimensions: translation, rotation and scale (see equation 6.1).

$$Score = \sqrt{RMS_{Trans}^2 + RMS_{Rotate}^2 + RMS_{ScaleWheel}^2 + RMS_{ScaleBrick}^2} \quad (6.1)$$

Each of the user's widgets have a root-mean-square (RMS) off target based on translation, angle and scale dimensions (see equations 6.3-6.5). Note that the scrubwheel and flipbrick do not have a scale component.

$$RMS_{Trans} = \sqrt{RMS_{Trans}^2 + RMS_{Rotate}^2 + RMS_{Scale}^2} \quad (6.2)$$

$$RMS_{ScaleBrick} = \sqrt{RMS_{Trans}^2 + RMS_{Rotate}^2 + RMS_{Scale}^2} \quad (6.3)$$

$$RMS_{ScaleWheel} = \sqrt{RMS_{Trans}^2 + RMS_{Rotate}^2} \quad (6.4)$$

$$RMS_{Rotate} = \sqrt{RMS_{Trans}^2 + RMS_{Rotate}^2} \quad (6.5)$$

For each trial (90 seconds, 1800 tracking steps) overall tracking performance was calculated by root mean square (RMS) error for each dimension (see equations 6.6-6.8).

$$RMS_{Trans} = \sqrt{\frac{\sum_{k=1}^N errorTrans_k^2}{N}} \quad (6.6)$$

$$RMS_{error} = \sqrt{\frac{\sum_{k=1}^N errorAng_k^2}{N}} \quad (6.7)$$

$$RMS_{error} = \sqrt{\frac{\sum_{k=1}^N errorAng_k^2}{N}} \quad (6.8)$$

At any tracking instant k , the translation tracking error $errorTrans(k)$ is defined as the Euclidean distance between the user and computer target. The $errorAng(k)$ is defined as the arc length ($arcLen = \phi \cdot length$) between the user and computer target where ϕ ranges from 0 to π and $length$ is the current length of the computer target. Finally, the $errorScale(k)$ is defined as the difference between the user and computer target lengths.

An analysis of variance (ANOVA) was conducted on the RMS score data and we now revisit the experimental hypotheses (see Appendix E). Note that pilot studies showed no ordering effect and thus our analysis does not group subjects as in experiment 1 (the degrees of freedom for subjects is 11).

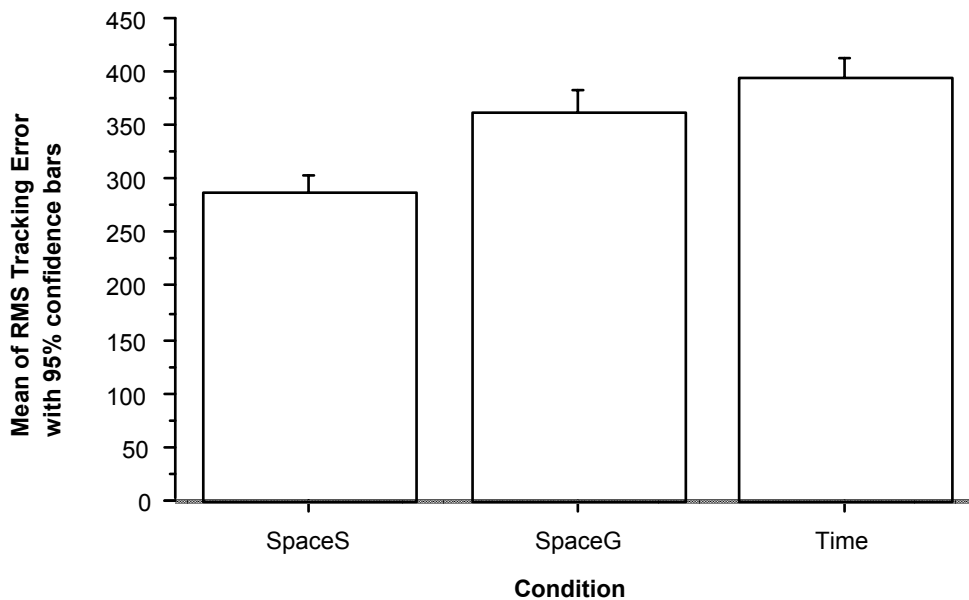


Figure 6.13. Mean RMS tracking error as a function of input device configuration.

Both of our hypotheses were supported (see Figure 6.13). We found that input condition has an effect on RMS score ($F(2,22) = 103.7, p < .001$). Specifically, the space-multiplex specialized devices condition performs best followed by the space-

multiplex generic devices followed by the time-multiplex condition. A significant difference was found when a pairwise means comparison was conducted between the conditions (SpaceS and SpaceG: $F(1,22) = 96.9, p < .001$; SpaceS and Time: $F(1,22) = 196.8, p < .001$; and SpaceG and Time: $F(1,22) = 17.5, p < .001$).

Further analysis of the data revealed how the 90 seconds worth of trial activity varied between each of the input device conditions (see Figure 6.14). With the time-multiplex condition, 45.2 seconds of the trial activity was accountable to logical widget manipulation. That is, the time when a subject has the input devices attached to a logical widget and the device is in motion (i.e., manipulating a widget). The majority of the remaining time (44.2 seconds) of the trial was dedicated to device motion without a widget attached. The bulk of this time can be considered the "switching cost" for acquiring different widgets. The remaining 0.6 seconds of the trial had no device motion. In contrast, we found that subjects in the space-multiplex, specialized device condition had 80.0 seconds of the trial accountable to device motion while the space-multiplex generic devices had only 71.6 seconds accountable for device motion. This difference is significant (pairwise means comparison between SpaceS and SpaceG: $F(1,22) = 22.75, p < .001$). In general, this suggests that roughly 10-20 percent of the time was used for switching between the physical devices.

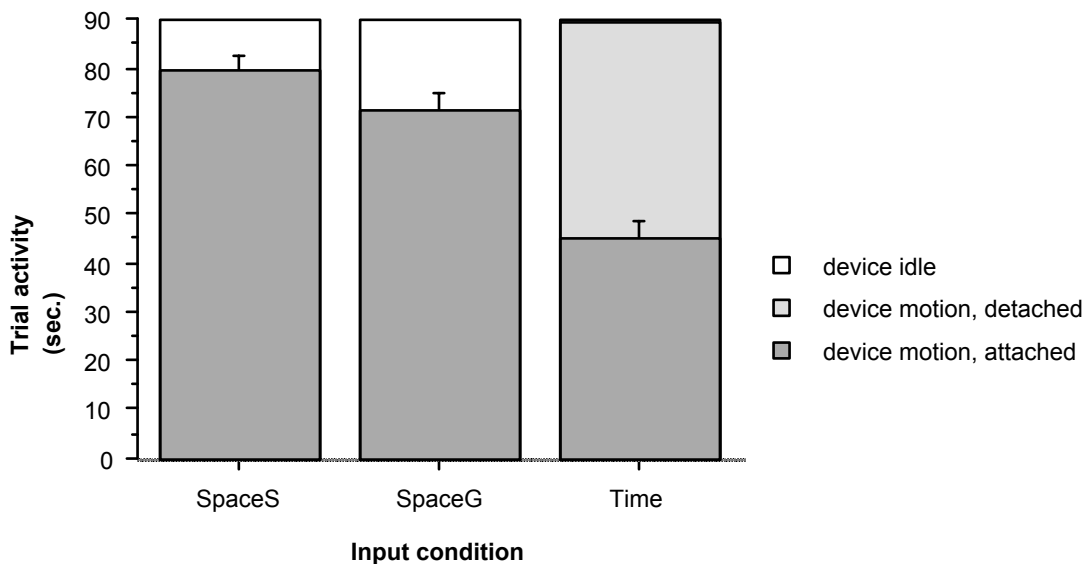


Figure 6.14. Trial activity breakdown between input device conditions.

If we examine the data by individual input device, we see a consistent trend for all four input devices across the three conditions (see Figure 6.15). This implies that our

conclusions are generalizable. All of our specialized devices had superior performance over the generic devices in both the space and time-multiplex conditions. However, a significant interaction exists between the input devices and input condition ($F(6,66) = 3.42, p < .005$). One explanation for this difference could be that some specialized devices perform better than others compared to the generic devices. For example subjects performed slightly better with the scrubwheel and flipbrick devices compared to the stretchable square and ruler devices. There are a number of competing explanations for the device differences. First, the location of the device and tablet could effect performance. Secondly, the distinct physical shapes could aid visual search when trying to acquire a device. Lastly, these results could suggest that beyond tactile mnemonics, some devices have physical affordances that facilitate the operation of the task.

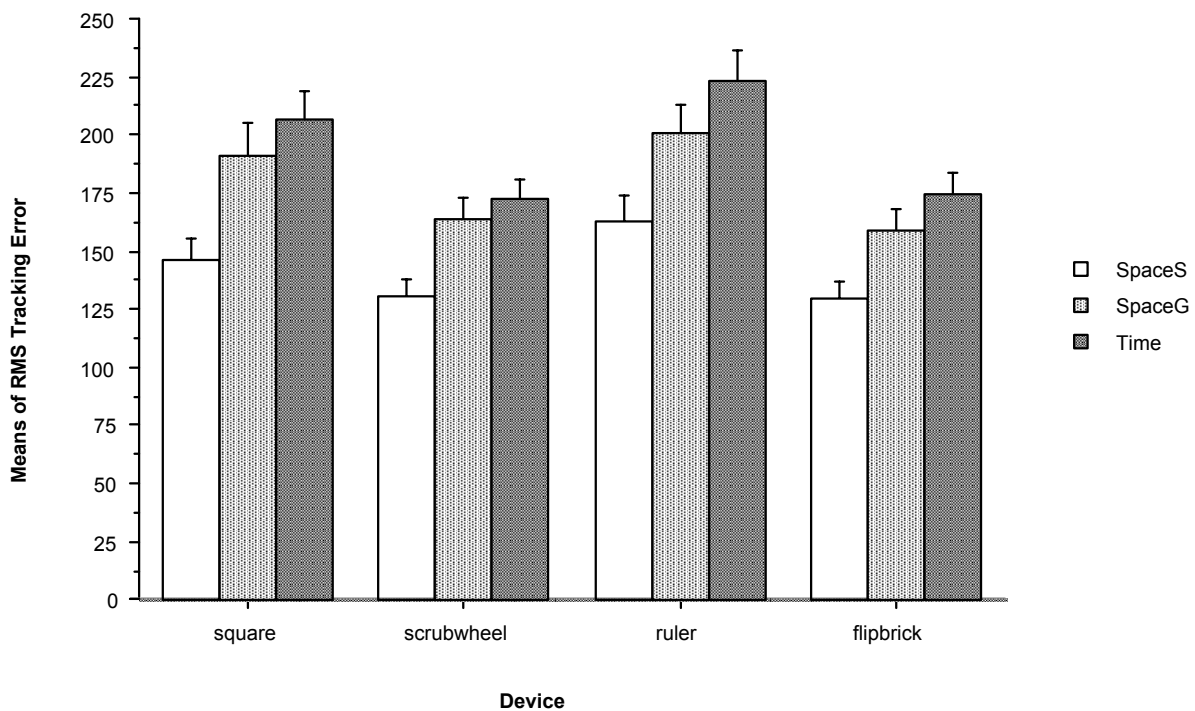


Figure 6.15. RMS tracking error by input device and condition

We were also interested in measuring learning effects across the six trials per input condition (see Figure 6.16). A significant learning effect was found across the trials ($F(5,55) = 4.8, p < .001$). There was no significant interaction between learning and input conditions. Thus, we cannot conclude that subjects exhibited different learning rates between the space or time-multiplex conditions. Indeed, due to the high variance of the data, we can not conclude much beyond the fact that learning is happening.

After the experiment, subjects were asked to quantify their preferences for each of the input device configurations. They were asked to rate the physical comfort (i.e., how fatiguing) each device was ranging from extreme discomfort to extreme comfort) as well as the ease at which they could solve the task (very difficult to very easy). A continuous scale from -2 to +2 was used for both ratings (as in Experiment 1). Figures 6.17 and 6.18 show the results.

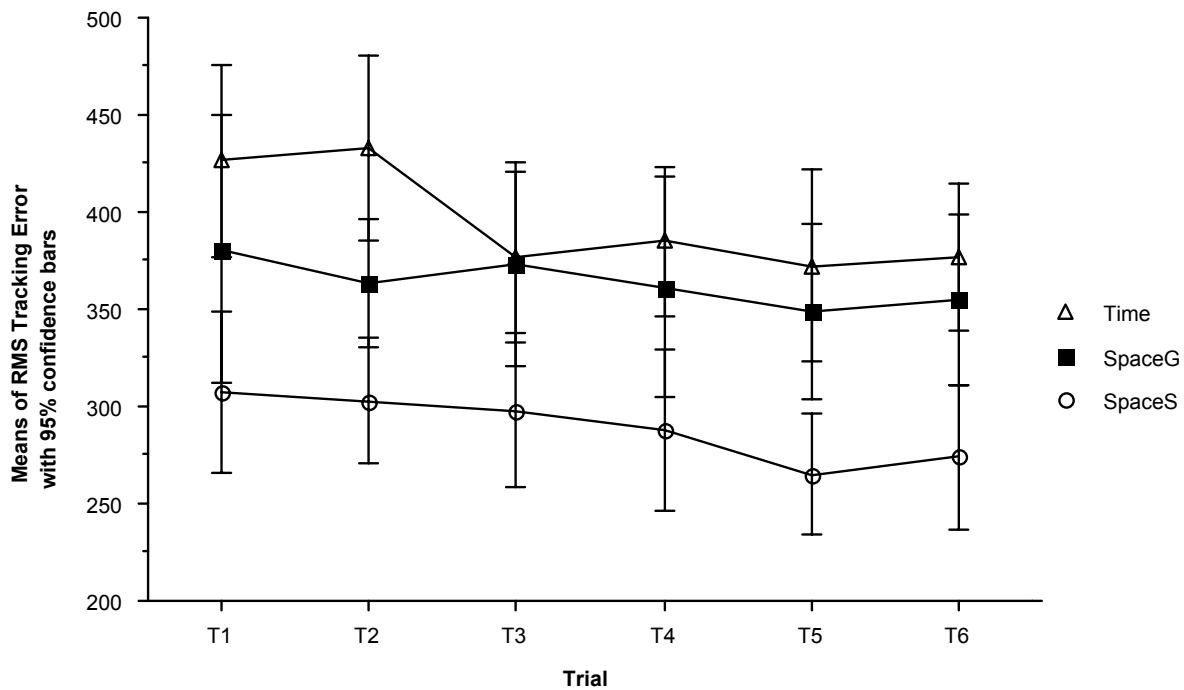


Figure 6.16. Mean RMS tracking error by trials across all input conditions (learning).

The space-multiplex with specialized devices was considered significantly more comfortable than the space-multiplex generic devices (student-t(11) = 4.61, $p < .0008$) or the time-multiplex conditions (student-t(11) = 4.15, $p < .0016$). No significant difference exists between the space-multiplex with generic devices compared to the time-multiplex condition for physical comfort.

As well for ease of use, the space-multiplex, specialized devices was viewed as significantly easier to use than both the space-multiplex, generic devices (student-t(11) = 5.74, $p < .0001$) and the time-multiplex condition (student-t(11) = 6.83, $p < .0001$). A pairwise means comparison indicates no significant differences between the space-multiplex, generic and time-multiplex conditions for ease of use.

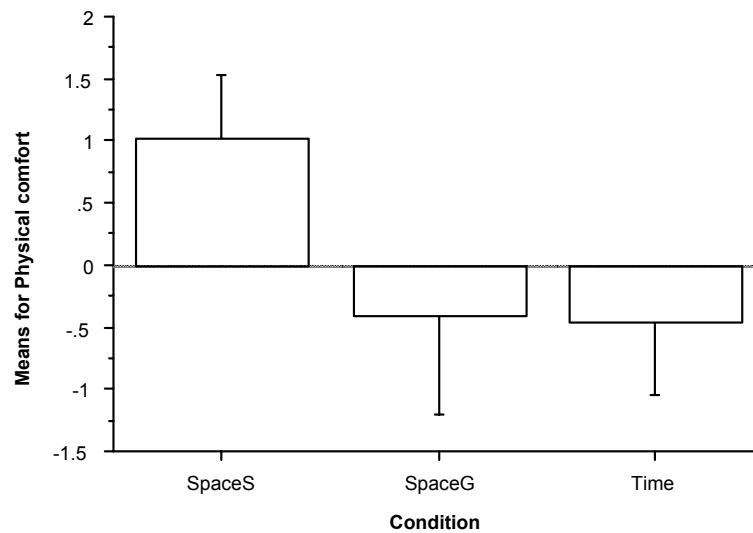


Figure 6.17. Physical comfort subjective ranking (from -2 to +2) for each input condition.

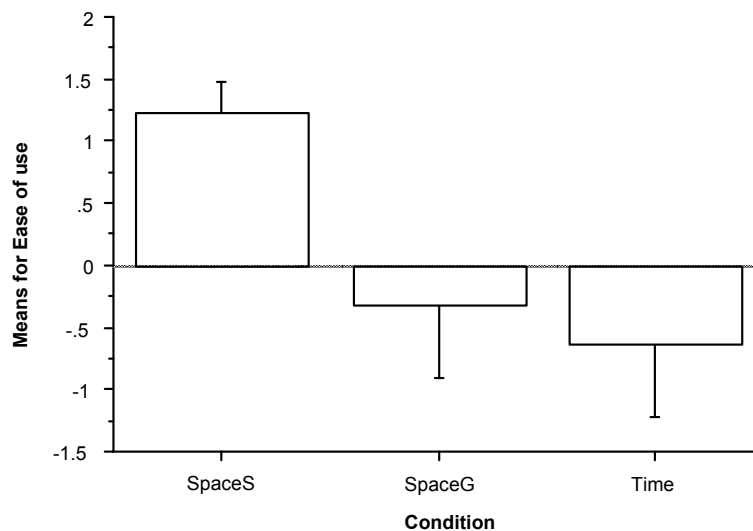


Figure 6.18. Ease of use subjective ranking (from -2 to +2) for each input condition.

Subjects were then asked to rank order the overall preferences for each input device configuration. The space-multiplex specialized condition ranked at the top preference (average ranking=1.1) followed by the space-multiplex generic condition (average ranking=2.3) and then the time-multiplexed condition (ranking=2.6).

Discussion

In general, a variety of strategies was observed throughout the experiment. The majority of the subjects used one hand to operate the specialized devices. The ruler

and stretchable square were more difficult to operate than the mobile scrubwheel and flipbrick. Some subjects keep their left hand on the ruler device and used their right hand to service the remaining three devices. It was not clear if this offered any improvement in performance. Nevertheless, all the subjects managed to operate the scrubwheel and flipbrick with one hand. Only one subject complained about grabbing the wrong input device.

In contrast, the space-multiplex, generic device conditions for the most part had subjects using two hands (one for the brick and the other for the puck) to manipulate each widget. However, at least two of the subjects used one hand to operate both the puck and brick simultaneously. We observed one subject who used one hand on the puck and drove the puck into the brick to move both of them. The graphic overlays on the tablets were designed to aid the subject in remembering what virtual widget could be controlled with a given brick and puck pair. It is not clear how frequently, if ever, the subjects used the graphic overlays. Questioning the subjects after the experiment, they claimed to make very little use of the graphic overlays. Two did say that they would look down at the tablets (i.e., graphic overlays) if they were confused. Five of the subjects complained at least once during this condition of grabbing the wrong device pairings.

In the time-multiplex condition, some subjects would occasionally attempt to select a computer target instead of the corresponding user target. This cannot be easily explained except for the fact that the multi-target tracking task is difficult. Subjects must constantly assess the scene and watch the moving targets to make a decision when to stop servicing the current widget and determine which target to service next. In contrast, the space-multiplex conditions does not suffer from mistakenly selecting a computer target instead of the corresponding user target. By using the physical devices, it is only possible to select user targets. Moreover, we believe that target acquisition is easier with physical targets than virtual targets. Physical targets can often be larger than virtual targets. Moreover, tactile feedback and mnemonics can facilitate the physical target acquisition and confirmation process.

One could argue that Fitts law [Fitts and Peterson, 1964] could serve as a model to predict our performance results of this experiment. This, however, would be misleading. In general, Fitts law defines the time to acquire a target as a function of the distance traveled (between the starting position and final target position) divided by the target size. While this has been shown to be true for rapid reciprocal

target tapping tasks, our experimental task has a number of different features: (1) requires more high level cognitive reasoning (e.g., to assess the scene and determine which and when to switch devices), (2) consists of device acquisition for the space-multiplex conditions, as well as (3) requires not only target acquisition but a significant portion of the task deals with the subject manipulating the device to perform a target tracking task.

Future data analysis is possible. For example, we could adapt the coordination metrics used in the first experiment to measure how well subjects coordinate the various dimensions in conjunction with the multiple targets and various input conditions. This analysis is beyond the scope for the current goals of this experiment. We also may speculate on whether the connected devices (e.g., specialized devices: stretchable square, ruler, scrubwheel and flipbrick) place a lower cognitive burden on the subject compared to the disconnected devices (e.g., puck and brick combination). This is again left for future research.

6.3 Summary

This chapter described two experiments that empirically investigated the property of space-multiplexing input for Graspable UIs. The first experiment focused on manipulation issues for tasks when users already have input devices acquired in their hands. Here we compare three space-multiplexed conditions with a time-multiplexed condition. As we predicted, for our task, the space-multiplexed conditions out-performed the time-multiplexed condition. The specialized space-multiplex conditions (ruler and square) performed statistically equivalent to the generic space-multiplex condition (bricks). This was not too surprising due to the nature of our task. We argue that the space-multiplex performs better than the time-multiplex conditions for a number of reasons. The space-multiplex designs (1) reduce interaction modes, (2) allow for more natural conceptual chunking and phrasing, and (3) tap into our everyday skills at physical object manipulations.

Nevertheless, we wanted to determine if the specific physical form factors for the graspable functions can be used to suggest and facilitate the functionality they offer. The second experiment again focused on the issue of space-multiplexed versus time-multiplexed input but examined the inter-device transaction phase of interactions. That is, the experiment was designed to study the relative costs of acquiring

physical devices (in the space-multiplex conditions) versus acquiring virtual controllers (in the time-multiplex condition). The experiment showed that space-multiplexed input provided a significant performance improvement given our task. Furthermore, the specialized physical form factors out-performed the generic devices within the space-multiplex conditions. There are a number of competing explanations for the specialized versus generic device performance differences. First, the location of the device and tablet could have effected performance. Secondly, the distinct physical shapes could have aided visual search when trying to acquire a device. Lastly, beyond tactile mnemonics, some devices have physical affordances that facilitate the operation of the task. These issues could be teased out in future experiments but they all suggest that there are significant benefits for using specialized input devices.

Chapter 7: Conclusions

7.1 Summary

This dissertation has defined and explored Graspable User Interfaces, an evolution of the input mechanisms used in graphical user interfaces. Graspable UIs provides users concurrent access to multiple, specialized input devices which can serve as dedicated physical interface widgets, affording physical manipulation and spatial arrangements. Like conventional GUIs, physical devices function as “handles” or manual controllers for logical functions on widgets in the interface. However, the notion of the Graspable UI builds on current practice in a number of ways. With conventional GUIs, there is typically only one graphical input device, such as a mouse. Hence, the physical handle is necessarily “time-multiplexed,” being repeatedly attached and unattached to the various logical functions of the GUI. A significant aspect of the Graspable UI is that there can be more than one input device. Hence input control can then be “space-multiplexed.” That is, different devices can be attached to different functions, each independently (but possibly simultaneously) accessible. This then affords the capability to take advantage of the shape, size and position of the physical controller to increase functionality and decrease complexity. It also means that the potential persistence of attachment of a device to a function can be increased.

We are proposing a conceptual shift in thinking about physical input devices not as graspable *devices* but instead as graspable *functions*. In the traditional sense, almost all physical input devices are “graspable” in that one can physically touch and hold them. However, in this thesis we have explored the utility of designing the physical devices as graspable functions. This can best be shown in Figure 1.2 (redrawn as Figure 7.1 for the reader's convenience). With traditional GUIs there are often three phases of interaction: (1) acquire physical device, (2) acquire logical device (e.g., a UI widget such as a scrollbar or button) and (3) manipulate the virtual device. Alternatively, with Graspable UIs, we can often reduce the phases of interaction to:

(1) acquire physical device and (2) manipulate the logical device directly. This is possible because the physical devices can be persistently attached to a logical device. Thus, the devices serve as dedicated graspable functions.

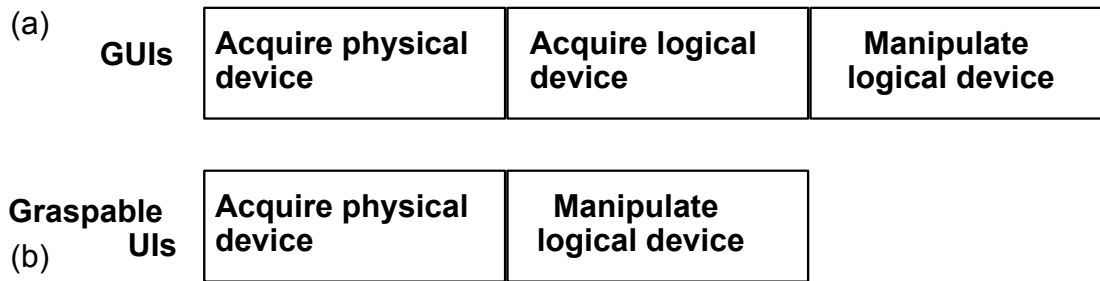


Figure 7.1. Phases of interaction. In (a) traditional GUIs have 3 phases of interaction while Graspable UIs (b) often need only 2 phases (removing the need to acquire the logical device).

Graspable UIs attempt to tap into the a user's existing skills at manipulating physical objects. These manipulations are possible by knowledge we have learned through a lifetime of practice. Our innate motor abilities, sense of touch and texture discrimination, and our everyday skill in grasping, gesturing and manipulation all contribute to the performance gains of Graspable UIs. The challenge lies in designing efficient Graspable UI objects that minimize the switching costs to acquire objects as well as minimize the learning needed to understand the relationship between the physical manipulation and corresponding virtual action.

We began this thesis by reviewing relevant motor, perceptual and cognitive psychology literature which provided the underlying theoretical support for workable Graspable user interfaces (Chapters 2). Next, existing input devices and research systems which exhibit some early traits of Graspable UIs were surveyed (Chapter 3).

We then motivated, and applied the five design properties for Graspable UIs in the context of a commercial software animation program. A byproduct of this process is the development of a new input device (the mobile scrubwheel) and novel interaction techniques (e.g., the time control mappings for the space mouse 6 degree of freedom input device). The five properties are summarized in Figure 7.2.

Next, we described a detailed implementation and case study for a specific set of graspable user interfaces which we call "bricks" (Chapter 5). First, a series of

exploratory studies was conducted to motivate and investigate some of the brick concepts. Primarily, we wanted to gain insights into the motor-action vocabulary for manipulating hand-scaled input devices on a desktop surface. After outlining the basic bricks design, we describe three prototype systems and applications: (1) a simple drawing program, GraspDraw, (2) curve editing within the context of a more robust commercial application and (3) flipbricks. Throughout these case study we set out to gain further design experience with the 5 Graspable UI design properties of (1) space-multiplex input and output, (2) concurrency, (3) physical form (weak general vs. strong specific), (4) spatially-aware devices and (5) spatial device reconfigurability.







Generic scale		Graspable UI Properties				
		Space-multiplexing input	Concurrency	Physical form	Spatially aware	Spatial reconfigurability
low  high		transient, always reassign, time-multiplexing	1	generic	unaware	permanent location
		3/4 time reassign	occasionally 2			stationary
		1/2 time reassign	2			track
		1/4 time reassign	3			tethered
		permanent never reassign	more than 3			specific

Figure 7.2. Graspable UI defining properties

Finally, Chapter 6 presents two experiments that empirically investigated the property of space-multiplexing input for Graspable UIs. The first experiment focused on manipulation issues for tasks when users already have input devices acquired in their hands. Here we compare three space-multiplexed conditions with a time-multiplexed condition. As we predicted, for our task, the space-multiplexed conditions out-performed the time-multiplexed condition. One could argue that the

performance difference we observed was mainly attributed to the parallel activities permitted in the space-multiplex condition. This issue was factored out in the follow-on experiment.

The second experiment again focused on the issue of space-multiplexed versus time-multiplexed input but examined the inter-device transaction phase of interactions. That is, the experiment studied the relative costs of acquiring physical devices (in the space-multiplex conditions) versus acquiring virtual controllers (in the time-multiplex condition). The experiment showed that space-multiplexed input provided a significant performance improvement given our task. Furthermore, the specialized physical form factors out-performed the generic devices within the space-multiplex conditions. That is, the graspable functions (i.e., input devices) suggested and facilitated the functionality they offered. The experiment offers proof that the "strong-specific" design of input devices can, in many cases, outperform the "weak-generic" designs.

7.2 Contributions

The main contribution of the thesis is the defining and exploration of Graspable user interfaces. Specifically, the contributions can be summarized as follows:

- A set of 5 design properties for Graspable UIs.
- Showed how the Graspable UI can be derived and applied for a complex spatial task.
- Explored the difficulties and issues of implementing Graspable UIs in our current commercial environment with current transducers and spatially-aware devices.
- Experimental comparison between space-multiplex and time-multiplex input schemes. We showed that a space-multiplex design performs better than a time-multiplex input design.
- Experimental comparison showing the advantage of having specialized physical form factors for input devices within a space-multiplex input scheme.

- Proposed and illustrated the utility of having spatially-aware computational devices (such as the bricks).
- Created novel input devices and interaction techniques derived from applying the design properties for Graspable UIs. Such devices and techniques include: the flipbrick, mobile scrubwheel, and the space mouse for temporal control through dynamic media.

7.3 Limitations, challenges and open issues

Physical and cognitive clutter. One of the design concerns with Graspable UIs is the opportunity for physical and cognitive clutter in the workspace. In terms of physical clutter, how many objects should be present to avoid clutter? This is almost entirely dependent on the task. One may argue that at most two objects be present (one for each hand). Alternatively, if one is sculpting shapes using a particle system representation, where hundreds of points are used to define a 3D surface model, perhaps there should be one physical object per particle. This could mean the user has access to hundreds of physical objects at any given time. Having too many graspable objects and thereby creating a challenge to find the one you are looking for may obviate any performance benefits they offered to begin with. Moreover, can our interface designs handle the situations when users lose their graspable objects? Here, we can suggest having dual representations for the task such that it can always be done with one or more specialized graspable devices but can still be done, perhaps more clumsily, with a generic input device (i.e., the mouse).

Both physical and cognitive clutter should be avoided when designing interactions. By cognitive clutter we mean cognitive overload arising from physical clutter and object discrimination problems. This type of clutter is perhaps more difficult to measure than physical clutter. However, one way of measuring clutter is through the use of search tasks. As with physical clutter, cognitive clutter may be induced by having too many similar type objects in a conceptual workspace. Both types of clutter run the risk of interfering with the intended benefit of providing externalized representations for a given task.

Physical vs. Virtual. While we advocate that the Graspable UI externalize some of the internal computer representations, which interface components should be physical and which should remain virtual? This is, once again, an important design issue in which there are no concrete rules. In general, however, one may tend to

physically instantiate those interface components which are very static in nature (e.g., tool icons or menus). Highly dynamic, visually demanding interface elements should remain in virtual form since the computer screen is very good at updating and displaying the dynamics.

Physical intermediaries. We could also eliminate graspable objects and instead use only our hands as physical input devices (as in systems like Videoplace or Multipoint Control [Kruger, 1991]). While this may be useful for some applications, in general using a physical intermediary (e.g., brick or stylus) may be more desirable (by physical intermediary we mean a physical device that operates between the user and the computer. Touch screens are an example of input that does not require any physical intermediary objects). We argue that having graspable objects serving as physical intermediaries between the user and the computer has important interface value. First, physical intermediaries like the scrubwheel prototype can both suggest and facilitate their functional operation. Secondly, the intermediary devices consume space (i.e., footprints) and allow for nonlinear interpretations of user's movements which can enhance the resolution and interaction of the user with the virtual environment. This is harder to support with gestures alone. Devices can also constrain movements and maintain relationships which the freeform flow of hands cannot. Finally, using only hand gestures (i.e., no physical intermediaries) is difficult as hand gestures lack very natural delimiters for starting and stopping points. This makes it difficult to segment commands and introduces lexical pragmatics. In contrast, the affordances of touching and releasing a physical object serve as very natural start and stop points.

System support. There are a number of system requirements that emerge when supporting a Graspable UI. First, operating systems should become more sophisticated in sensing and installing new devices (i.e., device drivers) when graspable objects (i.e., devices) are constantly being added or removed from the input control space. Currently, most computer systems have a very primitive model of device drivers which a system administrator must install by hand. Moreover, the system should support reassignable device drivers. When graspable objects are placed on the desktop, they must be recognized by the system and their communication protocols interface dynamically loaded and added to the pool of current system devices. This all should happen without having to rebuild the operating system kernel, rebooting the system or logging in and out (e.g., to restart

the X11 server). Computer network models may offer some design solutions when we consider a desktop-area-network for all of the graspable objects operating on an input surface.

Hardware support. In terms of input devices, we need cheap, rugged, wireless physical objects that serve as graspable objects for the interface. For sensing resolution, we can work with a wide range of granularity and dimensionality. Some devices could be very sensitive and sense fractions of a millimeter of 6 DoF motion with very high update rates. Alternatively, we could have some primitive sensing devices in which the system only detects the presence or absence of an object in the input space. The idea is that there is an economy of input devices in that all do not need to be highly sensed by the system.

Finally, the use of graspable objects may provide an interesting opportunity to the software industry which could sell specialized graspable objects along with their applications. One primary advantage to this approach may be as a deterrent to software pirating. That is, customers could copy the software, but not be able to operate the interface efficiently without the graspable objects.

7.4 Future work

We wish to continue to investigate and refine the concepts behind Graspable UIs. Specifically, we want to continue to explore our design space of Graspable UI properties. While this dissertation has focused on the space-multiplex input and physical form properties there remains the three additional properties of concurrency, spatial-awareness and spatial reconfigurability of devices to be researched in more depth. Still more study needs to be done on more formally classifying what specific tasks, in general, are the Graspable UIs most suited for.

In addition, we are interested in exploring the simultaneous use of multiple, free-ranging graspable objects. Technology is almost available to allow us to do this efficiently. Two promising areas are computer vision techniques [Schneider, S.A., 1990] and electric-field sensing [Zimmerman, et. al., 1995]. In addition we wish to look at interactions that last longer than fractions of a second, that is, having dedicated graspable objects that have a persistent attachment to a virtual object for durations of hours, days, weeks and months (e.g., graspable objects as handles to files). The concept of physically composing graspable objects is also left for future

exploration. Finally, we would like to expand our ideas to much larger interaction granularities such as graspable objects operating at a room or building scale.

7.5 Closing remarks

Our research into Graspable UIs encourages us to reflect upon how we classify input devices. Many of the traditional design spaces characterize input devices based only on the physical properties sensed and displayed, or they are organized based on the human sensory system alone. Instead we should consider input devices along multiple relationships: (1) inter-device relationships, (2) device-user relationships and the environment in which the devices operate.

From a designer's perspective we argue that it is no longer sufficient to design systems with only two input devices in mind (one for the dominant and one for the non-dominant hand). Furthermore, the input devices are not just “pointing” devices. Instead, the Graspable UI philosophy views input devices as handles to virtual objects and functions. A collection of input devices should be available that require minimal overhead to activate and manipulate them.

Designers also should consider interaction techniques that span both the virtual and physical domains. That is, designers can create virtual widgets (e.g., buttons, scrollbars) but also generic physical widgets or specialized widgets that operate on virtual objects.

We argue that the affordances of the physical handles are inherently richer than what virtual handles afford through conventional direct manipulation techniques. With Graspable UIs, a physical handle can be assigned to a virtual object until it is detached. Thus, the physical to virtual object association persists across many interactions. The physical handle acts as a persistent selection mechanism which is made active by a user touching the physical object. Having simultaneous access to multiple physical handles facilitates two handed interactions as well as providing parallel access and manipulation of interface controls. Moreover, the physical handles can be spatially arranged in a user's workspace to facilitate task workflow and rapid task switching. In short, the “directness” in the direct manipulation interface is enhanced through the affordances of the physical object and through the persistent attachment to the virtual objects.

We believe this thesis proposes a significant advance in human-computer interaction. Consider the analogy of transmitting text messages using a singular

contact switch to send Morse code (a time-multiplex design) vs. a QWERTY keyboard (a space-multiplex design). What this thesis proposes is the analogy of transmitting graphical and spatial information from a mouse (time-multiplex) to Graspable user interfaces which offer a space-multiplexed design.

Appendix A: An overview of Prehension

In this thesis we are mostly concerned with new, hand-scaled input and output transducers for the computer. Prehension plays a critical role in understanding the relationship between the hands and physical object grasps and manipulations. This section describes in detail some of the theory underlying movement and the high versatility of our hands. We first describe what prehension is and why it is relevant for human-computer interactions. Next we outline the phases of prehension and then conclude with prehension experiments that are particularly relevant for graspable interfaces.

A.1 What is Prehension

prehension: n. 1. a taking hold; a seizing, as with the hand or other limb.
2. mental apprehension. From the Latin *prehendere*, to take or seize.
(Webster's New Twentieth Century Unabridged Dictionary, 2nd Edition).

In the context of motor psychologists, neuro-physiologists, and kinesiologists, the definition of prehension can best be described as “the application of functionally effective forces by the hand to an object for a task, given numerous constraints [MacKenzie and Iberall, p. 6].” The study of prehension is critical to understanding the needs, constraints and design issues involved in building input devices for the hands.

From the most abstract level, our hand behavior is determined by a controller which accepts objects and tasks as input and generates prehensile behavior in the form of hand postures and forces over time (see Figure A.1).

One of the primary goals with prehension is that the object not be dropped so the establishment and maintenance of a “stable grasp” is of paramount importance. Potential instabilities and perturbations may occur through the task which must be compensated for by the hand.

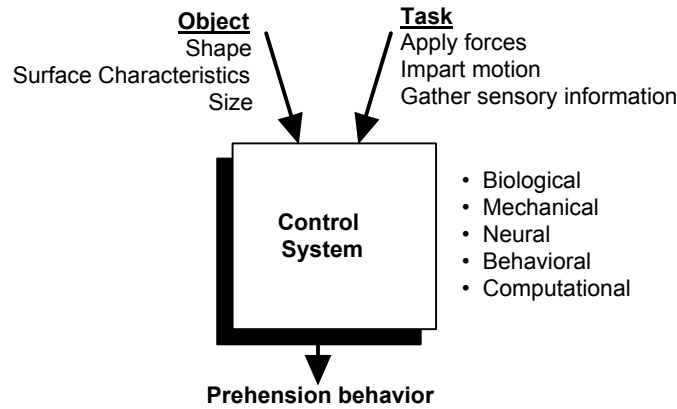


Figure A.1. Hand controller model accepts objects and task as input and generates prehensile behavior as output (from MacKenzie and Iberall, p. 7)

In general, we have two main types of grasps: power and precision grasps. While there are many subclassification schemes, Figure A.2 shows one classification that illustrates the variety of grasps that we are capable of selecting from to match the characteristics of the object and task. We can begin to appreciate just how well designed our hands are.

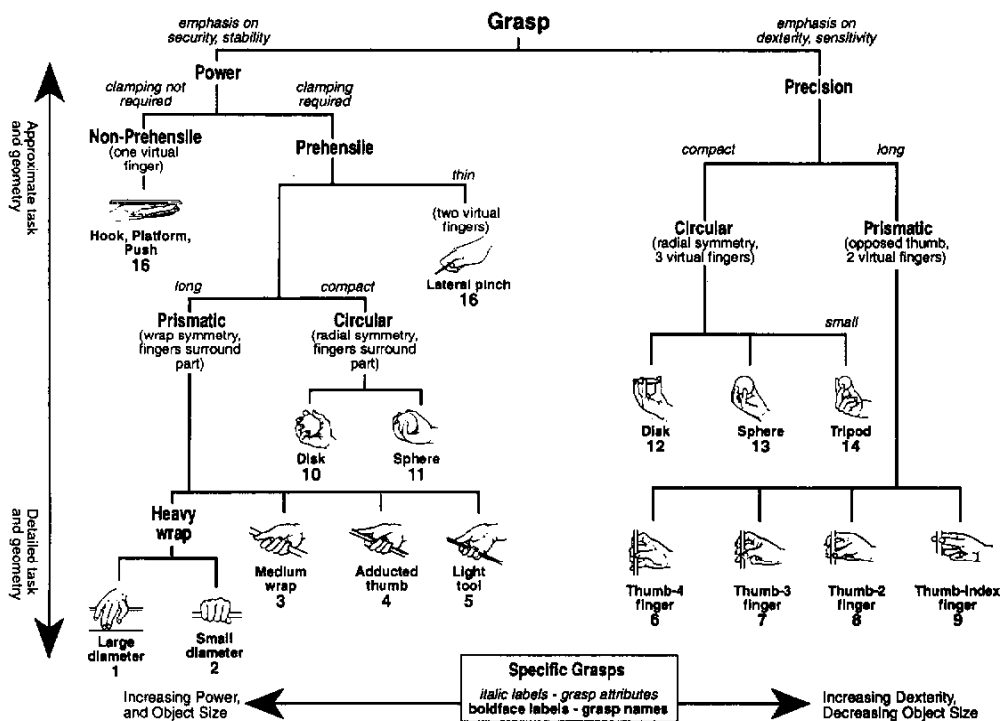


Figure A.2. Grasp classification. Power grasps are shown on the left while precision grasps are shown on the right. Originally from (Cutkosky and Howe, 1990) and adapted by [MacKenzie and Iberall, 1994].

Object properties are perceived in hand-sized dimensions. Moreover, objects that we grasp for have two types of properties: intrinsic and extrinsic properties.

“Intrinsic object properties are the physical identity constituents of objects, such as size, weight and shape. Extrinsic object properties are spatial properties of objects in an egocentric body space, such as distance, orientation with respect to the body, and, if in motion, direction and velocity of the object. [MacKenzie and Iberall, p. 76].”

Finally, our hands often require the use of tools which serve to increase the *strength* or *precision* of our hands. Input devices for computers can also be viewed as tools. Effective input devices strengthen and increase the precision of our interaction with the computer.

A.2 Phases of Prehension

In general, there are three main phases of prehension: planning, moving before contact, and during contact. Each will be briefly described next.

Planning of prehension

The planning of prehension simply is the “preparatory processes related to the organization and planning of the upcoming movement [MacKenzie and Iberall, p 63].” The planning process involves three components: “(1) perceiving task-specific object properties, (2) selecting a grasp strategy, and (3) planning a hand location and orientation [MacKenzie and Iberall, p. 63].” We have built up an extensive knowledge base of the intrinsic properties of everyday objects and how they can be expected to behave when grasped. This knowledge is used extensively to plan our reaching and grasping actions. For example, when we want to grasp a mug, we reach for the handles because we do not want to get scalded by any hot liquid inside the mug, plus we realize that we need to bring the mug close to our mouth and potentially tilt it to drink. We place a finger on the side of the handle to reduce torque. All of these actions are in anticipation of the object behavior during our interaction.

Movement before contact

There are two basic phases of prehension: a fast (high velocity) phase where the fingers preshape and a slow (low velocity) phase where contact of the object is made. This was experimentally shown by Jeannerod [1984]. The first phase lasts approximately 70% of the total movement time. During movement the grip fingers

reach their peak aperture also roughly at the 70% time for the total movement. A generalized model of movement before contact prehension can be seen in Figure A.3. The two phases of prehension are often called Phase 1: “ballistic movement” or “open-loop” and Phase 2: “adjustment” or “closed-loop”.

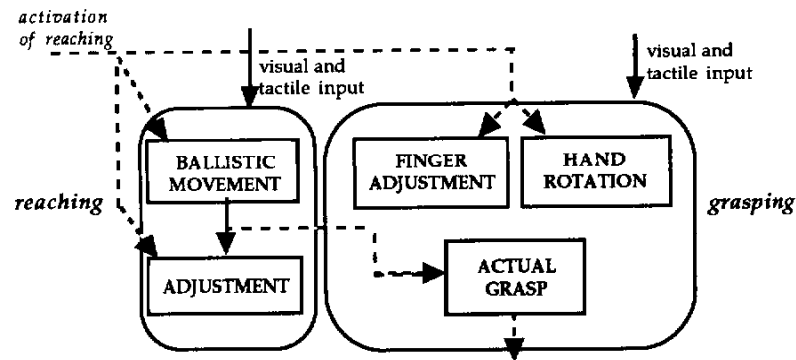


Figure A.3. Motor control model of movement. [MacKenzie and Iberall, 1994, p. 110]

During the ballistic phase of movement, the hand and palm orient themselves into a preshape grasp for anticipatory object contact. This is also considered as an anticipatory *feedforward* control phase.

During contact

Once we have grasped an object, we are very adept at discriminating object properties such as object length and weight. We are also very good at exploring and extracting salient object features such as texture, hardness, temperature, weight, and volume [Lederman and Klatzky, 1994]. During object contact, we are often interested in maintaining a stable grasp on the object which requires transmitting forces through the fingertips in order to counterbalance the weight of the held object. Maintaining a stable grasp also involves resisting perturbations by external forces (e.g., when a hammer comes in contact with a nail).

Also, there is often a distinction made between a static grasp that is used for holding and transporting an object versus a dynamic grasp for manipulation. The dynamic grasp has four forms of manipulation: fixed contacts, rolling contacts, sliding contacts and repositioning or regrasping [Elliott and Connolly, 1984].

A.3 Studies on Prehension

There have been many experimental studies in the field of prehension. We now describe one particular study which is relevant to the concept of graspable interfaces.

Marteniuk, et al., [1987] showed how task intention, context and object properties affect timing parameters for prehensile movements. In the first experiment they varied the goal (i.e., point or grasp the object). In the second experiment they varied object fragility by asking subjects to grasp a tennis ball or light bulb. The third experiment varied task intent by asking subjects to grasp an object and then fit it into a hole or throw the object. The findings revealed that the velocity deceleration phase was longer for grasping than pointing, for grasping a light bulb than a tennis ball, and for fitting rather than throwing. Less variability was observed between the conditions during the ballistic phase of movement. While the timing differences are relatively small between the conditions, it suggests that context, intent and object properties factor into our prehensile behavior.

Perhaps what prehension best reveals is that predictive models and performance evaluation of input devices need to acknowledge the finely tuned processes that occur before, during and after the hands make contact with input devices.

Appendix B: Design variations for Bricks

B.1 Bricks without dynamic virtual context

In some of the examples described in Chapter 5, the bricks operate on the Active desk, a highly dynamic surface presenting context information. The bricks may also be valuable if they are placed in our everyday physical environment (see Figure B.1). They could be stored anywhere; spatial organizations and spatial memory can be used. That is, the bricks would serve as external memory aids.

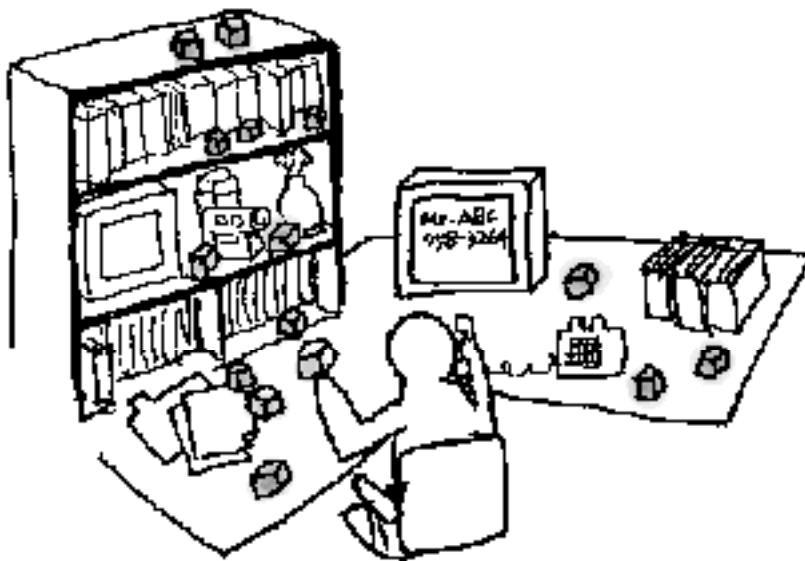


Figure B.1. Bricks can be placed in our environment to serve as external memory aids and potentially facilitate epistemic actions.

For example, placing a brick near the telephone while talking on the phone may store the caller's telephone number. The user could then move the brick to the bookshelf or near a file folder as a reminder to do something (e.g., check the references for a paper). The physical presence of the brick reminds the user of the "to do" task. Later, as a convenience, the same brick could be used to call back the caller by placing the brick next to the telephone (the brick would transmit the stored telephone number). Thus, placing the bricks in our working environment would

allow them to be associated within a given context or situation; their location will not be random but instead serve to simplify (1) choice, (2) perception, and (3) internal computation.

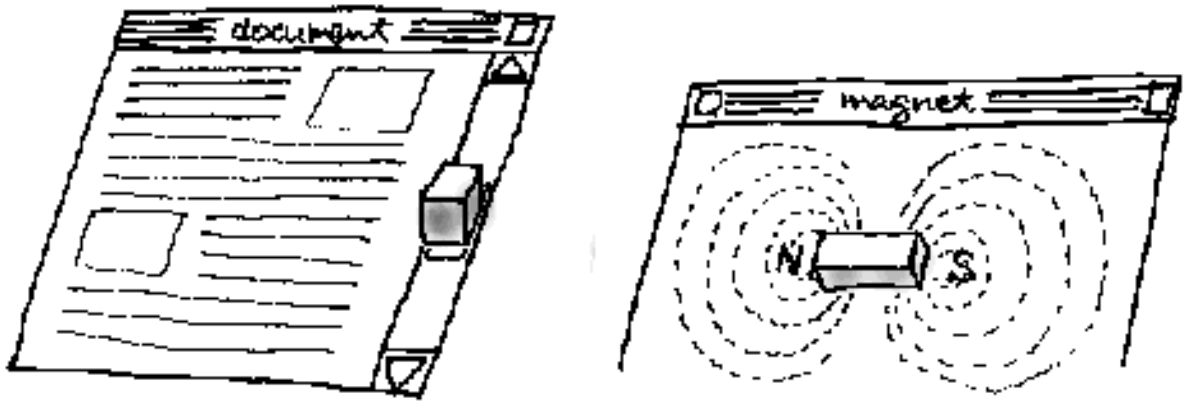
B.2 Not just bricks

While the term "brick" is used to describe the physical handle, the ideas are meant to include everyday objects (such as miniature ship and plane models, rulers, erasers, or just about any solid object that can be "recognized" by the system). The use of everyday objects may make it easier for users to recognize or realize the associated function that has been attached to the object. One danger to note, however, is that the everyday physical objects have capabilities which must be supported; otherwise, the physical object may give the user too high functional expectations.

B.3 Revealing affordances of virtual and graspable objects

What the graspable objects (i.e., bricks) attempt to do is merge the physical and virtual affordances. As a result, we can begin to think of hybrid objects. For example, a scrollbar may have a physical brick serve as its elevator thumb (see Figure B.2). Alternatively, we could define a special dedicated "scroll" brick which could be placed anywhere on the document to perform the scrolling action (i.e., not just on the scroll bar slider). The electronic medium may be used to express properties of the physical elements. Continuing with the idea of revealing affordances of physical objects, the virtual medium could provide visual cues for expressing a range of influence a physical object has. For example, magnetic rings that surround a physical brick may show the user the object's sphere of influence. Artifacts outside the magnetic rings are not affected (see Figure B.3).

In this case, the electronic magnetic rings show more than just a range of influence and indicates that there is a difference between the two ends of the physical brick (i.e., North and South ends). If we only want to indicate a sphere of influence we can draw a set of virtual concentric circles starting from the center of the physical brick. Thus we must be careful not to assign meaningless virtual (as well as physical) properties when designing feedback and interaction techniques.



Figures B.2 and B.3. A hybrid object is shown in Figure 5.28. An electronic document and scrollbar can have a physical brick serve as the elevator thumb. Figure 5.29 shows a physical brick with graphical magnetic rings to symbolize the object's sphere of influence.

Additionally, these hybrid objects will interact with one another and the physical or virtual appearances may suggest compatible types or operations. For example, one brick may be a spell checker tool which electronically appears as a wrench. A text document which has a brick as a physical handle may have a virtual appearance as a nut. The wrench and nut are compatible and thus one may operate on the other. Perhaps even twisting the wrench object one way would engage the spell checker while twisting it the other may undo the results. Incompatible objects with the spell checker appear in other forms (e.g., nails). The electronic visuals reveal more detail on the affordances of the physical artifact (see Figure B.4).

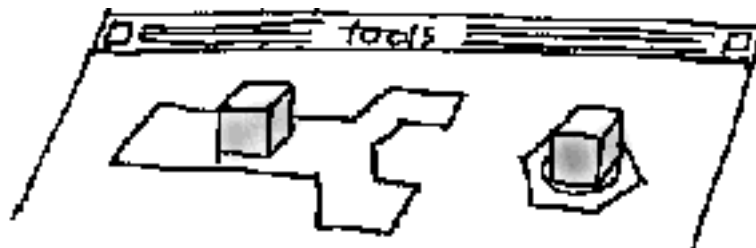


Figure B.4 Bricks can operate on other bricks. Here the electronic wrench is shown to indicate that this brick can only interact with other bricks being displayed as "nuts."

B.4 Virtual task to Real task back to Virtual tasks

Much of our design requires that the virtual objects follow the bricks in real time; some tasks may not need this real-time property. For example, a virtual world could contain a set of file icons. The user may request that the icons be "transferred" to

physical bricks for him to manipulate and classify. When this process is done, the user can request that the system read the brick layout and transfer the physical items back into their electronic (or virtual) form. Here the virtual icons inherit the final physical bricks layout. All of the intermediate brick movements are not registered with the system (see Figure B.5).

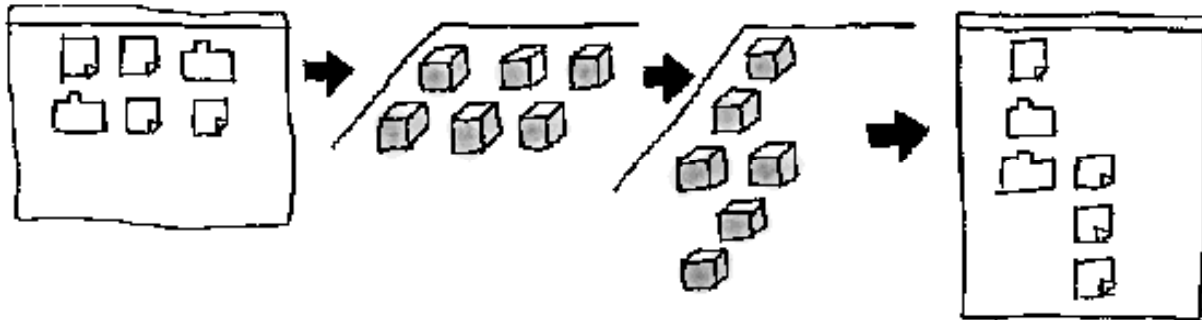


Figure B.5. Virtual objects are transferred to physical bricks. The bricks can be manipulated off-line. When finished, the brick information (layout, etc.) can be transferred back to virtual objects.

B.5 Wiping with a brick

Not only is position and orientation information useful at an instance in time, but also in an interval of time. Tracking a brick in an interval of time can also be called a wipe action (or wiping). A simple use of this idea is to designate one brick as a filter. When the filter brick is wiped across a surface, those items matching the filter query are highlighted (see Figure B.6).

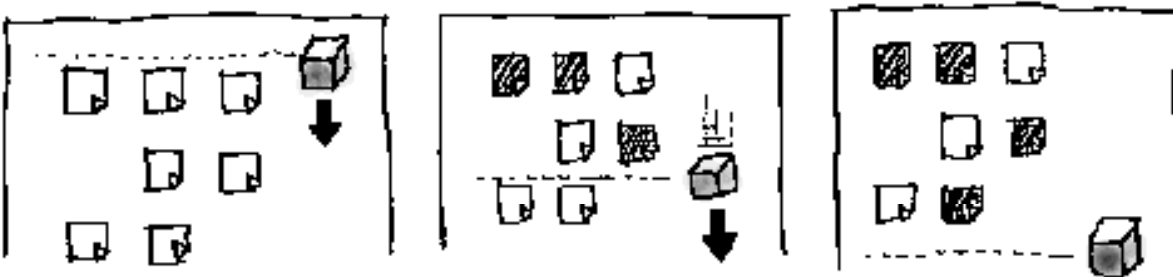


Figure B.6 Time sequence of wiping action of brick. As the filter brick is wiped from the top to the bottom of the display surface, the items matching the filter query are highlighted. An electronic "scanline" is provided for additional feedback.

B.6 3D controller brick for Toolglass

Bricks can be used for 3D interactions as well as 2D. Toolglass sheets and Magic Lenses [Bier, et. al., 1993] can be controlled by a special 3D Toolglass brick in the non-dominant hand. This brick serves as a handle for sheets and lenses (Figure B.7).

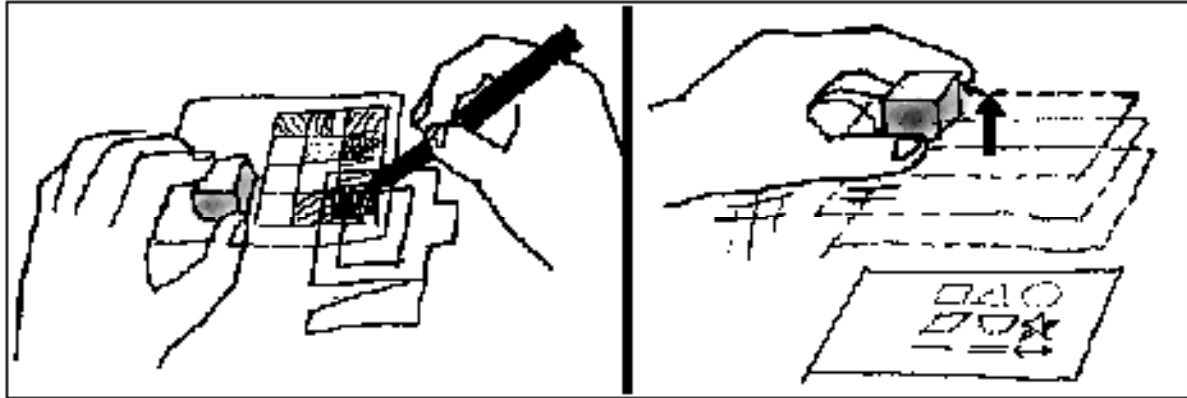


Figure B.7 A brick can be attached to a Toolglass palette (here a layer of "fill patterns" is shown) or Magic lenses. User browses through catalog of layers by raising and lowering the brick. Clicking the button on the Toolglass brick selects a layer.

Holding and moving the brick in one's hand causes the sheets and lenses to be moved and oriented with the brick. The brick can also be made aware of its height above the desktop surface. This dimension seems naturally to be used for scaling a Toolglass sheet or Magic Lens or for zoom controls. Adding a thumb button to the end of the brick makes selection articulation more explicit. Alternative button arrangements may be investigated. For example, we could place pressure buttons on the sides of the brick so that a squeezing action causes a selection to be made. Finally, we could imagine that a set of sheets or lenses are stacked like a pile of pancakes and attached to the brick. Raising and lowering the Toolglass brick above the desktop surface allows the user to browse each layer and select the desired one by pressing the brick button (see Figure B.7). Note that the usage of the 3D Toolglass brick is somewhat ideal in the sense that the majority of the time the brick glides on the 2D desktop surface and only occasionally takes flight. This will minimize the onset of fatigue.

B.7 System Reciprocity: Self-Propelled bricks

We want bricks to act not only as input devices but as output devices in terms of its location and orientation. Not only do we want visual or tactile feedback but also position and motion feedback. That is, some applications may benefit from having

bricks be self-propelled instead of always using our hands to move and orient the bricks.

System reciprocity exists in many graphical computer interfaces today. For example, users have the means of selecting and dragging icons on a computer desktop. However, the system has the same ability to move the icons itself without human intervention. This can be shown on the Macintosh desktop "Clean Up Window" option under the "Special" menu which tidies up icons in the current window by moving them around and aligning them in columns.

Consider a file management system which uses bricks to contain files and has action bricks such as "Print file" like with the LEGO wall. The user can move the action brick "Print" next to a file to be printed. By the concept of system reciprocity, as the file is being printed the "Print" brick should slowly move away from the "file" brick to indicate its status, similar to the "percentage done bar" on Macintoshes. The point is that if a system is designed where physical proximity binds operator to operand, then the system itself needs to be able to affect the proximity of objects for proper usage feedback.

For example, the Phantom electronic chess system (see Chapter 3) dramatically begins to illustrate the concepts of self-propelling bricks, position and motion feedback and system reciprocity. Chess pieces can be grabbed by the computer due to embedded magnets in the pieces and a hidden mechanical arm housed inside the playing board (see Figure B.8).

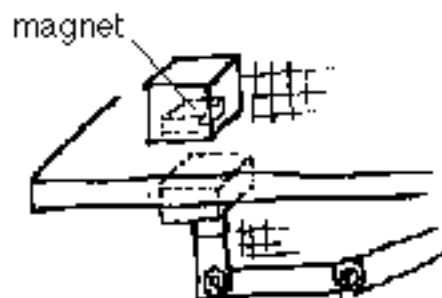


Figure B.8. A possible design for self-propelled bricks. A magnet is embedded inside a brick. A computer controlled mechanical arm operates underneath the surface. The brick is grabbed by magnetic attraction, moved with the mechanical arm and released by breaking the magnetic hold.

Appendix C: Experiment 1 — Coordination analysis

Below we present the coordination analysis used in experiment one to gain more insight into the manipulation styles for the four input conditions. Coordination is defined as the degree of inefficiency used in solving the task. This coordination metric, defined by Zhai [1995], was originally developed for measuring coordination in 6 degree of freedom input devices. It measures how much effort above optimal a task requires. An optimal solution is defined as all task dimensions being solved simultaneously and minimally. For example, an optimal path between two fixed points on a plane is a straight line. If subjects deviate from the straight line trajectory then, in general, it takes more effort to complete the task. Figure C.1 shows this situation for two dimensions of the task (rotation and translation). An optimal path (i.e., a straight line) is defined from the initial starting position (T_O, R_O) to the final target match (T_f, R_f). The subject's actual path often deviates from the optimal path. We can compute the amount of deviation by calculating the length of both the optimal path and the subject's actual path.

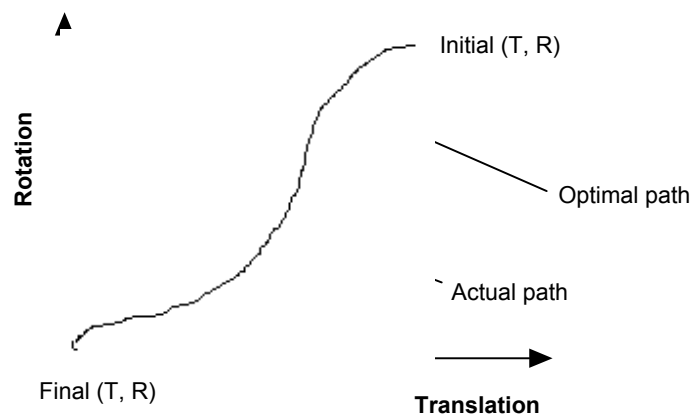


Figure C.1. Measuring task coordination within the translation and rotation dimensions.

The overall coordination inefficiency (CI) coefficient for our task, which uses a three dimensional space (translation, rotation and scale), is defined as:

$$CI = (\text{Length of actual path} / \text{Length of optimal path}) - 1.0$$

Thus a CI value of 0.0 indicates that the subject followed the optimal path. Values greater than 0.0 indicate the amount of inefficiency observed in the task. We now present our findings based on this coordination metric.

There was a significant difference for the total coordination values between the input conditions ($F(3,24) = 4.6, p < .01$). A pairwise means comparison indicates that the major effect was attributed to the stylus, time-multiplex condition which was less efficient than the space-multiplex conditions when we consider collectively the 1D, 2D and 3D tasks (stylus and bricks: $F(1,24) = 11.4, p < .005$; stylus and ruler: $F(1,24) = 6.9, p < .05$; and stylus and square: $F(1,24) = 8.7, p < .01$). There is no significant difference for coordination values among space-multiplex input conditions for the collective 1D, 2D and 3D tasks. However, the stylus, time-multiplex condition has greater coordination efficiency compared to the space-multiplex conditions for 1D tasks (see Figure C.2).

There was no significant coordination difference between the bricks and stretchable ruler and square. Interviewing the subjects after the experiment we noted that some subjects like the fact that the bricks were not "attached" to one another so that they could move one without affecting the other. However, other subjects considered this a deficiency in the design. Therefore, we cannot conclude either a coordination benefit or cost for having two independent brick devices compared to a single input device (e.g., stretchable ruler and square).

If we further decompose the coordination data by dimensionality we find a significant interaction effect on coordination for input condition and dimensionality ($F(6,48) = 14.9, p < .001$). Figure C.2 show the results more clearly by graphing the coordination values by input device configuration and task dimensionality. We can see that the stylus, time-multiplex condition has a lower degree of coordination inefficiency for the one dimensional tasks and gets progressively more inefficient for 2 and 3 dimensional tasks. A pairwise means comparison between the stylus and bricks conditions quantifies this significance (1D: $F(1,24) = 6.31, p < .02$; 2D: $F(1,24) = 10.4, p < .005$; and 3D: $F(1,24) = 140.8, p < .001$). The greater coordination efficiency for the 1D tasks and stylus condition is probably due to the fact that the stylus interactions allow for one dimension to be affected at any given time while

constraining the other dimensions. Thus, they can prevent alterations in other dimensions (e.g., one can move a rectangle without changing its rotation values).

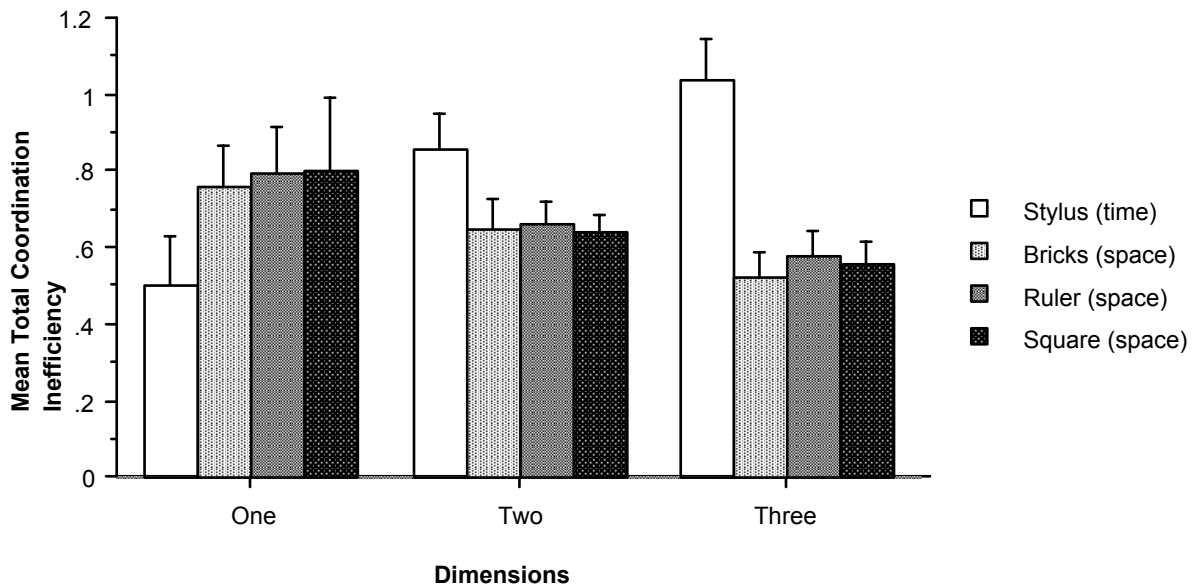


Figure C.2. Mean coordination inefficiency by input device configuration and task dimensionality.

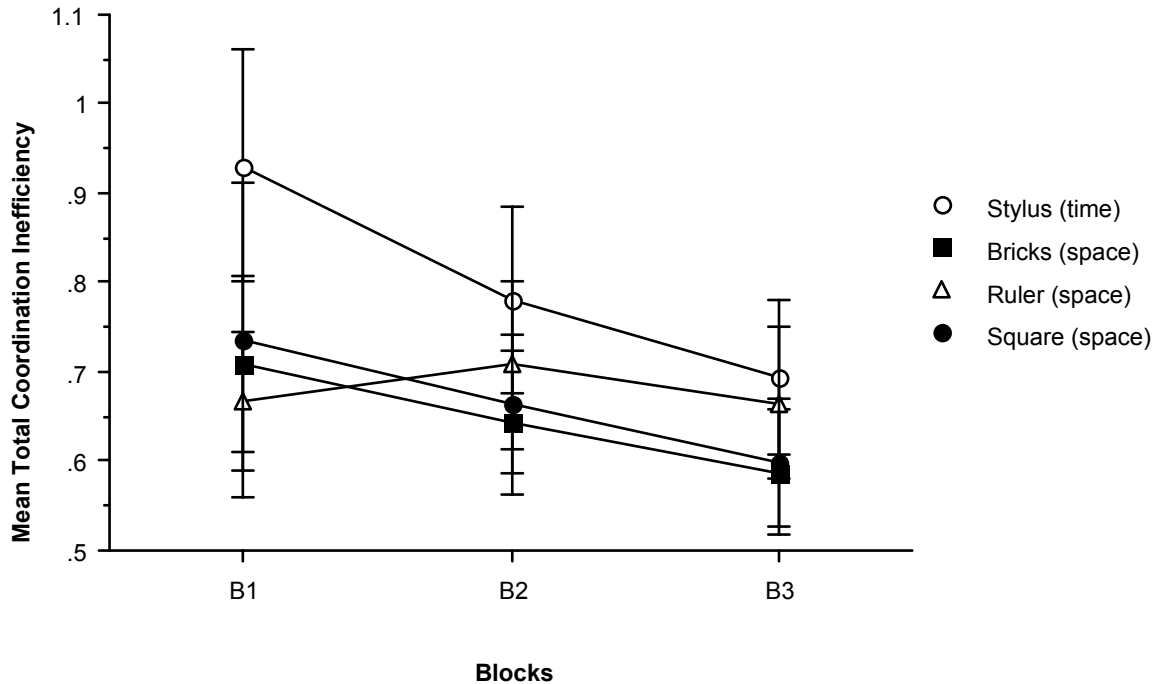


Figure C.3. Mean total coordination values across trial blocks and input device configurations.

In contrast, the degree of coordination efficiency for the space-multiplex conditions improve as the task dimensionality increases. This result may be explained by the fact that the devices have more degrees of freedom (DoF) than the task requires. Each of the space-multiplex input conditions use 2 sensors which sense an (x, y) value; thus, there are 4 DoF for the user to manipulate. For the 1D task, there are 3 extra DoF (for the 2D task there are 2 extra DoF and for the 3D task there is 1 extra DoF). These extra degrees of freedom have the potential to contribute to inefficiencies while solving the task. Said slightly differently, the extra device DoF allow for more "interaction noise" during a trial. Finally, note that within every task dimensionality for the space-multiplex, the bricks, ruler and square have statistically equivalent coordination values (see Figure C.2).

There was an overall learning effect across the blocks of trials for all input device conditions ($F(2,16) = 15.7, p < .001$). Moreover, if we decompose the learning by input device configuration, a weak interaction effect is found ($F(6,48) = 2.1, p < .07$). Figure C.3 shows the mean coordination values separated by blocks and input condition. Coordination improves as the subjects become more experienced in the task. The stretchable ruler is a curious deviator. We cannot easily explain why coordination decreased for the second block of trials.

Appendix D: Experiment 1 — Statistical Results

Source	df	Sum of Squares	Mean Square	F-Value	P-Value
Order	3	1745.090	581.697	1.075	.4127
Subject(Group)	8	4328.779	541.097		
Condition	3	29863.673	9954.558	97.990	.0001
Condition * Order	9	1715.377	190.597	1.876	.1055
Condition * Subject(Group)	24	2438.091	101.587		
Blocks	2	1295.635	647.818	24.832	.0001
Blocks * Order	6	234.190	39.032	1.496	.2418
Blocks * Subject(Group)	16	417.414	26.088		
Dimensions	2	6931.020	3465.510	88.297	.0001
Dimensions * Order	6	337.609	56.268	1.434	.2624
Dimensions * Subject(Group)	16	627.975	39.248		
Trials	5	622.560	124.512	13.254	.0001
Trials * Order	15	139.993	9.333	.993	.4800
Trials * Subject(Group)	40	375.779	9.394		
Condition * Blocks	6	1750.981	291.830	13.022	.0001
Condition * Blocks * Order	18	699.054	38.836	1.733	.0659
Condition * Blocks * Subject(Group)	48	1075.698	22.410		
Condition * Dimensions	6	11126.783	1854.464	57.022	.0001
Condition * Dimensions * Order	18	561.518	31.195	.959	.5183
Condition * Dimensions * Subject(Group)	48	1561.061	32.522		
Blocks * Dimensions	4	86.867	21.717	4.043	.0092
Blocks * Dimensions * Order	12	157.400	13.117	2.442	.0218
Blocks * Dimensions * Subject(Group)	32	171.869	5.371		
Condition * Trials	15	2024.608	134.974	12.471	.0001
Condition * Trials * Order	45	405.968	9.022	.834	.7537
Condition * Trials * Subject(Group)	120	1298.796	10.823		
Blocks * Trials	10	386.065	38.607	4.085	.0001
Blocks * Trials * Order	30	337.056	11.235	1.189	.2671
Blocks * Trials * Subject(Group)	80	756.042	9.451		
Dimensions * Trials	10	2425.625	242.563	19.180	.0001

Dependent: Time

Table D.1. Repeated Measure Variance Analysis of task completion time for Experiment 1.

Source	df	Sum of Squares	Mean Square	F-Value	P-Value
Dimensions * Trials * Order	30	333.361	11.112	.879	.6461
Dimensions * Trials * Subject(Group)	80	1011.705	12.646		
Condition * Blocks * Dimensions	12	300.644	25.054	2.885	.0019
Condition * Blocks * Dimensions * Order	36	368.449	10.235	1.179	.2607
Condition * Blocks * Dimensions * Sub(Grp).	96	833.639	8.684		
Condition * Blocks * Trials	30	911.083	30.369	3.219	.0001
Condition * Blocks * Trials * Order	90	596.477	6.628	.703	.9734
Condition * Blocks * Trials * Subject(Grp).	240	2264.007	9.433		
Condition * Dimensions * Trials	30	3539.382	117.979	8.515	.0001
Condition * Dimensions * Trials * Order	90	1373.424	15.260	1.101	.2804
Condition * Dimensions * Trials * Sub(Grp).	240	3325.490	13.856		
Blocks * Dimensions * Trials	20	159.100	7.955	.735	.7849
Blocks * Dimensions * Trials * Order	60	893.296	14.888	1.377	.0599
Blocks * Dimensions * Trials * Subject(Grp)	160	1730.543	10.816		
Condition * Blocks * Dimensions * Trials	60	523.504	8.725	.809	.8441
Cond. * Blocks * Dimen. * Trials * Order	180	2083.135	11.573	1.073	.2762
Cond. * Blocks * Dimen. * Trials * Sub(Grp)	480	5175.309	10.782		

Dependent: Time

Table D.2. Repeated Measure Variance Analysis of task completion time for Experiment 1 (continued).

Appendix E: Experiment 2 — Statistical Results

Source	df	Sum of Squares	Mean Square	F-Value	P-Value
Subject	11	469731.620	42702.875		
Condition	2	406914.812	203457.406	104.293	.0001
Condition * Subject	22	42918.015	1950.819		
Trials	5	48924.058	9784.812	4.853	.0010
Trials * Subject	55	110884.732	2016.086		
inputDevice	3	261461.184	87153.728	47.955	.0001
inputDevice * Subject	33	59974.794	1817.418		
Condition * Trials	10	17388.490	1738.849	.684	.7374
Condition * Trials * Subject	110	279686.164	2542.601		
Condition * inputDevice	6	12912.352	2152.059	3.421	.0053
Condition * inputDevice * Subject	66	41516.661	629.040		
Trials * inputDevice	15	12547.640	836.509	1.146	.3195
Trials * inputDevice * Subject	165	120410.944	729.763		
Condition * Trials * inputDevice	30	24615.683	820.523	.873	.6626
Condition * Trials * inDev. * Sub.	330	310322.996	940.373		

Dependent: RMS tracking error

Table E.1. Repeated Measure Variance Analysis of RMS tracking error for Experiment 2.

References

- Ayers, M. and Zeleznik, R. (1996). The Lego Interface Toolkit. *Proceedings of ACM Symposium on User Interface Software and Technology (UIST'96)*, To appear, New York: ACM.
- Azuma, R. (1993). Tracking requirements for augmented reality. *Communications of the ACM*, 36(7), 50-51.
- Baecker R. M. and Buxton. W.A.S. (Eds.), (1987). *Readings in Human-Computer Interaction, A multidisciplinary approach*, 357-365, Los Altos, CA: Morgan Kaufmann.
- Bier, E. A., Stone, M. C., Pier, K., Buxton, W. and DeRose, T. D. (1993). Toolglass and magic lenses: The see-through interface. *Proceedings of SIGGRAPH'93, Annual Conference Series (Anaheim, CA), Computer Graphics 27*, 73-80, New York: ACM.
- Bolt, R. A. and Herranz, E. (1992). Two-handed gesture in multi-modal natural dialog. *Proceedings of ACM Symposium on User Interface Software and Technology (UIST'92)*, 7-14, New York: ACM.
- Brewer, B. (1993). The integration of spatial vision and action. In N. Eilan, R. McCarthy and B. Brewer (Eds.). *Spatial Representation*, 294-316, Oxford, UK: Blackwell.
- Buxton, W. (In press). Living in Augmented Reality: Ubiquitous Media and Reactive Environments. In Finn, K., Sellen, A. and Wilber, S. (Eds.). *Video Mediated Communication*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Buxton, W. (1983). Lexical and Pragmatic Considerations of Input Structures. *Computer Graphics*, 17(1), 31-37.
- Buxton, W. (1986). Chunking and phrasing and the design of human-computer dialogues. *Proceedings of the IFIP World Computer Congress (Dublin, Ireland)*, 475-480. Reprinted in Baecker, R., Grudin, J., Buxton, W. and Greenberg, S. (Eds.). (1995). *Readings in Human-Computer Interaction: Towards the Year 2000*, 494-499, San Francisco, CA: Morgan Kaufmann, Inc.

- Buxton, W. and Myers, B. A. (1986). A Study in Two-Handed Input. *Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems*, 321-326, New York: ACM.
- Campbell, J. (1993). The role of physical objects in spatial thinking. In N. Eilan, R. McCarthy and B. Brewer (Eds.). *Spatial Representation*, 65-95, Oxford, UK: Blackwell.
- Card, S. K., Mackinlay, J. D. and Robertson, G. G. (1990). The design space of input devices. *Proceedings of CHI'90 Conference on Human Factors in Computing Systems*, 117-124, New York: ACM.
- Card, S. K., Moran, T. P. and Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum.
- Card, S. K., Robertson, G. G. and Mackinlay, J. D. (1991). The Information Visualizer: An information workspace. *Proceedings of CHI'91 Conference on Human Factors in Computing Systems*, 181-188, New York: ACM.
- Chamberlin, H. (1985). *Musical Applications of Microprocessors*. Hayden Books.
- Cooper, L. A. and Munger, M. P. (1993). Extrapolating and remembering positions along cognitive trajectories: Uses and limitations of analogies to physical motion. In N. Eilan, R. McCarthy and B. Brewer (Eds.). *Spatial Representation*, 112-131, Oxford, UK: Blackwell.
- Eilan, N., McCarthy, R. and Brewer, B. (1993). *Spatial Representation*. Oxford, UK: Blackwell.
- Elliott, J. M. and Connolly, K. J. (1984). A classification of manipulative hand movements. *Developmental Medicine & Child Neurology*, 26, 283-296.
- Feiner, S., MacIntyre, B. and Seligmann, D. (1993). Knowledge-based Augmented Reality. *Communications of the ACM*, 36(7), 52-62.
- Fitts, P. and Peterson, J. (1964). Information Capacity of Discrete Motor Responses. *Journal of Experimental Psychology*, 67, 103-112.
- Fitzmaurice, G. W. (1993). Situated information spaces and spatially aware palmtop computers. *Communications of the ACM*, 36(7), 38-49.
- Fitzmaurice, G. W. and Buxton, W. (1994). The Chameleon: Spatially Aware Palmtop Computers. *ACM SIGGRAPH Video Review*, issue 97, CHI'94 Video Program, New York: ACM.
- Fitzmaurice, G. W., Ishii, H. and Buxton, W. (1995). Bricks: Laying the Foundations for Graspable User Interfaces. *Proceedings of CHI'95 Conference on Human Factors in Computing Systems*, 442-449, New York: ACM.

- Fitzmaurice, G. W., Zhai, S. and Chignell, M. H. (1993). Virtual Reality for Palmtop Computers. *ACM Transactions on Information Systems*, 11(3), 197-218.
- Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F. (1990). *Computer Graphics: Principles and Practice*, Second edition in C. Reading, MA: Addison-Wesley.
- Fowler, R.L., Williams, W.E., Fowler, M.G. and Young, D.D. (1968). An investigation of the relationship between operator performance and operator panel layout for continuous tasks. Tech. Rept. 68-170. US. Air Force ARML (AD-692 126). Referred to In Sanders, M. S. and McCormick, E. J., *Human Factors in Engineering and Design*, 364-365, McGraw-Hill, Sixth edition.
- Fukuzaki, Y. (1993). Electronic pen according to the BTRON guideline and its background. *Tronware*, vol. 4, Personal Media publishers, Japan, 49-62.
- Gibson, J. J. (1950). *The Perception of the Visual World*. Boston, MA: Houghton Mifflin.
- Gibson, J. J. (1962). Observations on active touch. *Psychological Review*, 69, 477-491.
- Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin.
- Gleicher, M. (1993). A Graphics Toolkit Based on Differential Constraints. *Proceedings of the 1993 ACM Symposium on User Interface Software and Technology (UIST'93)*, 109-120, New York: ACM.
- Guiard, Y. (1987). Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model. *Journal of Motor Behavior*, 19(4), 486-517.
- Hinckley, K., Pausch, R., Goble, J. C. and Kassell, N. F. (1994). Passive Real-World Interface Props for Neurosurgical Visualization. *Proceedings of CHI'94 Conference on Human Factors in Computing Systems*, 452-458, New York: ACM.
- Houde, S. (1993). Iterative design of an interface for easy 3-D direct manipulation. *Proceedings of INTERCHI'93 Conference on Human Factors in Computing Systems*, 135-142, New York: ACM.
- Hutchins, E. L., Hollan, J. D. and Norman, D. A. (1986). Direct Manipulation Interfaces. In D. A. Norman and S. W. Draper (Eds.). *User Centered System Design, New Perspectives on Human-Computer Interaction*, 87-124, Hillsdale, NJ: Lawrence Erlbaum.
- Jacob, R. J. K., Leggett, J. J., Myers, B. A. and Pausch, R. (1993). Interaction styles and input/output devices. *Behaviour & Information Technology*, 12(2), 69-79.
- Jeannerod, M. (1984). The timing of natural prehension movements. *Journal of Motor Behavior*, 16(3), 235-254.

- Kabbash, P., Buxton, W. and Sellen, A. (1994). Two-Handed Input in a Compound Task. *Proceedings of CHI'94 Conference on Human Factors in Computing Systems*, 417-423, New York: ACM.
- Kandel, S., Boe, L.-J. and Orliaguet, J.-P. (1993). Visual detection of coarticulatory anticipation or...guessing what has not yet been written. *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS'93)*, 148-154.
- Kirsh, D. (1995a). The intelligent use of space. *Journal of Artificial Intelligence*, 73(1-2), 31-68.
- Kirsh, D. (1995b). Complementary strategies: why we use our hands when we think. *Proceedings of 7th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum.
- Kirsh, D. and Maglio, P. (1994). On Distinguishing Epistemic from Pragmatic Action, *Cognitive Science*, 18, 513-549.
- Knep, B., Hayes, C., Sayre, R. and Williams, T. (1995). Dinosaur Input Device, *Proceedings of CHI'95 Conference on Human Factors in Computing Systems*, 304-309, 589, New York: ACM.
- Kurtenbach, G. P. (1993). *The design and evaluation of marking menus*, Ph.D. Thesis, Dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada.
- Lederman, S. J. and Klatzky, R. L. (1994). The Intelligent Hand: An Experimental Approach to Human Object Recognition and Implications for Robotics and AI. *AI magazine*, 15(1), 26-38, Spring 1994.
- Mackay, W. and Pagani, D. S. (1994). Video Mosaic: Layout out time in a physical space. *Proceedings of Second ACM International Conference on Multimedia*, 165-172, New York: ACM.
- Mackay, W., Velay, G., Carter, K., Ma, C. and Pagani, D. (1993). Augmenting Reality: Adding computational dimensions to paper. *Communications of the ACM*, 36(7), 96-97.
- MacKenzie, C. L. and Iberall, T. (1994). *The Grasping Hand*. Amsterdam: North-Holland, Elsevier Science.
- Marcus, A. (1990). Designing Graphical User Interfaces. *UNIX World*, 107-111. August 1990.
- Marteniuk, R. G., MacKenzie, C. L., Jeannerod, M., Athenes, S., & Dugas, C. (1987). Constraints on human arm movement trajectories. *Canadian Journal of Psychology*, 41, 365-378.

- MIDI 1.0 Detailed Specification and Standard MIDI Files 1.0, International MIDI Association, Los Angeles, CA.
- Newman, W. and Wellner, P. (1992). A desk supporting computer-based interaction with paper documents. *Proceedings of CHI'92 Conference on Human Factors in Computing Systems*, 587-592, New York: ACM.
- Norman, D. A. (1993). *Things that make us smart: defending human attributes in the age of the machine*. Reading, Massachusetts: Addison-Wesley.
- Owen, R., Kurtenbach, G., Fitzmaurice, G., Baudel, T. and Buxton, W. (1995). Bimanual Manipulation in a Curve Editing Task. In preparation.
- Rekimoto, J. (1995). Augmented Interaction: Interacting with the real world through a computer, In Y. Anzai, L. Ogawa and H. Mori (Eds.). *Symbiosis of Human and Artifact*, 255-260, Amsterdam: North-Holland, Elsevier Science.
- Rekimoto, J. and Nagao, K. (1995) The World through the Computer: Computer Augmented Interaction with Real World Environments, *Proceedings of the 1995 ACM Symposium on User Interface Software and Technology (UIST'95)*, 29-36, New York: ACM.
- Resnick, M. (1993). Behavior Construction Kits. *Communications of the ACM*, 36(7), 64-71.
- Robertson, G., Card, S. K. and Mackinlay, J. D. (1993). Information Visualization Using 3D Interactive Animation. *Communications of the ACM*, 36(4), 56-71.
- Robertson, G. G., Mackinlay, J. D. and Card, S. K. (1991). Cone trees: Animated 3D visualizations of hierarchical information. *Proceedings of CHI'91 Conference on Human Factors in Computing Systems*, 189-202, New York: ACM.
- Robinett, W. (1992). Synthetic Experience: A Proposed Taxonomy. *Presence*, 1(2), 229-247.
- Sanders, M. S. and McCormick, E. J. (1987). *Human Factors in Engineering and Design*, "Chapter 13: Physical space and arrangement," 363-386, McGraw-Hill, Sixth edition.
- Sachs, E., Roberts, A. and Stoops, D. (1990). 3-Draw: A tool for the conceptual design of three-dimensional shapes. CHI'90 Technical Video Program, *ACM SIGGRAPH Video Review*, Issue 55, No. 2., New York: ACM.
- Schmidt, R. A. (1988). *Motor Control and Learning*. Champaign, IL: Human Kinetics Publishers.
- Schneider, S.A. (1990). *Experiments in the dynamic and strategic control of cooperating manipulators*. Ph.D. Thesis, Dept. of Elec. Eng., Stanford Univ., Stanford, CA.

- Siiio, I. (1995). InfoBinder: A pointing device for a virtual desktop system. *Proceedings of HCI International '95*, Yokohama, Japan. In *Symbiosis of Human and Artifact 20B*. (1995), 261-264, Amsterdam: North-Holland, Elsevier Science.
- Singer, R. N. (1980). *Motor Learning and Human Performance, An application to motor skills and movement behaviors*, 3rd ed., New York: MacMillan Publishing Co.
- Suzuki, H., Kato, H. (1993). AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning. *Proceedings of 4th European Logo Conference*, Aug. 1993, Athens Greece, 297-303.
- Tani, M., Yamaashi, K., Tanikoshi, K., Futakawa, M. and Tanikoshi, K. (1992). Object-oriented video: Interaction with real-world objects through live video. *Proceedings of CHI'92 Conference on Human Factors in Computing Systems*, 593-598, New York: ACM.
- Vicente, K. J. and Rasmussen, J. (1992). Ecological Interface Design: Theoretical Foundations. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(4), 589-606.
- Want, R., Hopper, A., Falcao, V. and Gibbons, J. (1992). The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1), 91-102.
- Warren, D. H. and Rossano, M. J. (1991). Intermodality Relations: Vision and Touch. In M. A. Heller and W. Schiff (Eds.). *The Psychology of Touch*, 119-137, Hillsdale, NJ: Lawrence Erlbaum.
- Weiser, M. (1991). The computer for the 21st Century. *Scientific American*, 265(3), 94-104.
- Weiser, M. (1993). Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7), 75-84.
- Welford, A.T. (1968). *Fundamentals of Skill*. London: Methuen.
- Wellner, P. (1993). Interacting with paper on the DigitalDesk. *Communications of the ACM*, 36(7), 86-96.
- Wellner, P., Mackay, W. and Gold, R. (1993). Special Issue on Computer Augmented Environments: Back to the Real World. *Communications of the ACM*, 36(7), July 1993.
- Zhai, S. (1995). *Human Performance in Six Degree Of Freedom Input Control*, Ph.D. Thesis, Dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada.
- Zhai, S., Milgram, P. and Buxton, W. (1995). The Effects of Using Fine Muscle Groups in Multiple Degree-of-Freedom Input. *Proceedings of CHI'96 Conference on Human Factors in Computing Systems*, 308-315, New York: ACM.

Zhang, J. and Norman, D. A. (1994). Representations in Distributed Cognitive Tasks. *Cognitive Science*, 18(1), 87-122.

Zimmerman, T., Smith, J.R., Paradiso, J.A., Allport, D. and Gershenfeld, N. (1995). Applying Electric Field Sensing to Human-Computer Interfaces. *Proceedings of CHI'95 Conference on Human Factors in Computing Systems*, 280-298, New York: ACM.