Mechanical Constraints as Common Ground between People and Computers

James McMichael Patten

Bachelor of Arts, University of Virginia, June 1999
Master of Science, Massachusetts Institute of Technology, June 2001

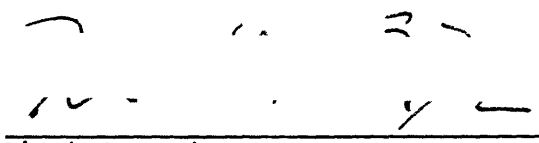
Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy at the Massachusetts Institute of Technology January 2006

Author
James McMichael Patten
Program in Media Arts and Sciences

Certified by
Hiroshi Ishii
Associate Professor of Media Arts and Sciences
Thesis Supervisor

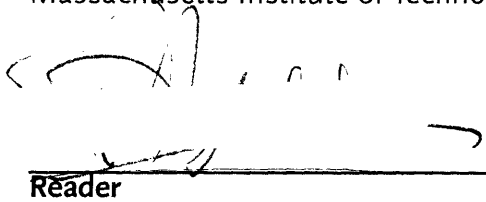
Accepted by
Andrew B. Lippman
Chair, Departmental Committee for Graduate Students
Program in Media Arts and Sciences

Thesis Committee



Thesis Supervisor
Hiroshi Ishii
Associate Professor of Media Arts and Sciences
Massachusetts Institute of Technology

Reader
John Maeda
Associate Professor of Media Arts and Sciences
Allen Professor of Media Arts and Sciences
Massachusetts Institute of Technology



Reader
Robert J. K. Jacob
Associate Professor of Computer Science
Tufts University

Mechanical Constraints as Common Ground between People and Computers

Abstract

This thesis presents a new type of human-computer interface based on mechanical constraints that combines some of the tactile feedback and affordances of mechanical systems with the abstract computational power of modern computers. The interface is based on a tabletop interaction surface that can sense and move small objects on top of it. Computation is merged with dynamic physical processes on the tabletop that are exposed to and modified by the user in order to accomplish his or her task. The system places mechanical constraints and mathematical constraints on the same level, allowing users to guide simulations and optimization processes by constraining the motion of physical objects on the interaction surface. The interface provides ample opportunities for improvisation by allowing the user to employ a rich variety of everyday physical objects as interface elements. Subjects in an evaluation were more effective at solving a complex spatial layout problem using this system than with either of two alternative interfaces that did not feature actuation.

Thesis Supervisor: Hiroshi Ishii
Associate Professor of Media Arts and Sciences
MIT Media Laboratory

Acknowledgements

I'd like to thank Mom, Dad and the whole family, especially my wife Mariliana Arvelo. I'd also like to thank Ana-Maria Cardenas for helping me in dire emergencies on more than one occasion.

Thanks to Chris Csikszentmihalyi, Hiroshi Ishii, Rob Jacob, John Maeda and Joe Paradiso for being sources of inspiration, knowledge, guidance and encouragement during my studies.

Thanks to Roberto Aimi, Ali Rahimi and Ben Recht for being great friends and collaborators.

Thanks to Dennis Cosgrove, Gordon Earlbacher, Worthy Martin, Randy Pausch, John Regher, John Stankovic, and John Viega for helping me take my first steps with computers and research.

Thanks to Mahoro Anabuki, Mike Anderson, Burak Arikan, Mike Ananny, Ari Benbasat, Leonardo Bonanni, Michael Broxton, Angela Chang, Mark Feldmeier, Brent Fitzgerald, Limor Freid, Amber Frid-Jimenez, Jim Gouldstone, Vincent Leclerc, Jeff Lieberman, Lisa Lieberson, Josh Lifton, Dan Maynes-Aminzade, Cameron Marlow, David Merrill, Dan Paluska, Joe Panganiban, Pangaro Gian, Parkes Amanda, Jonah Peretti, Hayes Raffle, Casey Reas, Brygg Ullmer, Paul Yarin, and all TMG, REG, ACG and PLW students for making the Media Lab a great place to be, being great friends and having tons of helpful suggestions and advice.

Thanks to Jason Alonso, Megumi Ando, Nelson Elhage, Jaryn Finch, Mario Marrufo, Orit Shamir, David Sheets, Jose Soltren, Cheryl Texin, Scott Torborg, Jay Watt, Richard Whitney, Henry Wu & friends and Clarissa Zimmerman for helping Pico become a reality.

Thanks to Joe Branc of Steelcase, Phil Fleming of Motorola and Mary Murphy-Hoye of Intel for their support of and feedback on my work.

Thanks to Linda Peterson and Pat Solakoff for helping me navigate MIT's maze of paperwork and regulations.

Table of Contents

1 Introduction

Long before modern electronic computers were developed, people relied on mechanical devices to perform computation. Many of these mechanical computers, such as the calculator in figure 1, share certain qualities that make them appealing from an interaction design perspective. The way these devices perform their task, and how they can be used is often apparent from a brief inspection. One reason is that the computational behavior of these devices is embodied in a mechanical system that is governed by the same basic laws of physics that govern our interaction with the world. In contrast, computation inside a microprocessor, while still governed by laws of physics, is much more difficult to understand as it is so far removed from our everyday experience.

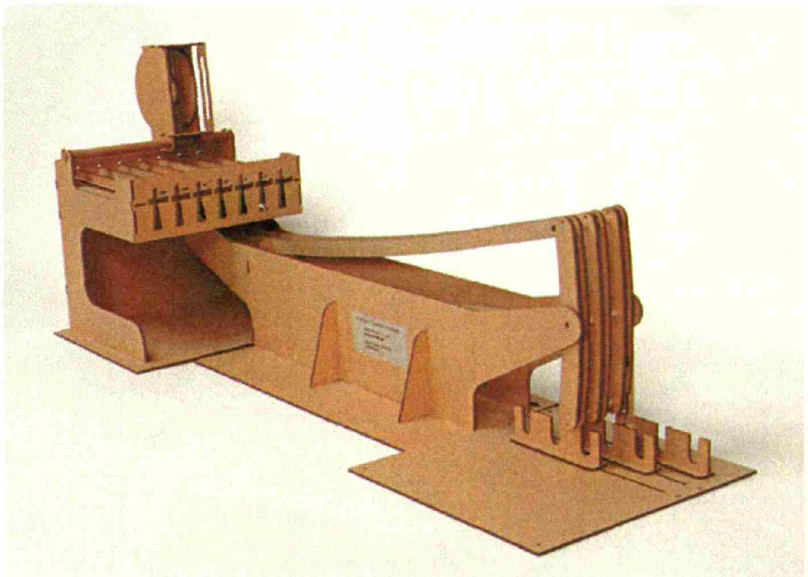


Figure 1: Modern recreation of Thomas Fowler's Ternary Calculating Machine photo: Mark Glusker http://www.mortati.com/glusker

Mechanical voting machines are another example of how a mechanical process that can be inspected can make a system easier to understand. In the 2004 United States presidential election, one of the criticisms of new touchscreen-based voting machines was that the machines' software was difficult to validate against tampering with the vote tally. In contrast, many mechanical voting machines could be inspected just by opening the back of the machine and watching it increment the counters as the vote levers were pulled. A mechanical system such as a voting machine provides a rich combination of visual, tactile and aural feedback about its function that makes it easier to understand than if only one type of sensory feedback were provided. Wireless internet connections are another example. While they can be convenient, they are often difficult to debug because the process of communicating with the wireless hub is invisible to the user. In contrast, wired connections expose the physical mechanism of communication to the user, where it can be more easily understood and debugged.

In this thesis I argue for a new type of interface that combines some of the advantages of mechanical systems with the abstract computational power of modern computers. Software-based computation is merged with dynamic physical processes that are exposed to and modified by the user in order to accomplish his or her task. By designing interfaces that employ the dynamic behavior of objects in the physical world, such as motion in response to physical forces, as an interface vocabulary we can create interfaces that allow people and computers to collaborate in novel ways. To explore this idea I have created an interface for solving two-dimensional spatial layout problems on an actuated tabletop sensing surface. Objects on this surface are moved under software control using electromagnets, but also by users standing around the table. The combination of these interactions, all governed by the friction and mass of the objects themselves directly affects the result of the task being performed. In

this thesis I will show how this technique can be applied to spatial layout problems, and discuss how it could be generalized to other types of tabletop interactions.

One example of a computationally complex problem that involves spatial layout is cellular network design. Cellular network design involves determining the location and configuration of cellular telephone towers in city in order to provide the best coverage possible. Currently these problems are solved by non-interactive software processes that take a long time to run and often give flawed solutions. I propose an alternative approach where the during the optimization process, the computer moves objects on the table around to reflect its current best guess about where the towers should be placed. Meanwhile the users physically intervene by grabbing towers and moving them away from problem areas and toward ones

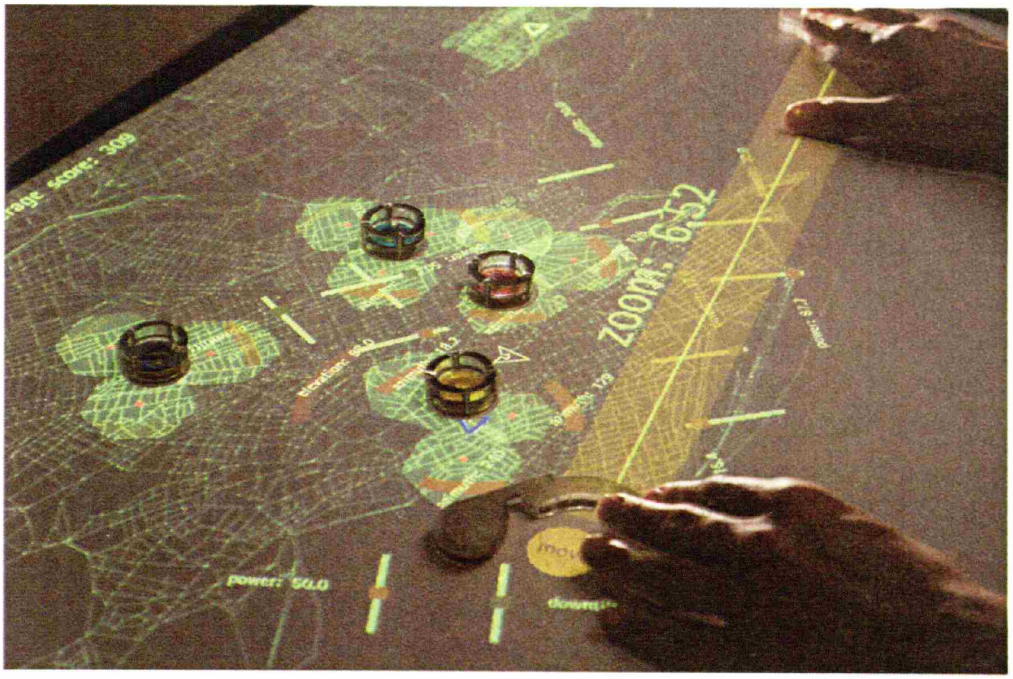


Figure 2: The cellular network design application running on the Pico platform.

that are more desirable. Users can employ different types of physical constraints, such as holding an object in one place with ones' hand, or placing physical barriers around certain areas to keep objects in or out. Due to the software association between the physical objects on the table and the towers being optimized, these physical constraints on the tabletop are implicitly translated into mathematical constraints in the optimization process. In this way, users can freely change the constraints of a running optimization process using their mechanical intuition for how the objects on the tabletop will respond to what they do.

Thesis Contributions

This thesis contains these contributions:

– A new approach to interacting with computers based on mechanical constraints, where the motion of physical objects provides a dynamic representation of a portion of the software's internal state, and gives the user a familiar vocabulary for interaction where his or her mechanical intuition is valid and useful.

– A series of interaction techniques for tabletop interfaces for performing tasks common to a variety of tabletop applications.

– An application that illustrates the above concepts in use to address a spatial optimization problem, sufficiently developed that it may be compared with other interface approaches to the same problem.

– Interface hardware, software, and overall system architecture that supports building new applications based on these principles.

– An evaluation of performance differences between tabletop interfaces based on tracked physical objects and those based on touch screens.

Spatial Optimization Problems

Spatial optimization problems are a class of engineering, design and operations research problems in which one must determine the best placement for a large number of items according to some sort of performance metric. Examples include factory floor planning, printed circuit board layout, and cellphone tower placement. Optimization algorithms run on computers play an important role in solving these problems. However, for a variety of reasons, computers are unable to solve most of these problems independent of human input. This limitation often leads to an iterative process in which a person or group of people sets the starting conditions for an optimization process, and then waits as the computer uses this input to calculate a solution. Upon seeing that the computer's solution is flawed in some unforeseen way, the user may reconfigure the starting conditions and ground rules of the optimization and try again. A cycle in this iterative process may take anywhere from minutes to days, depending on the type and size of the problem being solved.

The interface presented in this thesis replaces this model with one of real-time interaction between people and computers, in which physical objects allow the user to modify some internal constraints of the optimization as it is running and quickly explore alternative solutions to the problem at hand. Current systems for solving spatial layout problems often treat the objects to be laid out as a large parameter space, and then use a search process to try to find the best combination of parameters. The system presented in this thesis, called Pico (Physical Intervention in Computational Optimization), performs a similar search, and displays its current best guess on the tabletop as the search unfolds. At any point a user may influence this best current solution by moving the pucks and constraints on the table. The computer's solution search process in turn affects the positions of the objects. The dynamics of the physical objects on the table serve as a mechanism for reconciling the user's movements of objects with the computer's search process.

Mechanical Constraints and Computational Constraints

The user can easily modify this mechanical process as desired by placing mechanical constraints, or jigs, on the tabletop. In the cellphone tower placement application, one might want to avoid placing a tower in a park in the center of a city, even though the tower might provide good coverage from that position. If the software were to move a tower into the park, one could establish a mechanical constraint to keep the tower out. One could bend a flexible barrier, such as the one shown in figure 3, to match the shape of the border of the park, and then place it around the park on the map. As the computer tried to move a tower into the park, the puck representing the tower would be blocked by the barrier around the park. Through the iterative process of reconciling the puck positions with the tower positions inside the optimization engine, the tower would be prevented from entering the park. The mechanical constraint on the table affects the optimization results in the same way as would a computational constraint inside the optimization software.

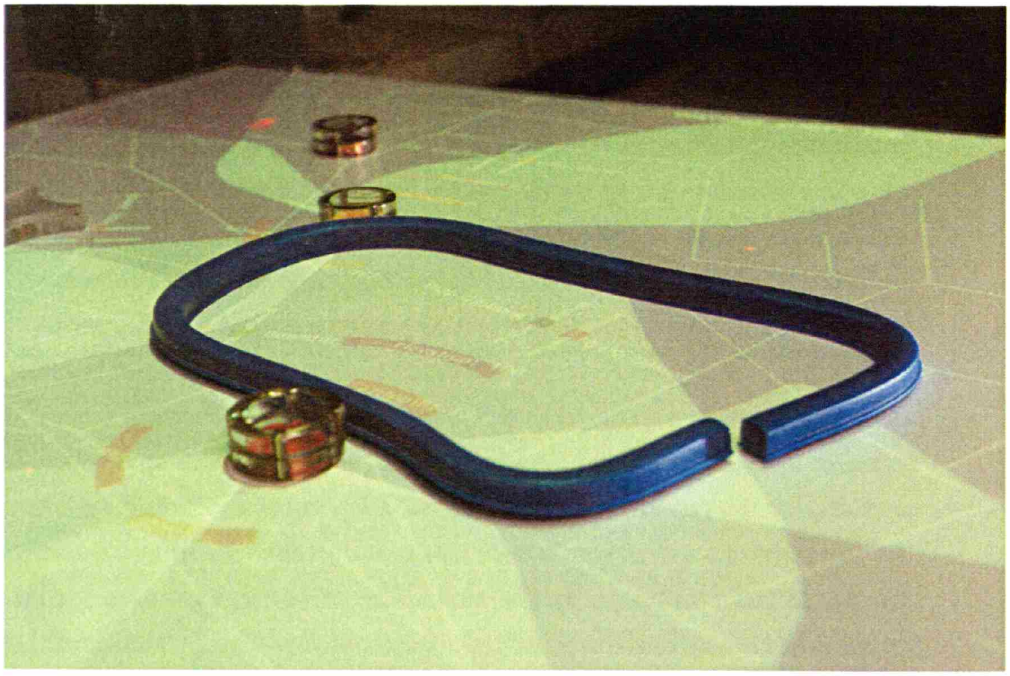


Figure 3: A physical barrier constraining the motion of a cellphone tower in the Pico application

If one wanted to ensure a certain minimum spacing between towers, one could place circular collars around the corresponding pucks on the table, thus physically enforcing the constraint, as shown in figure 4. Because the process of optimizing the tower layout is governed in part by this mechanical process, a wide variety of possible interactions is available to the user based changing the objects' physical interacting with the objects on the table.

Figure 4: A collar that enforces a minimum distance constraint between objects.

Interface Affordances

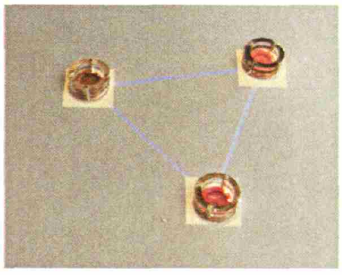
Donald Norman argues that people look for the perceived *affordances* of an interface to determine how to use it [Norman]. An example of an affordance is a door handle that through its form communicates whether the door should be pulled or pushed to open. In the case of tangible interfaces, where interface affordances may have metaphorical connections to familiar objects, the designer sometimes tries to communicate a set of available functionality to the user through association to a familiar object chosen based on a set of features the

interface is designed to support. However, this approach can be problematic when the user misinterprets the set of functionality that is available. For example, in Ishii's musicBottles project [Ishii 2001], users can access sound recordings by removing corks from bottles. The metaphor is that the sound is "contained" inside the bottle. However, this metaphor suggests a variety of other interactions that do not work in the interface, such as pouring sound from one bottle to another, holding an open bottle up to one's ear and hearing the sound, etc. When a user tries one of these interactions and finds that it does not work, he or she must adopt a more complicated mental model of the interface in order to understand how to use it, in essence reverse engineering the system through a process of experimentation with various possible functionality. While the affordances of the object suggest a certain set of functionality, the interface may contain only a subset that the designer explicitly designed in. This discrepancy comes from mediation between the interface and the actual process performing the computation. Where this mediation is absent the discrepancy can be avoided. One example is the record turntable. While it was originally conceived strictly as a way to play back recorded sound, the mechanism through which that sound is created is exposed to the user and as a result, a variety of new ways to make music with this device by interacting with it physically have emerged, such as "scratching" and other "turntablist" techniques [Katz 2004]. By exposing part of the device's actual computational function as an interface, one can extend the possibilities for interaction beyond those explicitly designed into the system to other possible interactions inferred by users. The work presented in this thesis applies this principle to tabletop interactive surfaces, so that users can apply their lifetime of knowledge about how physical objects interact dynamically to types of problems that are difficult to solve without a computer.

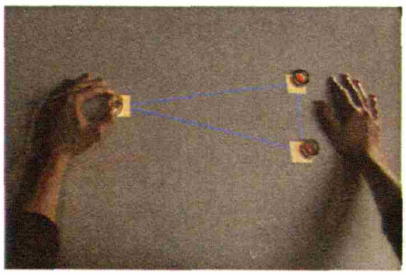
In the next chapter I present an extended example of how the Pico system is used in the context of cellular telephone tower placement. Chapter 3 covers related work in computer interfaces and supporting theory. Chapter 4 describes the hardware and software implementation of the work. Chapter 5 presents a series of interaction techniques, some of which are suitable for a variety of tabletop sensing platforms, and others that specifically apply to interfaces including actuation. Chapter 6 describes two experiments that evaluate different aspects of this work. The final chapter presents some conclusions and future directions for this research.

2 Extended Example

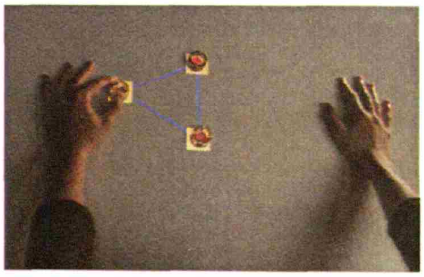
Pico works by iteratively attempting to resolve a series of software-defined constraints among a set of pucks on the interaction surface. A simple example of this process is shown below. A set of rules in software says that the distance between each of these three objects should be equal. The system iteratively measures the distances between the objects, and gradually moves them to satisfy the constraints, forming a triangle. This simple set of constraints can be satisfied by an infinite number of different positions of pucks on the tabletop. However, the user can add additional mechanical constraints to the tabletop to further constrain the problem. For example, the user might hold one of the pucks in place with his or her hand, or he or she might place an obstruction between one of the pucks and the others. As the system iteratively applies the set of rules in its software, the position of the pucks adjusts to conform to both the mechanical constraints applied by the user on the tabletop and the software constraints previously programmed into the application.



1 Software rules state that the three pucks should be an equal distance from each other.



2 A user grabs one of the pucks, moves it to the left, and holds it there. The system senses this movement and tries to pull the lone puck toward the other two. At the same time, it pulls the two pucks on the right toward the one on the left.



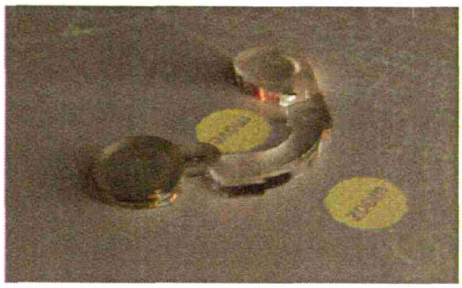
3 As the user constrains the position of the leftmost puck, the computer's attempt to move it has little effect. The two pucks on the right move to the left until the rules defined in software are again satisfied.

Pico allows users to collaborate with computers to solve complex problems using physical constraints as mathematical constraints. These physical constraints are easily defined and changed by users, and the cause and effect relationships between them are readily predictable and observable in a way that can be difficult with constraints implemented in software.

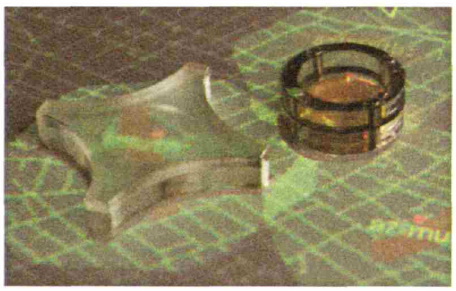
While creating equilateral triangles on a tabletop is a simple problem, this approach can also be applied to more complex spatial tasks. One such task is determining the placement and configuration of cellphone towers in a network to provide the best telephone coverage. This problem is extremely complex, and teams of engineers armed with many computers often work for weeks to find good solutions. Computers are not able to solve these types of problems on their own because of the variety of subtle issues that must be considered. For example, if a certain politician is instrumental in getting a large cellphone infrastructure project approved, one must assure that this politician's house has good cellphone coverage. There are often a variety of zoning laws and other regulations, some of which may be negotiable while others are not.

Because of these complex issues, there is often not a clear optimal solution to this type of spatial layout problem. Rather, there are sets of competing tradeoffs and interests that must be considered and balanced. Pico aims to allow the various interested parties to collaborate in such problem solving tasks by making it easy to change underlying constraints while the system is running, and make it easier to see and understand the cause and effect relationships present in these changes.

As the application starts, a map of the area of interest is projected on the interaction surface. The user has at his or her disposal three types of objects: the standard Pico puck, a star shaped selector puck,



The navigation puck



The standard Pico puck, and star-shaped selector puck

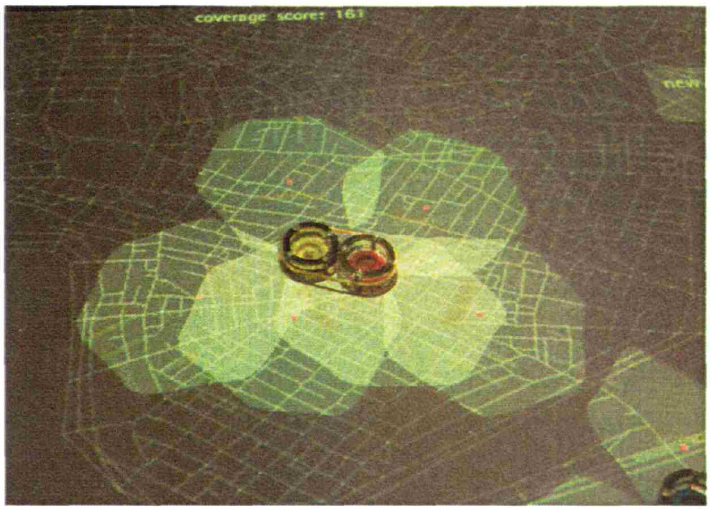
and a boomerang-shaped navigation puck. The standard pucks include magnets so that the electromagnet array below the sensing surface can pull them as necessary. These pucks can be mapped to individual cellphone towers to move them. The selector puck can change a variety of parameters of each tower, such as the elevation, and angle and power output of each antenna element. The selector puck fits together with the navigation puck to activate pan and zoom functions, discussed in chapter 5.

The user adds new radio towers to the map by placing a puck on the "new tower" icon. A tower appears and moves on the map along with the position of the puck. This association is a variation of the "binding" concept used with pucks on the Sensetable [Patten 2001] system. While in Sensetable the position of a puck completely determines the position of the underlying digital information, with Pico the position of the underlying digital object and the software-based "forces" upon it influence the position of the corresponding puck, while the position and physical forces upon the puck in turn influence the position of the underlying digital object. The constraint engine tries to keep these two positions (physical and digital) as consistent as possible while attempting to satisfy other constraints.

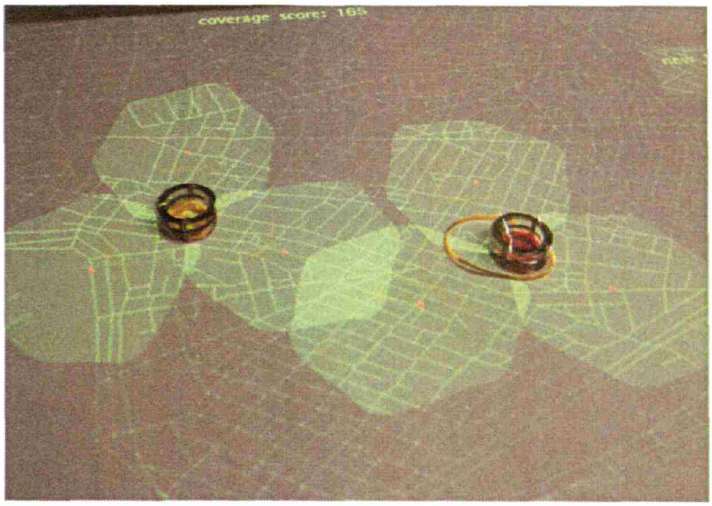
As soon as the new tower is added to the map, the computer begins searching for the best place to put that tower according to a fitness function based on a variety of factors, including the coverage area obtained by placing a tower in a given location (coverage score), and the cost associated with that placement (cost score). The computer carries out its search using simulated annealing [Metropolis 1953]. It compares the coverage and cost scores of nearby locations to the current one, and tries to move the tower away from areas that score poorly and closer to areas that score well. If the user moves the tower around the map with his or her hand, he or she will feel these forces as the computer identifies nearby desirable and undesirable areas for tower placement. If the user releases the puck, it will slowly move around the map on its own as the computer continues its annealing process, searching for the best location. Once the computer identifies a local minimum, the puck will tend to stay in that area.

If the user places another tower on the map directly adjacent to this local minimum, several redundant areas of coverage may be created, as shown on the next page. These areas will likely disrupt the previous local minimum, and the computer will begin searching for a new local minimum. As it does so, the towers will spread apart, reducing the redundant coverage area. If the users tries to squeeze the two pucks back together, he or she will feel the computer's attempt to improve the overall coverage by separating the towers as a physical force pulling the two pucks away from each other.

The user can temporarily override the computer's attempt to separate these towers by simply holding them together, or connecting them with a rubber band or a ring. In this case the system continues to optimize the layout of the towers within the constraint established by the user. The user might want to establish a constraint like this one if, for example, he or she wanted to explore what the implications might be if a certain geographic area were to need more network capacity than originally anticipated.



Two adjacent cellphone towers in the Pico application. The computer is trying to separate these towers to improve the overall coverage, but is unable to because the towers are physically attached by a rubber band.



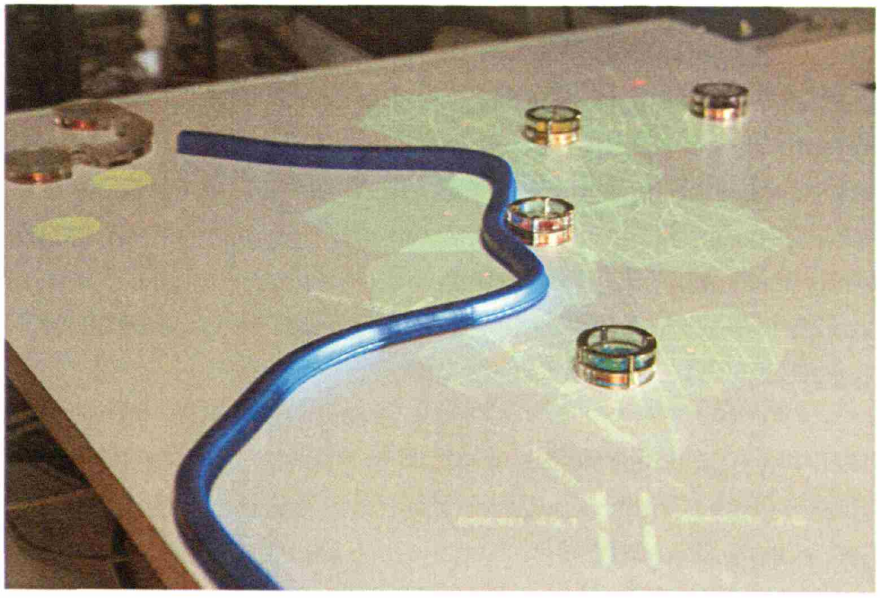
When the rubber band is removed, the towers move apart in response to the removed constraint as the computer continues searching for a better layout.

The user can continue to add towers and exercise as much or as little control as he or she desires in the placement of any particular tower. The assumption behind this collaboration between users and the computer is that the users have high level ideas, concerns, requirements and intuition about what would constitute a good solution to the problem at hand. The computer on the other hand, has none of these things, but is very good at comparing thousands of similar candidate solutions and determining which is best according to a set of criteria defined in a fitness function. By merging constraints defined in application software with physical constraints that can be constantly edited and adjusted by users, Pico aims to combine the unique strengths of both the users and the computer to solve complex spatial problems.

Because Pico's optimization works in an incremental fashion, it can sometimes get "stuck" in a local minimum during the optimization process. When this happens, a user simply has to push a puck out of its equilibrium, and the search process continues. In many systems that use simulated annealing, the "temperature" of the system is a global parameter that affects all variables equally. However, the spatial nature of Pico and the physical vocabulary of the interface allow the user to increase the temperature of some variables while leaving others unchanged, or even forcing them to stay the same by holding them in place.

As the optimization process continues, Pico may move a tower into a location that is obviously incorrect from the user's perspective. For example, early versions of the Pico cellphone tower optimization engine thought that it was very inexpensive to build towers in the middle of rivers, because the map data listed the price of real estate in these areas as zero. As a result, towers tended to move toward bodies of water near densely populated urban areas.

However, one could compensate for this error by placing a flexible boundary around the border of the river to prevent the towers from moving into it, as shown below. While it is unlikely that such an error would be incorporated into a production system for cellphone tower optimization, normally the users of such a system will have a more nuanced view of the problem at hand than will the computer. Such users can add and adjust boundary constraints to control the motions of towers. For example, if a network planning team were unsure whether they would receive the necessary zoning approvals to place a tower in a park in the middle of a city, they could compare the results for both conditions by placing a boundary around the park and comparing the best solution in that condition to one where the boundary was removed.



A barrier preventing towers from moving out of a specific geographic area on the map.

Summary

In this example we can see some of the key advantages that Pico provides over previous systems:

Collaborative Interaction Between User and Computer

In past tabletop interfaces such as the Sensetable, properties of objects in the interface were either completely determined by the user's movement of pucks, or instead controlled completely by software. This dichotomy was related to the notion of "binding", in which a physical object is associated with digital content projected on the table, such that the graphical projection moves wherever the puck moves. Pico replaces this notion of binding with a more flexible one, where the system attempts to keep the physical and digital representations in the same place by moving them gradually toward each other when their positions are in conflict. As a result the user feels mathematical inconsistencies in the form of physical forces pulling the puck in a direction that will resolve inconsistency. This method of constraint resolution opens up a spectrum of possibilities between the behavior defined in software and the user's goals. The more persistently the user pulls an object away from the position it would otherwise take, the more the user influences the object's final position. This spectrum of possibilities between the user's instructions and the software's autonomous activity allows the interface to incorporate the computer's and user's inputs to achieve a final result, rather than having to choose one and reject the other.

Rich Physical Vocabulary

Because some of the computation in Pico is a result of the mechanics of physical objects interacting on the sensing surface, the user can apply his or her knowledge and intuition about the way objects interact in the physical world to change the interaction of objects on the tabletop. For example, increasing the mass or friction of an

object will make it harder to move (making the associated parameter harder to change) and putting a slippery surface underneath an object will make it move more easily. The goal is to support any physical interaction that affects the position of the pucks on the 2D surface, not just the ones that were explicitly planned for during the design of the system, so as to leverage the user's physical intuition as much as possible.

Combination of Mechanical and Mathematical Constraints

An important part of the physical vocabulary referred to above is the idea that physical constraints have the same effect as mathematical constraints during the problem solving process. These constraints provide several advantages over screen based constraints, specifically that they are legible, flexible and ad hoc. By legible, I mean that users and bystanders can look at a constraint and understand the cause and effect relationships between that object and the motion of other objects on the table without having to learn a new set of computer commands. By flexible I mean that the constraints can be easily changed without having to pause the program or use many mouse movements or keystrokes. Rather, one can grab a constraint and manipulate its physical form, with all of the advantages that entails over manipulating virtual objects in terms of tactile feedback. By ad hoc, I mean that user's can easily make exceptions to physical constraints. They serve as guidelines rather than strict rules. For example, if one defines a boundary on the table with a physical barrier to keep objects out of a certain part of the table, one may later see an object push against that barrier, and realize that the barrier should in fact not apply to that object, while still applying to others. One simply has to pick up the object and place it on the other side of the barrier. A separate set of software logic and commands is not necessary to handle this special case.

3 Supporting Work

Tabletop Interfaces

A common mechanism employed in modern graphical user interfaces (GUIs) is the "desktop metaphor" [Johnson 1989]. In this approach, a computer system presents its logical structure to the user as a graphical representation of files and folders on a simulated desktop. Common operations, such as deleting a file, are carried out based on analogies to other operations in the physical world. For example, in many GUIs one can drag a file to a "trash can" in order to delete it. However, as Pierre Wellner [Wellner 1993] points out, even with this desktop metaphor, we are forced to deal with on screen information in a different way than we interact with objects on a physical desktop. With objects on the physical desktop, we can draw on a lifetime of experience in the physical world to help us understand how to use them, and our senses work together to provide rich information about the objects as we interact with them. On the other hand, graphical objects on screen cannot be touched with our hands, and we must rely on tools such as a keyboard and mouse to interact with them. While a keyboard and mouse are indeed useful for tasks such as word processing, at times they can force users to interact with information in a manner that seems complex or convoluted when compared to interacting with objects in the physical world. However, the computer's power also makes possible many operations that are not commonly available on a physical desktop, such as instant sorting, search and undo.

Digital Desk

With the Digital Desk [Wellner 1993], Wellner brought some of the functionality we typically associate with GUIs onto the physical desktop. This table used a camera and a microphone to detect finger presses on a graphical interface displayed on a desk with a video projector. Wellner used this desk for tasks such as graphic design and spreadsheet computations on physical paper. This system also employed some physical props, such as a scanner that would scan items and place them directly on the tabletop interaction surface. Much of the interaction relied on established GUI metaphors such as buttons and copy-and-paste. Wellner's research pointed the way toward enabling the computer to perform some of the operations we traditionally associate with GUIs in a tabletop environment. The Digital Desk also illustrated some of the compelling reasons for considering computer interfaces based on horizontal interactive surfaces. Because many work surfaces in our environment are already planar, horizontal or nearly horizontal surfaces, integrating computer interfaces into these surfaces may provide an opportunity for new types of relationships between computation and physical objects, and may help create computer systems that are more relevant to problem domains with established work practices based on tabletops.

Graspable Interfaces

Graspable Interfaces [Fitzmaurice 1996] use physical handles, such as six degree-of-freedom magnetic trackers, to grab and manipulate graphically-displayed information. These include interfaces such as GraspDraw, which provides multiple inputs to allow users to manipulate graphical forms to draw pictures.

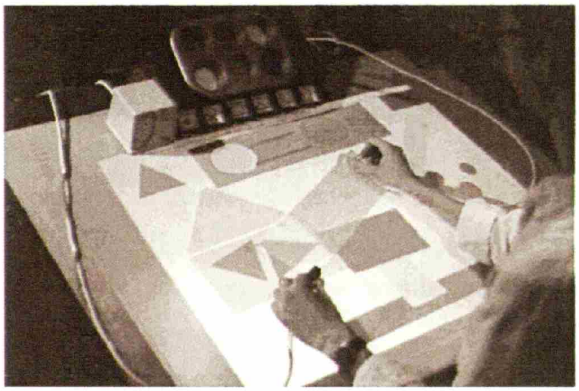


Figure 1: The GraspDraw application running on the Active Desk [Fitzmaurice 1996]

Fitzmaurice characterized Graspable Interfaces as having five qualities:

Spatial-multiplexing of output and input

Instead of using the same device for a variety of functions at different times, Graspable Interfaces tend to spatially assign different functions to different physical regions of the interface. This approach leads to more engagement of the user's motor abilities, and the tendency to represent objects and data in the application as physical objects that the user can manipulate.

Ability to use multiple devices simultaneously

The fact that there are typically multiple physical input devices means that users can often engage both hands, and the interface can support multiple users.

Use of specialized input devices

Graspable Interfaces emphasize the suitability of the physical aspects of the interface to the task being performed, such that the affordances of these objects can give the user clues about how they are to be used.

Spatial awareness of interface

The computer can track the position and orientation of physical objects in the interface and make this information available to application software. This functionality is particularly relevant in applications like drawing and computer aided design, because the input space of the physical interface can be directly mapped to coordinate space of the drawing or design task.

Spatial reconfigurability

Finally, Fitzmaurice points out the advantages of allowing users to customize the spatial arrangement of interface elements to suit the task at hand, and facilitate rapid switching between tasks [Fitzmaurice 1996].

Tangible Bits

In their Tangible Bits work, Ishii and Ullmer [Ishii 1997] also emphasized the value of specially designed physical objects in an interface that take advantage of skills people already have. "Tangible Bits allows users to 'grasp & manipulate' bits in the center of users' attention by coupling the bits with everyday physical objects and architectural surfaces."[Ishii 1997] It aims to "bridge the gap between ... cyberspace and the physical environment" [Ishii 1997].

One advantage of interacting with computers through physical objects is that users receive some passive haptic feedback from the objects as they grasp and manipulate them. Typically there are two feedback loops, as shown in figure 2. The passive haptic feedback loop provides the user with an immediate confirmation that he or she has grasped the object. The user can begin manipulating the object as desired without having to wait for the second feedback loop, the visual confirmation from the interface. This visual feedback loop takes longer because in order to respond to the user's actions

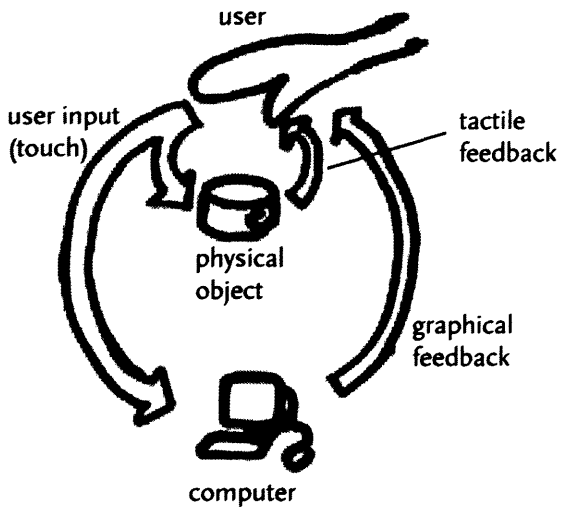


Figure 2: The double feedback loop in Tangible User Interfaces

it must first sense the user's input, then process that information in software and finally send the output to a graphical display. The implications of the passive haptic feedback loop are further explored in chapter 6.

Another key aspect of the Tangible Bits style of interface is that the input and output occur in the same physical space. Ishii and Ullmer refer to this concept as the "seamless coupling of bits and atoms" [Ishii 1997]. An example of this seamless coupling of is Underkoffler's urban planning project Urp, shown in figure 3 [Underkoffler 1997]. A series of architectural models serve as the input devices, and output in the form of a wind and shadow simulation is projected down onto the same tabletop surface, on top of and around the building models. Another notable aspect of Urp is its use of objects with very application-specific physical forms as a fundamental part of the interface. Physical building models represent the buildings themselves in the interactive simulation. Thus they give the user important visual and tactile information about the computational object they represent. Indicators such as a clock

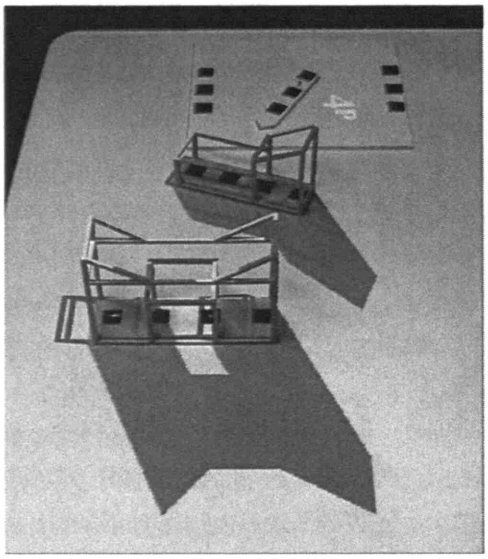


Figure 3: Underkoffler's Urp system for urban planning

and weather vane work in reverse in the Urp system. Instead of the clock hands moving to indicate the passage of time, the user can move the clock hands to change the time of day for the shadow study. Likewise, he or she can change the orientation of the weather vane to control the direction of the wind.

Sensetable

The author's Sensetable [Patten 2001] project explored the use of tabletop interactive surfaces for applications that did not feature an obvious spatial mapping such as that present in urban planning or sketching. These applications included supply chain visualization and musical performance. Sensetable incorporated objects that could be physically modified to control and represent state within the application such as buttons, dials, switches and tokens. For example, a button on top of a puck could start or stop an audio track in the Audiopad music system. Dials on top of pucks were used to control the values of parameters in a business simulation in the context of the SCVis supply chain visualization application. Tokens placed on top of pucks could change the electrical charge of atoms and molecules in a chemistry application.

Tabletop Tangible Interfaces

The diversity of interface approaches that incorporate tabletop interaction suggest that tabletop interaction with a computer may provide some compelling benefits over traditional GUIs. Some of these are:

Multiple users

Tabletop systems are generally able to accept input from multiple users at once. In contrast, the standard keyboard, mouse and display screen hardware was originally designed to accommodate one user at a time at a particular physical console. This distinction implies a difference in how the systems can be used in face-to-face collaboration scenarios. With a keyboard and mouse, switching control from one user to another can involve coping strategies such as one user telling the other what to click and type, or users trading chairs or passing the keyboard and mouse back and forth. In contrast, the effort required in many tabletop interfaces for users to "take turns" is comparatively low.

Coincident input and output

When tabletop interactive systems incorporate input and output in the same physical space, it becomes easier for onlookers to grasp the cause and effect relationships present between the user's input and the computer's response, because they can focus their visual attention on the table, rather than having to watch three separate spaces, (keyboard, mouse, screen) each with their own coordinate system.

Tactile feedback

When the interaction includes a group of physical objects on the tabletop surface, these objects provide tactile feedback when touched. We can grasp and manipulate these physical representations faster than we can analogous graphical representations with a touch screen or keyboard and mouse.

While these advantages are relevant for a variety of applications, tabletop interfaces based on physical objects also have drawbacks when compared to GUIs. Many of these center around issues of flexibility and consistency. In a GUI, the computer itself is the final arbiter of what is displayed on the screen. The software is free to change the graphical state of the interface to limit the user's choices, and prevent him or her from performing forbidden or invalid operations. For example, options can be "greyed out" from menus, or buttons can disappear or simply respond to mouse clicks with a beep instead of performing the desired action.

In many tabletop tangible interfaces, these types of dynamic changes to the interface are not possible. To avoid inconsistency between physical and digital state, the interface software must be designed to contend with every possible manipulation of the objects on the table, with the understanding that it may be difficult to forbid the user from doing any of them.

An alternative approach is to provide graphical feedback to the user when they violate a software constraint through their movement of a physical object. For example, if one were to move an object was supposed to be "locked" in position, the system could simply draw an arrow from the object's position to the required position, and possibly refuse to respond to other user input until this constraint were satisfied. However, this method would run the risk of defeating some of the advantages one normally gets from a TUI, such as being able to infer something about the state of the application using the sense of touch.

TUI designers have addressed this challenge in past applications by carefully drawing the line between information that will be represented physically and information that will be represented digitally. For example, early versions of the SCVis supply chain visualization

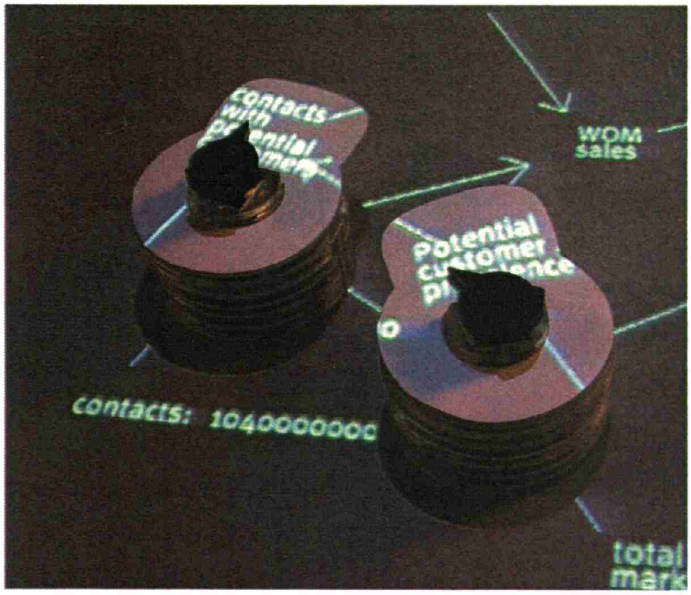


Figure 4: Physical dials in the SCVis system, running on the Sensetable, provide an added affordance but can become inconsistent with underlying digital information.

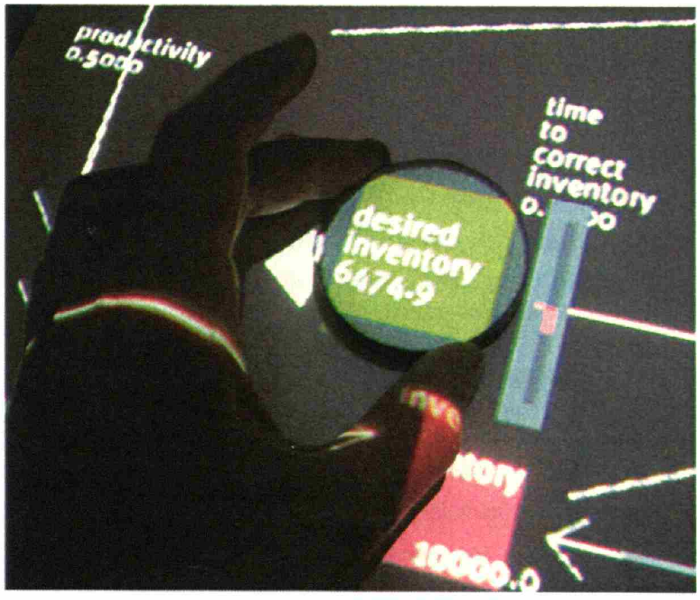


Figure 5: Graphical dials in the SCVis system address the consistency issue by using graphical interface elements instead of tangible ones.

system used physical knobs to change parameters within a continuously running business simulation. The knobs gave extra tactile and visual feedback about the state of the parameters they were controlling. However, the physical pucks could be "unbound" from their associated notes within the simulation and reassigned to others. Once this reassignment was complete, the position of the dial was often incorrect with respect to the value of the new parameter being controlled. Bill Buxton refers to this issue as the "nulling problem" [Buxton 1986]. One can change the state of the new simulation parameter to reflect the setting on the physical dial, or ask the user to reset the dial, or simply ignore the inconsistency, but none of these solutions is ideal. In the SCVis system, our experiences suggested that the benefits provided by having a physical dial with a position indicator were overshadowed by these consistency issues. As a result, we changed the way users adjusted continuous parameters in SCVis to a method using the rotation of the pucks themselves, with a graphical position indicator projected near the puck as shown in figures 4 and 5.

A related approach to inconsistency between physical and digital state is to design the application with a degree of simplicity such that inconsistencies can never occur. While this approach can enforce discipline on the interface designer, it can also limit the overall functionality of the application, because features that might lead to inconsistent interface states must be avoided altogether. For example, Urp's support for zoning rules is limited by the fact that the software has no way to prevent a user from placing a building in a location forbidden by such rules. Likewise, panning and zooming are not supported because the physical building models cannot change size under software control.

Closing the Interaction Loop

However, if one has the means to move physical objects on a tabletop under software control, one can eliminate many of these consistency issues, while opening up the possibility for a wide variety of other interactions. In an effort to address the issue of consistency between physical and digital state in tabletop TUIs, Gian Pangaro and Dan Maynes-Aminzade developed the Actuated Workbench [Pangaro 2002]. This system included an array of electromagnets below a position sensing antenna. These electromagnets could move physical user interface elements along the tabletop surface, enabling new types of application features, such as distributed physical collaboration, where the motion of a certain object on one table would cause the corresponding object on another table to move. Pangaro and Maynes-Aminzade also showed how one could pan and zoom on a tabletop TUIs while maintaining spatial consistency between the pucks and their associated digital information. Finally, they showed how the computer could correct the user's mistakes in a rule-based object placement task such as the eight queens problem.

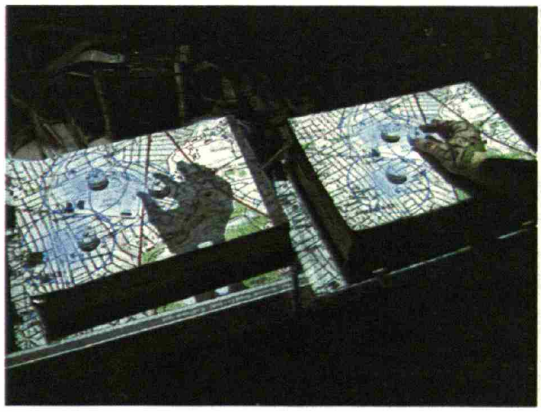


Figure 6: Pangaro and Maynes-Aminzade's Actuated Workbench. When the user moves an object on one table, the corresponding object moves on the other one. Note the projected shadow on the left table to give a sense of remote presence.

The Planar Manipulator Display [Rosenfeld 2004] takes a different technical approach to the same problem. Rather than using an array of electromagnets, the PMD uses a series of small battery powered robots that receive motion commands from a central computer. Perlin developed sophisticated path planning algorithms to efficiently control the movements of many of these objects at the same time, so that they could take a short path from one point to another without colliding with other robots that were doing the same thing.

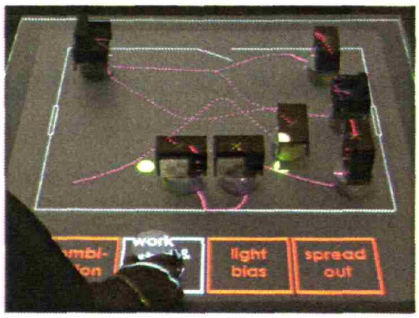


Figure 6: Rosenfeld's Planar Manipulator Display. Battery powered objects drive around under computer control.

Actuation has also played a various roles in other types of tangible interfaces. Figure 7 shows some different uses of actuation. In some projects, actuation is used as a visual (and perhaps auditory) display. These include the Dyna-Lux[Dahley 1998], Pinwheels[Ishii 2001], and Phidgets [Greenberg 2001] projects. In these systems the quantity to be displayed is typically a virtual one, not a result of other physical motion at another place or time. These systems could be considered ambient displays [Wizneski 1998].

Other systems deal with physical motion both as an input and output mechanism. These include Snibbe's haptic systems for media control[Snibbe 2001], InTouch[Brave 1998], Topobo[Raffle 2004], Curlybot[Frei 2000], Actuated Workbench[Pangaro 2002], and the subject of this thesis, Pico. With Topobo and Curlybot, the user

and the computer take turns controlling the motion of the same object in a record and playback function. Much work has been done with haptic feedback as a user-interface tool for applications ranging from scientific visualization to entertainment. Surgical interfaces and simulators have used haptic feedback to simulate the feel of tissue during medical operations[Madhani 1998]. In the GROPE system[Brooks 1990], Fred Brooks et al. used a 6 degree-of-freedom haptic display together with visual display to help chemists explore and understand how drugs "dock" onto the surfaces of proteins. The haptic display provided feedback about the forces between molecules. In some tests, they found that haptic feedback provided an extra two-fold performance improvement over systems using graphical feedback alone. In the GROPE system as with many haptic displays, feedback occurs through a single object that the user holds in his or her hand.

	the same object is normally used for input and output.	different objects are normally used for input and output.
input and output motions happen at the same time	Haptic Media Control Pico Surgical simulators	Actuated Workbench InTouch Pico Psybench
input and output motions happen at different times	Curlybot Topobo	

Figure 7: mappings of input and output motions in actuated tangible interfaces

Snibbe's work on haptic media interfaces [Snibbe 2001] involves the use of haptic feedback to aid media navigation, manipulation and annotation. Pico is capable of interactions based on combining the user's movements and mechanical constraints with the computer's actuation of the same physical object. Pico may also move other physical objects in response to user input to resolve a system of constraints. InTouch,

Psybench[Brave 1998] and the Actuated Workbench can also move a remote object simultaneously in response to the movement of object. This functionality provides the opportunity for tactile communication over a distance.

The categories above suggest another: interfaces in which physical movement is used as both an input and an output, but at different places and times. While I am not aware of interfaces that fit well in this category, it is an interesting one to consider. One possible application would be a variation on the Audiopad that recorded the user's motions during a performance, so they could be played back later, perhaps on a different Audiopad. As the motions were played back, the user could grab the pucks and override the recorded motions to produce a derivative work.

These systems introduce the notion of two-way control, where both the computer and the user can control the positions of objects in the interface. This ability can allow one to construct interfaces that take advantage of the dynamic physical properties of tabletop objects, but to date these interfaces have been used primarily to insure consistency between physical and computational representations, as well as to enforce constraints previously defined in software.

Supporting Psychology Literature

Kirsh's Epistemic and Pragmatic Action

A variety of psychological research supports the idea that these tabletop interfaces may better support certain tasks for which computers are often used. One example is David Kirsh's work on how people use space to accomplish different types of tasks [Kirsh 1995].

Kirsh makes a distinction between epistemic action and pragmatic action. The former is action taken to help one think about a problem, while the latter is action taken to actually solve the problem. Epistemic action is a way of "offloading computation"[Kirsh 1995] into the environment. For example, when counting a pile of coins one might separate them into two piles, ones that have been counted, and ones that have not. A bicycle mechanic might lay the parts of the bicycle down on the ground such that their spatial arrangement revealed the order in which they needed to be put back on the bike [Kirsh 1995].

Some of these techniques can be applied in the context of a graphical user interface. For example, Kirsh found that when playing the video game Tetris, people often rotate the bricks as they are falling because it is faster to do this and figure out where to place the brick than it is to do the mental rotation. However, this type of modification of one's environment to offload computation only makes sense when it requires less effort than performing the task in question without environmental modification. Thus as an interface becomes more complex to use, it supports epistemic action less well. Buxton's three state model of graphical input succinctly represents the complexity of interacting with objects in a GUI. First, one must grasp the pointing device with one's hand. Second, one must select the graphical object (e.g. an icon) to be manipulated with the pointing device. Finally, one can interact with the icon or other graphical object with the pointing device[Buxton 1986]. Fitzmaurice proposed a corresponding two-state model of graspable input. First one grasps the physical object, and then one interacts with it [Fitzmaurice 1993].

Guiard's Kinematic Chain Model

Evidence suggests that tabletop tangible interfaces can support two handed interaction to a greater degree than on-screen GUIs. One

line of research supporting this idea is Guiard's Kinematic Chain model [Guiard 1987]. This model relates to how people use their hands in two-handed tasks that involve asymmetric role division between the hands. Guiard found that in tasks that involve a tool and a target, the non-dominant hand often orients the target in space, while the dominant hand acts upon the target in the reference frame of the non-dominant hand. For example when writing with a pen and paper, people tend to orient the paper with the non-dominant hand, while writing on it with the dominant[Guiard 1987]. Hinckley found that people can perform tasks requiring manual dexterity faster and more accurately in this way than if the hand roles are reversed [Hinckley 1997]. Interaction with modern graphical computer interfaces often requires significant manual dexterity, as often a great number of functions and commands share limited screen real estate. Fitts' Law [Fitts 1954] shows that in these circumstances target acquisition time increases as the target size decreases. While tabletop tangible interfaces may also require significant manual dexterity from the user, the use of physical objects representing "tools" and "targets" on the tabletop supports asymmetric role division between the hands in the manner outlined by Guiard. While the role division of the hands in a screen-based GUI is also often asymmetric, the relation between the activity of the hands is based on the logic inside of the computer, rather than on the relative positioning of the hands in physical space. With a tabletop tangible interface, the non-dominant hand can provide a spatial reference frame for the activity of the dominant hand, while with a standard GUI this spatial reference frame does not exist.

Gray and Boehm-Davis' Microstrategies

In their paper Milliseconds Matter, Gray and Boehm-Davis introduce the concept of microstrategies [Gray 2000]. Microstrategies are combinations of the various primitive actions that an interface affords. For example, one can move a mouse, or click it. Thus two

possible microstrategies are to click the mouse and then move it, or move, and then click. Gray and Boehm-Davis point out that subtle differences in the amount of time an interaction takes (on the order of milliseconds) can drastically affect the type of strategy users employ with an interface [Gray 2000]. They point to several studies with dramatic results on this topic. Svendsen, O'Hara and Payne found that "when the cost of making a move in solving simple puzzles increased from one keystroke to several the strategy used to solve the puzzles shifted from one in which search was 'reactive and display-based' to one in which search was more plan-based" [Svendsen, O'Hara, Gray]. Other researchers found that subjects' strategies changed significantly even between conditions where eye movement was required versus head movement [Ballard 1995, Ballard 1997].

By breaking down an interaction task into elementary "cognitive, perceptual, and motor" operations, Gray and Boehm-Davis compute the critical path to completing a task based on a combination of primitive operations such as "perceive cursor location" [Gray 2000]. This analysis makes it possible to predict how changes in an interface will affect task performance times, and microstrategies. One case where their work suggests task performance times can suffer is in situations where the motor system must wait for the visual system to process information before proceeding [Gray 2000].

Planning vs. Experimenting

Given that the computational power of microprocessors has been increasing exponentially for years, while the computational power of the human mind has remained roughly constant, we should design interfaces that make it easy to offload computation to the computer from the minds of users. Kirsh, Gray and Boehm-Davis' research suggests that the faster one can execute the possible microstrategies presented by an interface, the less thinking the user will have to do to interact with it, and new types of problem solving approaches may

become plausible (e.g. iterative experimentation instead of planning). Relying on the sense of touch may be an affective way to pursue this goal. Humans are able to process some types of tactile information very quickly. The update rate considered acceptable for haptic rendering interfaces ranges from 200 Hz up to 1000 Hz [Burdea 2000]. For systems relying on graphical rendering, 30 Hz is often considered acceptable [Burdea 2000].

Tangible interfaces can provide certain types of tactile information very quickly. Consider the task of moving an on-screen object, either with a large graphical interface displayed on a table, or a tangible interface. With the graphical interface, the user must rely on a visual cue from the interface that he or she has successfully acquired the object to be moved with his or her finger. The user does receive a cue that his or her finger is touching the display surface, but this is not enough to know if the object has been successfully acquired. In contrast, with a tangible interface, the user immediately receives a tactile cue when he or she grasps the physical object. No intermediate sensing, computation or rendering need take place. The tactile response allows motor movement to take place without looking for further graphical confirmation, a more efficient microstrategy. More information on this comparison is provided in chapter 6.

Collaborative Interfaces

The term "Collaborative Interfaces" refers to the notion of the computer as a collaborator with unique skills, rather than just a tool that responds to commands from users. In contrast to approaches based on "intelligent agents" or "expert systems" or other approaches based on artificial intelligence, the Collaborative Interfaces approach emphasizes the computer's brute force computational ability, and leaves the user responsible for the higher level reasoning. In his discussion of "Collaborative Interfaces" [Shieber 1996], Shieber points out that many types of problems can be thought of as opti-

mization problems, such as "writing a (maximally) convincing memo, determining the (ideal) price for a product, constructing a (maximally) communicative diagram." As computers are unable to perform these kinds of tasks autonomously, some human guidance is needed. Shieber proposes letting users manage the global structure of the process while the computer performs local optimization. Shieber concludes that "the key to designing an interface then becomes representing the problem in such a way that this nice division of roles is feasible" [Shieber 1996].

One example of a collaborative interface is the Design Galleries (tm) system [Marks 1997]. Used for setting groups of animation parameters, Design Galleries allows users to explore a multi-dimensional parameter space as a three-dimensional graphical space on the screen. The computer's computational power is leveraged to populate the design space with many examples of possible parameter combinations and the results they provide. The user is left to explore this space and identify regions of it that correspond to desirable animation results, and thus desirable starting parameters.

The idea of collaborative interfaces is a compelling one for problems that are very computationally complex, yet cannot be solved autonomously by a computer. There are a variety of types of problems in this category, and along with them, a variety of reasons while computers cannot handle them alone. Some possible reasons include:

Desired result is difficult to quantify

Often computer optimization involves specifying a quantitative fitness metric with which to compare alternative solutions to a problem. If this fitness function is difficult to define or evaluate, it can be difficult for the computer to solve the problem alone.

For example, in the task of placing cellular telephone towers in a city, one must consider quantitative metrics of signal quality, but one must also consider sensitive and subtle political issues that are difficult to quantify.

Limited software tools

Computer optimization tools rarely reflect the depth of domain knowledge of experts who have been working on a given problem for decades. In such circumstances, experts often use the software tools as a starting point where one can begin a fine tuning process.

Enormous computational complexity

In some applications the space of parameters to be adjusted is so large that computer algorithms can greatly benefit from the user pruning the space by eliminating irrelevant or invalid parameter combinations that the computer itself would not be able to identify.

Spatial Applications

Many of these applications involve the placement of objects in physical space. Examples include:

Placement of rooms in a building

With this problem one wants to minimize the distance workers must travel to accomplish common tasks, while minimizing the construction and maintenance costs of the facility.

Placement of integrated circuits on a PCB

Many software packages exist for routing electrical paths between components on PCBs. However, layout applications usually are not able to place the actual components themselves.

CNC machining

One task in the manufacturing domain is the arrangement of parts to be CNC machined so as to use the smallest amount of raw materials possible.

Cellular network planning

The aim is to place and configure cellular telephone towers to optimize coverage while minimizing cost.

Placement of bus stops in a city

The goal is to provide easily accessible transportation coverage while minimizing travel times and cost.

Collaborative Tabletop TUIs

Given that many of these applications involve a strong spatial component, and that past tabletop TUIs have successfully address spatial applications, [Underkoffler 1999, Fjeld 1998], and that collaborative interfaces may benefit from an interface that effectively combines the strengths of user and computer [Shieber 1996], the collaborative interface approach is a compelling one to consider in the context of tabletop TUIs.

In Pico I explore this idea using an actuation system to provide opportunities for direct, physical collaboration between the user and computer. Past user interfaces based on computer controlled actuation have used the actuation as a means for the computer to move objects from point A to point B. If an object encounters a physical obstruction on the way from A to B, the system's control algorithms may attempt to compensate by using more force to move the object at the desired speed. Such an obstruction would normally be considered an error, an exception to the normal function of the application. However, if the positions of objects on the table

are considered to be a portion of the system's computational state, users could employ such obstructions to *collaborate* with the computer, opening up a rich space of possibilities between the extremes of doing exactly what the computer wants, or exactly what the user wants.

Constraints

Ullmer's work on TUIs explores a rich set of physical constraints to impart structure to physical arrangements in token systems. [Ullmer 2002] Ullmer often uses these constraints to help users formulate and adjust complex database queries. At times he refers to them as "interpretive constraints" because of their role in "mapping compositions of physical tokens to various digital interpretations." [Ullmer 2002] Ullmer also emphasizes the ability of computers to sense the position of tokens, and change the way the tokens are interpreted accordingly. For example, one might place a series of tokens representing different database parameters into a rack representing a database query. This action would be sensed by Ullmer's system, which would then interpret tokens that were immediately adjacent to one another as having an "AND" relationship, while other tokens would have an implicit "OR" relationship. Here the physical constraint is sensed by the computer and provides context to the motions the user is making. The constraint also limits the physical motions of the tokens to a predefined set of valid motions in the context of the application, preventing the user from manipulating the tokens in a way that has no valid interpretation in software [Ullmer 2002].

While constraints within Pico also serve to limit the physical motion of objects in the interface, their role within the system is different than in Ullmer's work. I refer to these constraints as "mechanical constraints" to emphasize their relationship to the movement

of objects in the interface over time. Mechanics is the branch of physics dealing with "the set of physical laws governing and mathematically describing the motions of bodies and aggregates of bodies." [Goldstein 2002]. The general concept is that users can add, remove and manipulate constraints on the tabletop to influence the way objects on the table move. The computer does not sense these constraints, rather it only senses the positions of objects that are being influenced by them. Because computer controlled motion is part of the software's real-time interaction loop, the results of the computer's attempt to move objects on the tabletop within the constraints established by the user are directly fed back into the ongoing computational process. In some circumstances these mechanical constraints can be thought of as performing computation, just as a series of gears can be used to perform multiplication. As the objects move on the tabletop, their motion as guided by constraints will "compute" an equilibrium between the various mathemati-

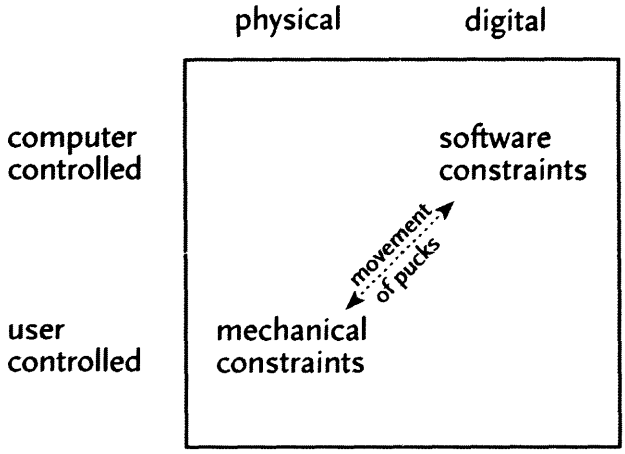


Figure 8: Mechanical constraints and software constraints combine to affect the motion of the pucks.

cal forces acting on the tabletop objects. For example, if two pucks that are trying to move simultaneously to two separate locations A and B are bound together with a rubber band, they will settle at a position near the midpoint between A and B. For more advanced examples of this concept please see chapter 5. A model for the relationship between physical and computational constraints is shown in figure 8.

In the Pico system, the motion of pucks on the tabletop is subject to the physical constraints imposed by objects on the table, as well as the user's hands. While the computer cannot sense these, it can sense the positions of the objects as they are affected by the constraints. The software application running on Pico may have it's own set of internal rules, heuristics and constraints that govern how it tries to move computational objects in the application i.e. cellphone towers etc. These rules may be arbitrarily complex and difficult for the user to change without reconfiguring the software. However, just as the position of the physical pucks influence the positions of their associated digital objects, the positions of these digital objects in turn reflect back upon the physical objects. The system continuously tries to keep the physical and software representations consistent, establishing a relationship between the mechanical constraints and the mathematical ones. This association between mechanical and mathematical constraints brings the cause and effect relationships between objects in the application software into the physical world, where the user can bring his or her mechanical intuition to bear on the task.

Because we experience the mechanical properties of objects on a daily basis, it is reasonable to expect users to have some intuition for how small moving objects on the tabletop will interact with each other. Mechanical constraints use this knowledge as a foundation for a variety of interaction techniques to guide and constrain mo-

tion on the tabletop. In his work on "Reality-based Interaction," Jacob argues that many recent interaction styles employ the strategy of leveraging preexisting knowledge and skills to make interaction easier, and that this strategy is a promising direction for future research [Jacob 2006].

Summary

A variety of past work in tabletop interfaces has demonstrated alternatives to the ubiquitous graphical user interface that take better advantage of the skills the people have developed in their lifetime of experience with the physical world [Wellner 1993, Fitzmaurice 1996]. One recent theme in the context of tabletop interfaces are interfaces using groups of tracked objects on an interaction surface [Underkoffler 1999, Fjeld 1998]. One limitation of this approach is that the interface designer must be careful to ensure consistency between physical and digital state in the interface. Actuation of the objects on the tabletop is one solution to this problem [Pangaro 2002, Rosenfeld 2004].

Recent work in tangible interfaces has also investigated the use of physical constraints to help users and computers interpret the meaning of spatial relationships between objects in the interface [Ullmer 2002]. Another possible use for constraints in a tangible interface is to map them directly to mathematical constraints in the context of an application using actuation. With this approach, one's intuition about the behavior of mechanical systems in the physical world would become relevant to the application at hand. These mechanical constraints could be used to guide the computer's efforts in the context of a collaborative system in which the user focuses on the high level structure of the problem to be solved, while the computer focuses on the details.

Research suggests that people will readily change their problem solving strategies when using computer interfaces on the basis of differences in the time to complete a task on the order of milliseconds [Gray 2000]. When it is more efficient to do so, subjects offload computation onto the interface itself to help solve a problem [Kirsh 1995]. If Pico could make it easier for users to express their intentions by leveraging users' mechanical intuition, it might encourage users to change their problem solving strategy to one in which Pico handles more of the work, and the user provides high level guidance using mechanical constraints.

4 Implementation

Hardware

The development of Pico involved the combination of the existing Sensetable [Patten 2001, Patten 2002] hardware platform with newly developed actuation technologies, and newly developed software. A high-level system diagram is shown in figure 1. The Sensetable is used to determine the positions of objects on the tabletop surface. An array of electromagnets is used to move these objects. Application software draws graphics and responds to user input, while several pieces of middleware allow these various system parts to communicate.

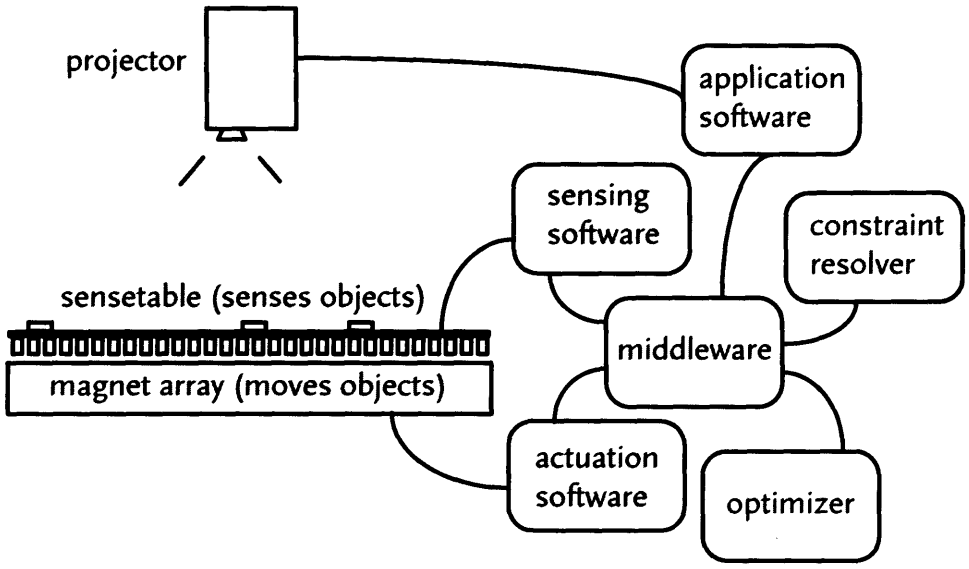


fig. 1. system architecture

Actuation

One key component of the development is a system for moving objects on a tabletop surface. Several existing systems already do this. Rosenfeld's Planar Manipulator Display [Rosenfeld 2004] uses a series of small battery powered robots that receive motion commands from a central computer. The robots are placed on a translucent horizontal surface, where they can move using a set of wheels. Each robot has a set of batteries and a small LED on it's undercarriage. A two dimensional position sensitive detector tracks the position of this LED. A computer plans the motion of the all of the robots on the table, and transmits motion commands to them. Reznik and Canny's Universal Planar Manipulator [Reznik 2001] uses an ultrasonic approach to move a series of small objects on a tabletop surface. Interference patterns on the tabletop would create vibrations that could be controlled to move multiple objects in arbitrary directions at the same time. Pangaro and Maynes-Aminzade's Actuated Workbench [Pangaro 2002], uses an array of electromagnets below the interaction surface. When coupled with a sensing system such as the Sensetable or a video camera, objects with embedded magnets could be moved by triggering the correct combinations of electromagnets.

Design Considerations

Holomonic drive

One of the fundamental aspects of the motion of physical objects in Pico is that objects in the interface must respond to physical constraints in a way that is transparent to the user. One aspect of this transparency is that objects must be able to move equally easily in all directions. For this reason, a propulsion system based on a series of parallel wheels, such as in a car, would not be acceptable as it moves easily in the direction of the wheels, and not very easily when moved perpendicular to this direction. There are holomonic drives for robots that do have this property, however.

Size of table and objects

The ratio of the size of the interaction space to the size of the physical objects within it is an important factor in its utility. As the objects become larger, it becomes more difficult to precisely arrange them in relation to each other without running out of workspace. As well, the diameter of an object may be too large to unambiguously identify the location of the object it represents in the context of an application. For example, it would be difficult to place cellphone towers on a map of a city when the pucks representing the towers were each several city blocks wide on the scale of the map.

The ratio of these sizes aside, one generally would not want the table to be so large that one could not comfortably reach all parts of it, and one would want pucks of a size that could be easily grasped. Pucks from 1" to 2" in diameter work well for this purpose.

No batteries

Batteries in the objects should be avoided if possible, to avoid the need to replace or recharge them.

The Actuated Workbench

These constraints are similar to the ones described by Pangaro and Maynes-Aminzade regarding the Actuated Workbench system. The primary additions to their design considerations for the construction of Pico were the ratio of object size to surface size, and the need for pucks to move equally easily in all directions. Given that the Actuated Workbench already satisfied the latter of these, I decided to use its design as the starting point of the actuation system for Pico. The main technical issues to be addressed with the Actuated Workbench were scalability and robustness. Parts in the Actuated Workbench system fail at a rate that made producing a larger replica of it difficult. As well, the field generated by the magnets is weak enough

that sometimes it is not able to move objects on the table. Finally, the system requires control circuitry and wiring that rests on the table next to the magnet array, making tiling of multiple arrays difficult.

In order to resolve these issues I redesigned the actuation hardware to be more scalable and robust. Electromagnetic coils are driven with a grid of N and P channel MOSFETs, controlled by an Atmel AVR 8 bit microcontroller. The circuitry exists on three separate circuit boards, A, B, and C in figures 2, 3 and 4.

The electromagnets are mounted to the top of circuit board B with a stainless steel mounting bracket. A large bipolar capacitor, critically damped with the electromagnet, is mounted in parallel with it, to make the switching of the magnets more efficient. Board B is designed to support driving the magnets in both directions. However, in practice the ability to repel objects on the table was not used in any Pico application, so much of the circuitry for this mode has been left unpopulated on the board. Each magnet has an N-channel and a P-channel MOSFET associated with it which controls the flow of current into or out of one side of it. The other side of all of the magnets is connected to a single N-channel, P-channel pair, which can enable or disable all of the magnets on the card. This MOSFET pair is driven by an International Rectifier IR 2181S driver chip, which is designed to drive two N-channel MOSFETs. The output of one of the N-channel MOSFETs controls the P-channel MOSFET. These inputs are controlled by logic level inputs from board A. The remainder of the MOSFETs are controlled by 35V-45V signalling inputs from board C.

Board C contains a series of IR2181 chips that switch the 5V logic inputs from board A up to the 35V-45V range necessary to control the MOSFETs. It connects to board A with a ribbon cable, which carries the 32 control signals for the 16 magnets on board B. The main purpose of Board C is to separate the higher voltage circuitry from board A.

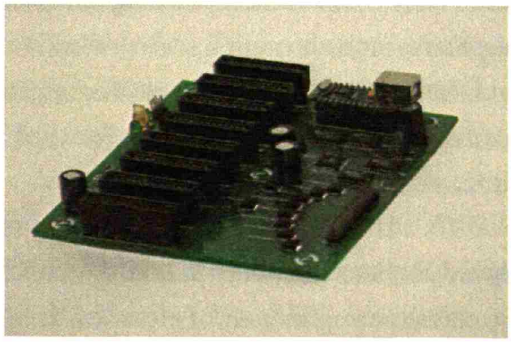


Figure 3: Pico board A connects the computer to the magnets using USB, and sends switching signals to the magnets.

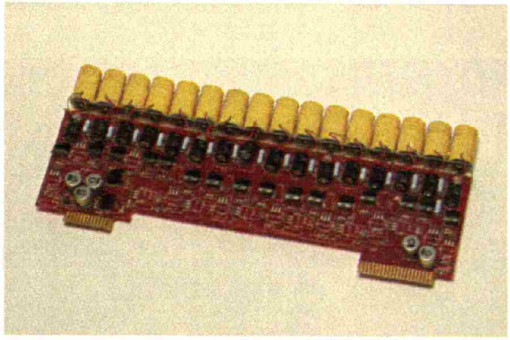


Figure 4: Pico board B contains the electromagnets and associated higher voltage circuitry.

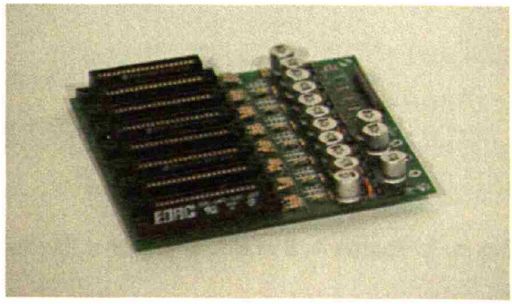


Figure 2: Pico board C connects the mosfets on board B to board A's logic outputs.

Board A contains the microprocessor that controls the entire system. It is an 8-bit Atmel AVR AT Mega 32 processor, which connects to a desktop computer by way of an 8-bit parallel data bus to a USB245M USB controller board. Data is transmitted to and from the computer using a USB 1.1 connection. Mainly the computer sends a series of commands about which magnets to turn on, and at what duty cycle. The AVR chip turns magnets on and off by clocking their states through a series of buffers and into a series of NOT and AND discrete logic gates. These gates make it impossible to inadvertently turn a magnet on forward and backward at the same time, which causes a short.

Board B slides into slots in boards A and C. This is intended to ease troubleshooting in case of electrical failure. A suspect board B can be replaced quickly to determine whether an electrical fault exists in the control circuitry or on the board B itself.

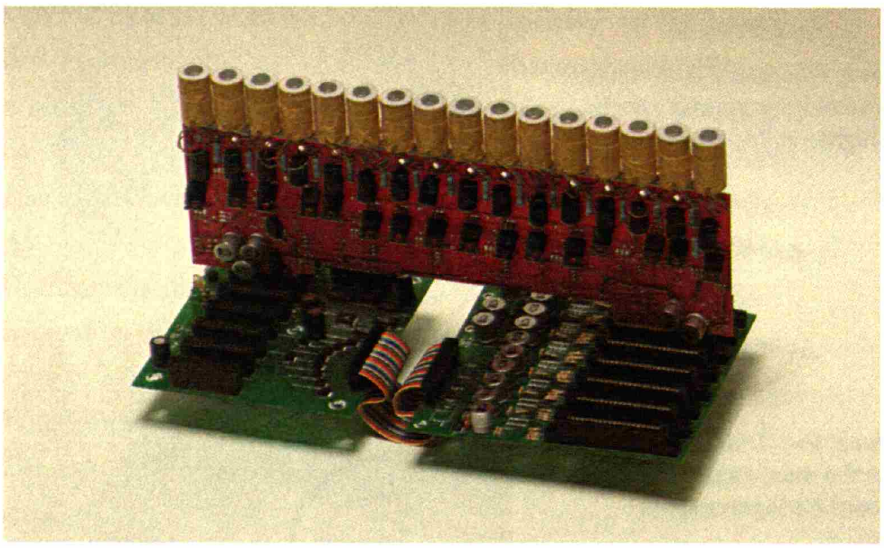


Figure 5: Boards A, B and C fit together like this.

Ratings

The lowest rated component in the system is the capacitor in parallel with each magnet, which is rated at 50V. Typically the system is driven at 45V using a supply capable of producing 5A of current. This 5A current limitation protects other components in case of electrical failure. The lowest rated component in terms of current is the card edge connector that connects board B to board C, where each connection is rated at 3A. Two connections are used in parallel to carry the high voltage and ground supply lines.

Firmware

The desktop computer transmits 50 magnet updates to the AVR chip each second. Each update contains the desired state and duty cycle for every magnet in the 16x8 array. The AVR adjusts the magnet control lines based on these commands, and as a safety feature, shuts down all magnets if it does not receive any commands for 4 seconds.

During the magnet drive process, each row in which magnets are to be used is turned on, while the corresponding column MOSFETs are also enabled. When both of these events happen at the same time, the electromagnet corresponding to the given row and column is enabled.

Because the magnets are arranged in a grid, many computer graphics ideas are relevant if one thinks of the magnets as pixels. Essentially, the computer transmits a frame to be rendered, and the AVR renders that frame to the magnets repeatedly until a new frame is received. The AVR refreshes the magnets at a rate of roughly 400 Hz.

Design Issues

Overheating

Due the high levels of current involved, overheating is a concern when the electromagnetic array is driven for an extended period of time. This issue is addressed by software that continuously monitors the duty cycle of each magnet in the array as it is turned on and off. If a given magnet exceeds a 25% duty cycle for more than 30 seconds, it is turned of briefly to give it a chance to cool down. Typically, this rarely happens except when the user is pulling an object or blocking it for more than 30 seconds. Users may notice this duty cycling as a reduction of the system's magnetic pull of a puck they are holding,

Calibration

In order to know which magnets must be activated to move a given puck to a desired position on the interaction surface, one must have precise measurements of the position and orientation of the sensing surface with respect to the magnet array. Pico performs this calibration automatically upon startup as described below. For this autocalibration to work, the system requires that only one puck be on the table when the system starts. First the magnet array activates each column of magnets in sequence from right to left. Then the leftmost two columns of magnets are activated in sequence. These two series of activations have the effect of moving the puck to the far left corner of the actuation surface even though its initial position is not known. A position reading is taken at this point with the Sensetable antenna, and this process is repeated for two more corners of the surface. With three known points in the space it is possible to create a calibration matrix with which one can multiply points in one coordinate space to achieve points in the other. Computationally, this approach is somewhat more intensive than algorithms to calibrate touch screens, or digitizers, because these algorithms typically assume no rotational offset between the two coordinate spaces. Because there are only

three calibration points, the magnets must be evenly spaced for this calibration scheme to work well. Braces on the sides of the circuit boards help ensure even spacing.

Tiling

Each 8 x 16 magnet array is controlled by its own independent USB connection. These smaller arrays are mapping into one large array for control purposes.

Duty cycling

The software supports four different magnet power levels which are obtained by turning the magnets on at varying rates under the control of the AVR CPU on board A. This increases the precision with which the system can position objects from about 1 cm with the original Actuated Workbench design [Pangaro 2002] to about 2 mm with Pico's magnet array.

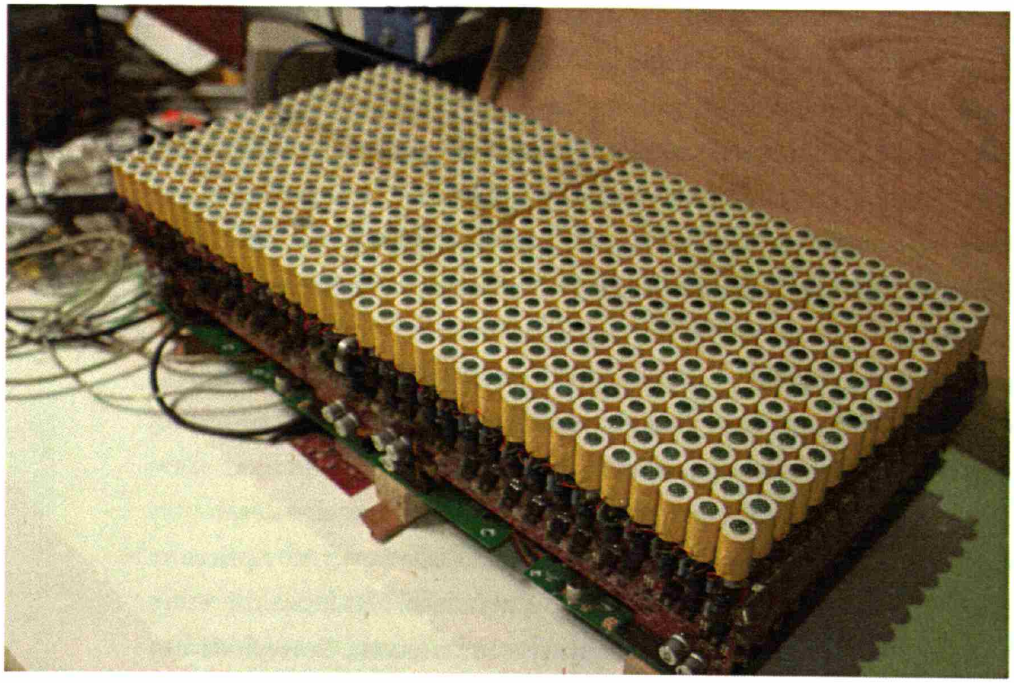


Figure 6: The completed magnet array

Dimensions

Each board B contains 16 magnets in a row. Each is roughly .75" in diameter leading to an overall length of 12" for the group. A single board A and board C can accommodate 8 board Bs, yielding an 8 x 16 magnet array measuring approximately 6" x 12". Four of these 8x16 arrays are tiled in a 2x2 configuration to yield an overall size of 16x32 magnets, and 12" x 24". The total number of magnets in the system is 512.

Sensetable

The original Sensetable prototype was developed as part of my master's thesis at MIT. The original version used modified wireless mice from a Wacom digitizer tablet to sense the positions of several pucks on a table. While a standard Wacom tablet can track up to two objects on it's surface at a time, the Sensetable used custom-designed hardware to track six objects simultaneously. The custom hardware rapidly enabled and disabled the inductor used to harvest power from the sensing surface in a random fashion. Because each of the pucks had a unique digital ID, custom software was able to disambiguate the position readings from the various pucks, though the period of continuous tracking for each tag was around 0.2 seconds. The custom circuit also included a capacitive touch sensor, so that as it was moved by a human hand, it would keep the inductor powered for longer periods of time, ensuring smoother tracking.

The position sensing system used for Pico involves a modified LC tag sensing antenna from a children's toy called Ellie's Enchanted Garden, which was once produced by the Zowie Intertainment Corporation. Media Lab sponsor NTT Comware has also developed a commercial version of the Sensetable. However, at a price of approximately $5-10 online, Ellie's Enchanted Garden is the most cost-effective approach for this type of research. Several identical circuits removed from the Zowie "Ellie's Enchanted Garden"

playset are aligned next to each other on a flat surface with a slight overlap to provide a larger sensing area with no gaps. Each of these circuits consists of some digital communication and control circuitry and circuitry to excite and sense excitation on eight antenna loops. Four of these loops are for sensing the position of tags in the X dimension, and four are for the Y dimension.

The objects tracked by the system contain LC tags, which are an inductor and a capacitor in series. These tags resonate at different frequencies according to their inductance and capacitance. In the case of the Zowie system, there are nine unique frequencies, ranging from 790 kHz to 4.59 MHz. The system locates these LC tags using several steps. It sends out an excitation pulse consisting of several cycles at the resonant frequency of the tag. If the tag is close to the antenna plane, ideally with the axis of symmetry of its coil perpendicular to the antenna plane, the tag will magnetically couple with the exciting antenna. This induces EMF in the tag, causing it to oscillate and generate its own magnetic field. The system then measures the amount of coupling between this tag and several receiving antennas. Through ratiometric comparison of the excitation of these antennas, it is possible to determine the position of the LC tag.

Perhaps the most clever aspect of the design is the antenna geometry. As mentioned above there are four antennas used to compute the position in each dimension. These are divided into two pairs, one of which is used for fine grain positioning, the other for coarse grain positioning. The antennas are wound such that the coupling with an LC tag varies sinusoidially as a function of the tag's position. For each pair of antennas, the period of this excitation is the same, but the phase is offset by 90 degrees. Essentially one varies as a sine function of position, while the other varies as a cosine. This means that one can compute the position of the tag by taking the inverse tangent of the ratio of the excitation of a pair of antennas.

In the case of the coarse grained antenna pair, the periods of these sinusoidal functions are the length of the sensing surface itself. For the fine grained antenna pair there are more periods. The sensing area is rectangular, and the shorter dimension has periods in the fine grained antenna pair oscillation, while the longer dimension has nine. One can use the coarse measurement to determine which period to consider for the fine grained measurement.

A specific antenna geometry is used to cause the excitation to vary sinusoidally as described above. A diagram of this geometry is shown below. This diagram only shows one antenna loop. The others will have different patterns as a function of the desired periods and phases. As the tag moves across this geometry, it moves from an area where it is completely inside the antenna loop (shown in white in the figure) to one where it is outside the loop (light gray), and then back inside the loop, but this time the path of the wire loop around the antenna travels in the opposite direction (shown in dark gray). The excitation of the receiving antenna by the LC tag is 180 degrees out of phase between the white and dark grey areas in the figure. As the tag moves over the light grey areas in the figure, a portion of the area of the coil inside the tag is in a white area, while another portion is in the dark grey area. This leads to a resonance value between the two extremes. An important size relationship here is that the diameter of the inductor in the tag should be double the spacing labeled as d/2 in the figure. Also the spacing for areas where the tag is inside the antenna loop, labeled as d in the picture, should be roughly the diameter of the

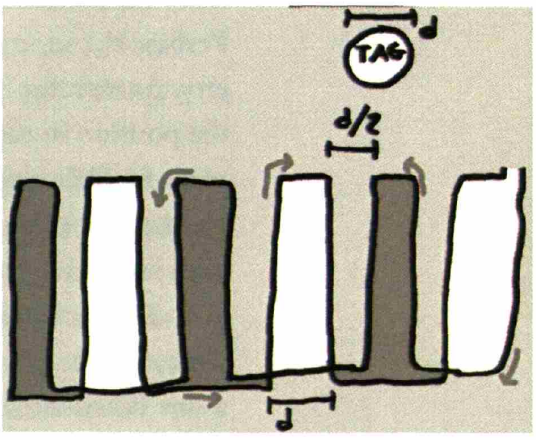


Figure 7: A special antenna geometry creates a resonance that varies sinusoidally in amplitude as a linear function of tag position.

LC tag. This helps yield a smooth sinusoid as the tag moves across these boundaries. The spiral weave of these antenna loops also helps cancel out external interference.

The system can also sense a small amount of Z position information. While the ratio of the intensities of the excitations of the receive antennas can be used to compute X and Y position, both of these intensities will decrease as the tag is lifted off of the sensing surface. By measuring this decrease one can tell when a tag is lifted off of the surface. However, this measurement is less reliable than the X and Y measurement because the amplitude of the resonance varies as a function of the tag frequency and can also decrease if the tag becomes detuned over time, or approaches one of the edges of the sensing surface.

Interfacing the Sensetable hardware with the Actuated Workbench hardware complicates the tracking problem to an extent, because both the electromagnets in the Actuated Workbench, along with the permanent magnets inside the pucks themselves change the way the tags resonate. Based on tests with a network analyzer, the per-

Figure 8: A Pico tag with an LC resonator in the middle and four rare-earth magnets around its circumference

manent magnet seems to saturate the inductor inside the tag, effectively decreasing the inductance. This decrease in inductance increases the frequency of the resonance, while damping it somewhat as well. The electromagnets underneath the sensing surface reduce overall amplitude of the resonance, without shifting the frequency much. While the original Actuated Workbench used permanent magnets placed at the center of the inductor in the LC tag, in this prototype I avoided some of these resonance issues by placing a series of tall, thin magnets outside the diameter of the inductor. Due to their position and alignment, these magnets interfere with the tracking system to a much smaller degree. These cylindrical rare-earth magnets measure 4 mm in diameter and 12 mm in height. Four are placed along the outside edge of the tag, as shown in figure 6. With the signal strength issues caused by the permanent magnets almost completely alleviated, one can simply ignore the interference caused by the electromagnets and still have sufficient signal quality to reliably track the positions of objects on the interaction surface.

One challenge with overlapping multiple sensing antennas is preventing interference between them. In the Sensetable system, a sophisticated piece of software ensures that two adjacent antenna arrays never poll for the same tag frequency at the same time. This software, called the Board Manager, keeps a running estimate of which antenna array will be most likely to find the position of each given tag. This estimate is determined by the last known tag position, as well as its speed and direction of travel. To compensate for the limited number of tags

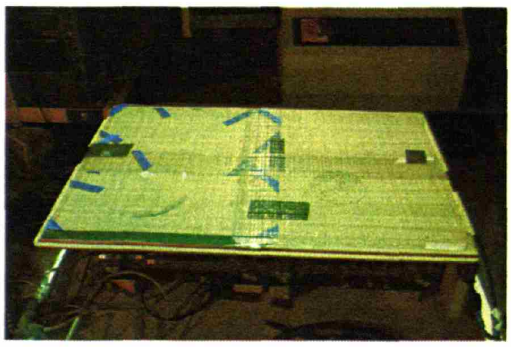


Figure 9: The Pico hardware, partially disassembled to reveal the overlapping sensing antenna arrays.

that can be sensed in a given time (approximately 40 per second in the current system) the Board Manager also prioritizes the tags, reading the positions of stationary tags less frequently than the positions of moving ones. Application software can also request that specific tags be given a higher tracking priority than others.

Magnet Control Software

Before the construction of the new version of the magnet array, I anticipated that changes in the Sensetable control software would be required to reliably control the movement of objects on top of the magnet array. The concern was that without more rapid feedback from the Sensetable into the control loop, the control loop would not be fast enough to avoid instability. However, this instability turned out not to be an issue.

The control loop works as follows: If the software is trying to move an object from point A to B, it first finds a point that is 0.8 units away from the object's current location, where each unit is the diameter of one electromagnet (0.75"). This point is located inside of a square, such that the corners of that square are defined by four electromagnets that are adjacent to each other. Based on the distance of the point to each of the four corners, a duty cycle for each of these four magnets is calculated, such that the sum of the forces will draw the puck toward the point. These duty cycles are sent to the control hardware, which then turns the magnets on and off appropriately. With a standard control loop, an opportunity for instability might arise at this point, due to the chance that the puck might undershoot or overshoot its destination. However, as the puck approaches the electromagnets that are pulling it, the strength of their attraction increases, and a larger component of that force is pulling the object down, perpendicular to the direction of travel. This downward pull increases the friction of the puck with the interaction surface, slow-

ing the puck down as it reaches it's goal. When new position information about the puck is received from the Sensetable, the control software selects a new goal point for the motion of the puck. If the puck has not moved, the new goal point will be the same as the old one. If the puck has moved closer to its final destination, then so will its next goal point.

One important result of this simple design is that the control system is robust in the face of mechanical obstructions on the interaction surface. This robustness is important because adding and removing mechanical constraints on the tabletop is one of the primary ways that one interacts with Pico. Rather than changing the force applied to the puck if an obstruction is encountered, the control software maintains a constant force, trying to move the puck in the desired direction. The resulting motion of the puck is determined by the physics of the interaction of the puck with the various forces upon it, the magnetic force, as well as the friction, mass and force resulting from other entities, such as the user's hand, or a heavy object placed on top of the puck.

In the case that the ultimate destination of the puck is less than 0.8 units from the current object location, a different actuation pattern is used to move the puck more slowly, preventing overshoot. This actuation pattern drives the magnets at a lower frequency (about 10 Hz), inducing a mechanical vibration in the puck which slowly moves it into position. Using this method, the current prototype is able to move objects within the precision afforded by the Sensetable (currently about 2mm). Without this technique, the positioning accuracy is about 1cm, as with the original Actuated Workbench.

One addition to the Sensetable control software added in the process of building Pico was more sophisticated filtering code for the raw position data read from the sensing antennas. Past versions of this software use a thresholding scheme, in which position data is ignored

if it is too different from the last position observed. This had the effect of making the system occasionally unresponsive when objects were moved rapidly across the interaction surface. The new model, rather than considering distance alone, also considers velocity and acceleration. Given that the minimum mass of a puck is known in advance, one can predict that its velocity and acceleration will generally fall within certain bounds, given the forces expected to be acting upon it. Using this assumption, one can reject readings that indicate a sudden acceleration, while allowing ones that show more reasonable acceleration values, even if this movement results in a large travel distance between sensor readings. The actual threshold values used were determined experimentally by quickly moving a puck from one side of the interaction surface to the other from a stationary position. The units of the acceleration observed using this technique are pixels/s/s, where a pixel measures about 1mm, depending on the projector alignment. Based on this experimentation a value of 20,000 pixels/s/s was obtained, which is roughly 2Gs. This value could be adjusted to be more responsive to extremely fast motions while allowing more noise, or to allow less noise, while requiring slower motions.

Application Software

The application software is responsible for displaying information on the tabletop, based on data received from the optimizer and the sensing software. Its architecture is similar to an application for the Sensetable system, which in turn is based on the "rendering loop" model often used with the OpenGl graphics library. The pseudocode for a Pico application is as follows:

```
def renderCallback():
  drawApplicationUI()

def movementCallback(puck):
  doSomethingApplicationSpecific(puck)

def idleCallback():

  pucks = tagTracker.service() #Update object positions using
Sensetable tracking data.
  for p in pucks:
    movementCallback(p)
  runApplicationOptimizer() #Iteratively improve the spatial layout
with application specific code
  requiredMovements = constraintManager.resolve() #Find pucks whose
positions are violating a constraint, and compute motions necessary
to resolve them
  for puck,destination in requiredMovements:
    awb.updateMotion(puck, destination) #Set each puck on a path
toward resolving currently unresolved constraints
  if awb.isReady(): #Make sure we are not flooding the magnet hard-
ware with data
   awb.render() #Send control commands to the magnet hardware
```

Most of the extra behavior required of a Pico application that does not exist in a standard on-screen OpenGL application happens in the OpenGL idle callback. The application must periodically read data from the position sensing hardware, run the constraint engine to detect any constraints that are being violated, compute the actions necessary to resolve those constraints, and periodically issue the corresponding commands to the Actuated Workbench server, which in turn relays them to the magnet array itself.

For the cellular tower planning application, this system displays tower locations and associated parameters such as elevation, output power, tilt, azimuth, etc. It also displays other data such as ground elevation, "urban clutter" produced by nearby buildings, etc. The user is be able to change most of these parameters by interacting with the interface. The interaction techniques used in this software are described in the next chapter.

Optimizer

The optimizer is a software module specific to the application domain that performs a parameter space search for the best parameter values according to a given fitness function. For example, in an application regarding the placement of machines on a factory floor, the primary parameters would be the x and y position of the machines as well as their orientation. The Pico software architecture is designed to minimize the software development necessary to connect a batch optimizer designed for non-interactive use into the system. However, an optimizer must meet several requirements to be usable with Pico:

Fitness function

The optimizer must have a function for quantifying the "fitness" of a particular set of parameters, and that function must be able to run at an interactive rate. This function will be called periodically as potential movements of objects on the table are considered.

Iterative approach

Rather than taking a set of parameter values and searching at length for an optimal solution, the optimizer must be able to take a set of starting conditions and return a similar, but incrementally better configuration than what it started with. When applied iteratively, this type of incremental convergence in the parameter space leads to smooth motion of objects on the table that is easy for users to understand and modify.

If an optimization algorithm is not well suited for this type of iterative use, one approach to making it so is to build a helper function that compares the value of the fitness function for the current best solution to that of each of a group of nearby points in the solution space. Based on the fitness of each of these alternatives, determine the gradient toward the best solution, and adjust the parameter values by a small amount in the direction of that gradient. A well defined interface is used to connect this software with other parts of the system to reduce the amount of effort required to replace one optimizer with another.

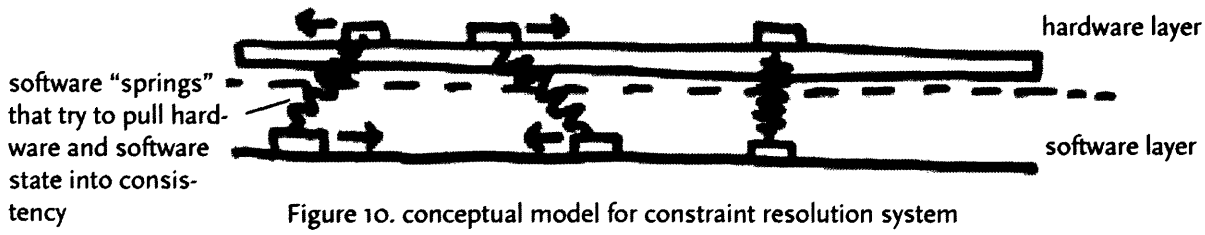


Figure 10. conceptual model for constraint resolution system

Constraint Resolver

The constraint solver in Pico is not a "solver" per se, in that it does not aim to satisfy all constraints present in the system at all times. Its main purpose is to pursue consistency between the internal state of the optimizer and the state of the user interface.

Its model for doing this is conceptually similar to figure 10. Each parameter consists of a pair of values: the optimizer's current value of the parameter, and the interface's current value of the param-

eter. The optimizer's current value is projected graphically on the interaction surface, while the interface's current value is embodied by the position of the puck. Connecting those parameters is a "computational spring" that continually tries to make the two parameters equal. It does this by iteratively computing the average between the two points, and trying to move the two points a bit closer to this average each time. The farther away the two points are from each other, the harder it tries to bring them together. This constraint model in Pico replaces the "binding" model in Sensetable. In Sensetable, the graphical representation of data can be "bound" to a particular puck, in which case it moves wherever the puck moves until it is unbound. Pico also tries to keep the puck and the graphical representations in the same place, but does so in a more elastic manner.

This functionality is implemented on top of a general purpose constraint engine written in Python. Each type of constraint inherits from a general purpose constraint class, and must implement a method which takes the positions of the object affected by the constraint and determines if the constraint is satisfied. If the constraint is not satisfied, the method returns a set of adjustments to the current object positions that would satisfy the constraint.

The constraint solver treats constraints that have been violated for a long period of time differently than those that have been broken only briefly. Initially I had planned to develop code specifically to support this functionality, but the magnet duty cycle monitoring software indirectly accomplishes this task. If the system tries to move a puck for more than 30 seconds (less in some cases) without being successful, the magnet monitoring software will temporarily shift them to a lower duty cycle, reducing the pull on the puck in question. The result of this reduction is that if the constraint resolver cannot resolve an inconsistency by moving a puck and its digital association together toward their midpoint, it will subsequently try moving the digital association more than it moves the puck.

To be usable, the Pico constraint resolver needed to produce results that were deterministic and continuous. While there may be a variety of possible solutions to a given constraint problem, the iterative nature of the software requires that similar solutions be given for similar inputs. Otherwise, one might observe pucks frantically moving around on the table in response to subtle changes in application state, making the system difficult to understand and interact with. This requirement was one of the reasons a new constraint engine had to be written from scratch, instead of using an existing one. Some existing constraint engines, such as Cassowary [Badros], can sometimes return non-continuous solutions for constraint problems with continuous input.

5 Interaction Techniques

The interaction techniques used in the Pico system were developed through a period of experimentation over several years with various types of techniques for use of tabletop interfaces, both with and without actuation. While graphical user interfaces (GUIs) have a set of generally accepted interface building blocks, such as buttons, sliders, menus and windows, tangible user interfaces lack an analogous vocabulary. The development of generally applicable interaction techniques is an important step toward being able to use tabletop tangible interfaces as a general purpose tool for solving abstract problems.

Initially, the interaction techniques I developed centered on the ability to augment Sensetable pucks with additional interface hardware, such as buttons, dials and interchangeable tokens. Over time, I moved to a more simplified system, in which all of the application functionality was accessible by moving the pucks into different positions relative to each other. One prime motivator for this change, as well as an example of its progression is the Audiopad system[Patten 2002]. Audiopad is a system for live musical performance, a very demanding context for developing and testing a new interface. In this context, it was difficult to quickly access functionality mapped to buttons in Audiopad, so I switched to an alternative method which did not use them, described below. This section first discusses techniques that are relevant with or without actuation, and then discusses techniques specifically relevant to interfaces that include actuation.

Sensetable Techniques

Over several years I have developed a variety of tabletop TUI applications such as musical performance and business simulation using tabletop tangible interface platforms, such as RF tracking systems like the Sensetable[Patten 2001] and computer vision-based object tracking platforms. In the context of these applications, I have developed and evaluated a variety of interaction techniques which can be applied to other applications. These include techniques for modifying continuous and discrete parameters, and navigating hierarchical datasets. With these techniques it has been possible to include features in tabletop TUI applications that previously seemed difficult to do.

The Applications

The two primary applications in which I developed and tested these techniques were electronic music performance and business supply chain visualization. I developed each of these applications over several years, experimenting with new sensing technologies and interaction techniques along the way.

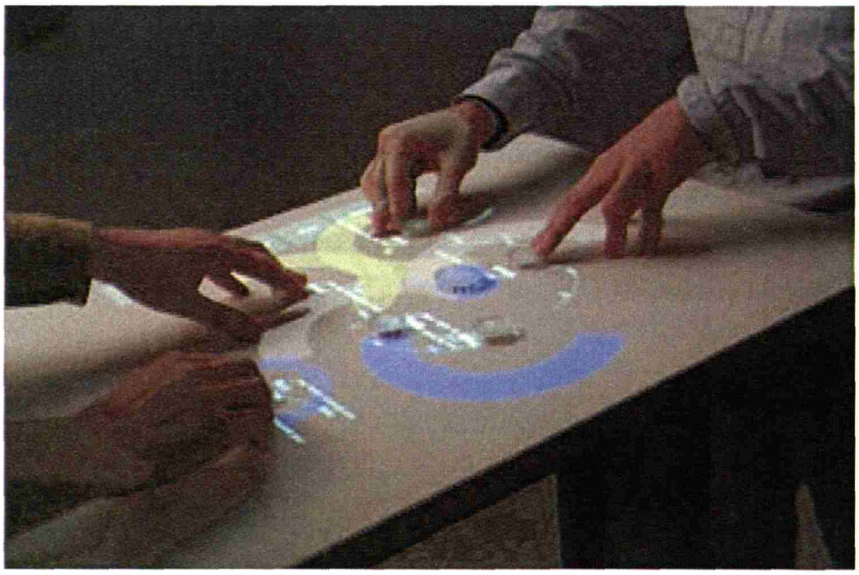


Figure 1: An electronic music performance application called Audiopad. Audiopad was one of the applications used to develop and explore the techniques described in this section.

The electronic music performance application, known as Audiopad [Patten 2002], is designed to combine the modularity of software synthesizers with the ability to expressively control multiple musical synthesis parameters at the same time. Another one of the project's aims is to interact with electronic music in a tactile and visual manner, such that the audience in a performance can see how the music is actually being created, rather than just watching the performer interact with a laptop using a keyboard and mouse on stage. I was excited about exploring new TUI interaction techniques in the context of a musical application for two reasons. First, musical performance is a very demanding application from an interface perspective, particularly as far as timing is concerned. The quality of a performance depends in part on the ease of interaction with the interface. Second, musical applications often involve the manipulation of many different parameters, both continuous and discrete, so there were many opportunities to explore interaction techniques for setting these parameters. During the process of its development Audiopad has been used in more than ten public musical performances and three museum installations. During this process I have observed users with a variety of musical and computer skill levels interacting with it.

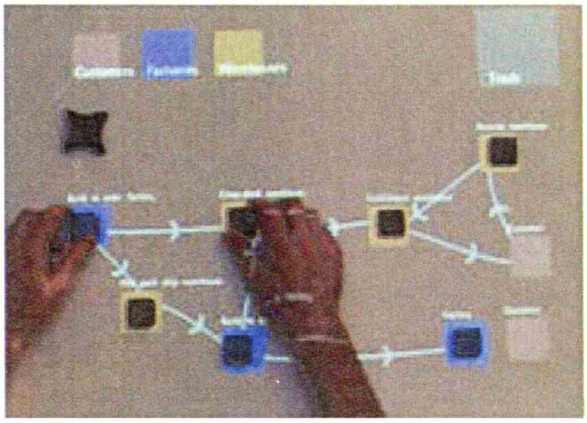


Figure 2: Creating a simulation model of a business supply chain in the SCVis application.

The business supply chain application, known as SCVis[1], allows users to create and simulate models of how businesses work, using a method known as system dynamics simulation. The SCVis system uses a database called the Process Handbook [Malone 1999] to store a taxonomy of business processes, which one can browse to retrieve, edit and analyze existing simulation models of other businesses. One notable aspect of this application from the perspective of interaction design is the need to quickly and easily navigate and edit a complex hierarchical data structure. In addition, this application involves a variety of discrete and continuous parameters that the user must be able to modify. During the development of the SCVis application, people ranging from simulation experts to business managers used the system to visualize and analyze hypothetical supply chain problems.

Most of the techniques described here employ a relatively generic set of tracked objects on the table, or pucks. Usually there are between five and eight pucks that represent data, such as the different tracks in a musical composition, or factories and warehouses in a business. In addition, there is one modifier puck that is used to change the properties of other objects. This modifier puck always has a star shape as shown at the top-right corner of figure 2.

Hierarchical item browsing and selection

In a graphical user interface, pie menus [Hopkins 1987] are useful for selecting items from sets of choices. We have explored a variety of related approaches for use in tabletop tangible interfaces for modifying the properties of pucks. The most common of these approaches is a two-handed, asymmetric approach in which the user's non-dominant hand holds the puck to be modified, and the dominant hand holds the modifier puck. This approach is based on Guiard's Kinematic Chain Model [Guiard 1987], which suggests that in asymmetric two-handed tasks, one's dominant hand acts in the frame of reference provided by the non-dominant hand. For example, when writing with a pen on a piece of

[1] SCVis was a collaboration between Mary-Murphy Hoye of Intel, Tom Malone and his research group at MIT's Center for Coordination Science, Jim Hines and his students at MIT's Sloan School of Management, and Hiroshi Ishii and myself at the MIT Media Lab.

paper, right-handed people often orient the paper with their left hand, and this improves their performance in the writing task [Guiard 1987]. Figure 3 shows the two-handed technique in use.

In Audiopad this approach is used to select a musical sample from a set of samples. The samples are arranged into various groups, and those groups may be collected into larger groups and so on. When the user places the modifier puck close to an area marked with a small '+', known as a hotspot, near the puck to be modified, the first level of choices spring out of the modifier puck. When the user moves the modifier puck over one of these items, any of its child items spring out, and so on, as shown in figure 3. A terminal node in this tree contains a colored square. Selecting one of these nodes by placing the modifier puck on top of it indicates the selection process is finished, and the tree disappears. In the case of Audiopad, these terminal nodes represent the actual musical samples, and selecting them causes a new sample to start playing. If the user wishes to cancel the selection of a new item from the tree, he or she can move the modifier puck away from the tree and the tree will disappear after a couple of seconds.

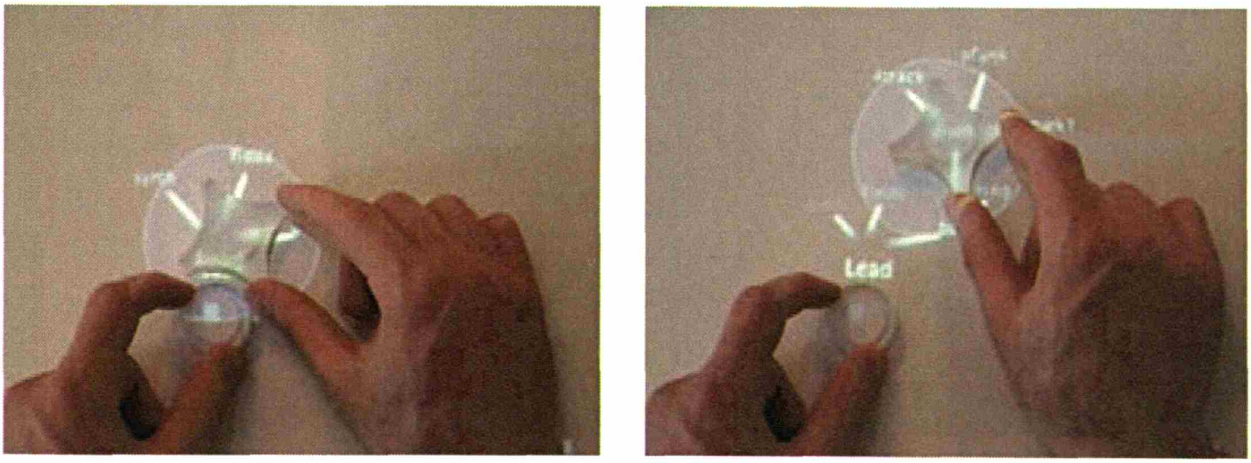


Figure 3: A two handed method for selecting items from a hierarchical menu.

As this technique depends only on the relative positions of the modifier puck and the object it is modifying, one can also select items from the tree using one hand. This hand can move either puck alone to select items. In informal demonstrations or museum installation settings users almost always select items using one hand on the modifier puck, while in performance contexts, performers typically use both hands, though sometimes use only one hand when the other is occupied with another task. I believe this difference is due to the stricter timing requirements in the performance context, as well as the performers being more familiar with the interface.

One problem with the first version of this interaction technique was that the selection process gave no feedback about recently selected items. In the context of Audiopad, similar sounding samples are located near each other in the selection tree. During a performance one often wants to focus on a certain group of samples for awhile, and then move to another group. Without feedback from the interface about which items had been used recently, performers wasted time repeatedly searching for certain samples within the tree. To address this issue, I changed the interaction such that the location of the most recently selected item is displayed when the tree is first activated. While this greatly reduces the time spent searching for an item, it is still difficult to switch quickly between items that are located several levels deep in the tree because the user must repeatedly move the modifier puck between the hotspot and the item to be selected. For cases in which quick selection among a few items is needed, I developed a separate technique called floating menus which is discussed in the next section. A condition in which two handed interaction becomes important is when the tree extends off of the table while selecting an item several levels deep. In these cases, the user can simply move the base of the menu using the non-dominant hand to bring the entire tree

onto the table. Two user interfaces of note which have employed asymmetric two handed interaction are Toolglass[Bier 1993] and GraspDraw[Fitzmaurice 1996]. With the GUI-based Toolglass, one hand controls the mouse cursor, while the other hand positions a set of tools in the workspace. The GraspDraw system uses two 6 degree-of-freedom trackers as a method of physically interacting with a drawing application. In his thesis, Fitzmaurice states that he originally focused on using asymmetric gestures to create objects such as circles. [Fitzmaurice 1996] He notes that the hands obscured portions of the circles, and thus any benefit achieved through the asymmetric use of hands was overcome by not being able to see the results of the interaction.[Fitzmaurice 1996]. I did not observe users having problems with occlusion of graphics during the selection of items using the tree, probably due to two differences between these applications and GraspDraw. First, GraspDraw, running on

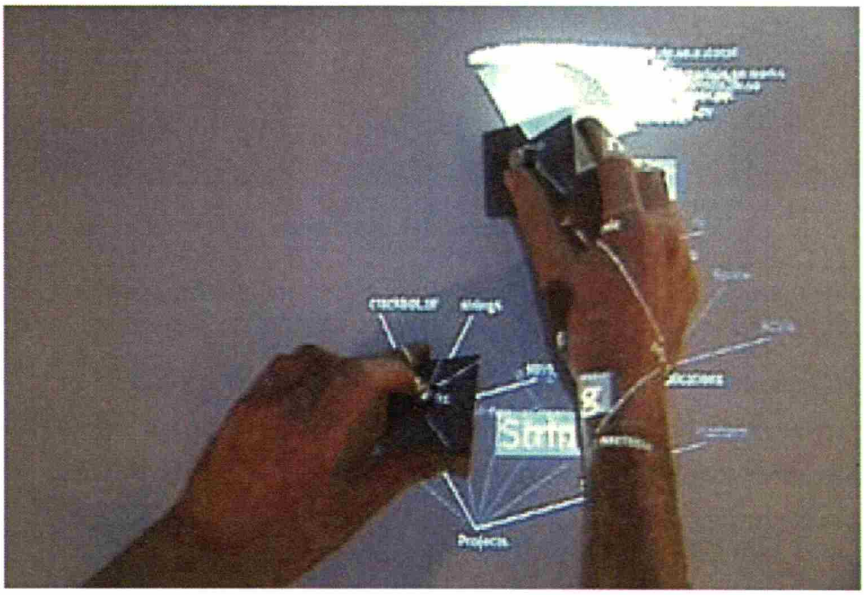


Figure 4: Navigation of a hierarchical filesystem. Each puck has a button which can be held down to drag the tree across the table.

the ActiveDesk[Fitzmaurice], uses rear-projection. The Audiopad and SCVis projects rely on projection from above. If the user places his or her hand on the table on top of some graphical information, the information shows up on top of the hand, though it will be somewhat distorted. Second, while navigating the tree the most important graphical elements, the children of the current node in the tree, are displayed in front of the modifier puck where they will not be occluded by the puck or the user's hand. To further avoid occlusion, each series of choices in the tree is displayed within 120 degrees of arc in front of the modifier, rather than completely surrounding it.

Navigating deeper trees of data

The technique described above works well for trees that are up to four levels deep. Beyond this point, it becomes cumbersome to use, because the interaction with the two pucks may take up a large amount of table area, and may be difficult to perform without accidentally bumping other pucks on the table. A clutching mechanism works better in cases where a very deep tree is needed, for example in the navigation of the taxonomy of business processes in the SCVis application, or when navigating a hierarchical file system.

This technique can be used with one or two hands. Each object used to navigate the tree has a button on top. When the button is not pressed, one can move the puck around the tree to select or browse information about various nodes in the tree. One can move the entire tree by holding down the button on the puck while moving it. In this way, one can navigate a large tree by clutching and unclutching the puck. When used with two pucks (one in each hand), one hand can move the tree while the other selects items from within it. With two pucks this technique can also be used to edit the arrangement of items in the tree. By holding down the

buttons on both pucks at the same time, one can 'break off' a part of the tree, and reattach it somewhere else. This technique might be useful to rearrange a collection of digital photos, for example.

One limitation of this tree browsing technique is that the display can become unreadable when a node in the tree has many children. One approach I have explored for this problem is 'fanning' the tree out to use more space when the user is looking at a node with many children. This works well for up to about 20 chil-

Figure 5: Flipping the puck on its side to see a different representation of it's digital data.

dren. For larger collections of data one might consider using magic lens[Bier 1993] based approaches, or more sophisticated tree display techniques.

Trees with different types of data

A related challenge is navigating trees where each node has several types of subnodes. I developed a technique for navigating these trees when working on a business process browser for SCVis, in which each business process may have three types of child nodes:

- instances - Companies where the process is put into place.
- subclasses - Variations on the process.
- navigation nodes - Connections similar to hyperlinks.

Depending on the task at hand the user is typically only interested in one of these types at a time. Other examples where an approach like this is relevant are a file browser, where one might want to occasionally see hidden files, or an object browser in a software development environment, where one might be interested only in specific kinds of items such as object methods, instances and subclasses.

Our approach to navigating this sort of dataset involves a puck in the shape of a cube or triangle that can be flipped onto its various sides. Each side can represent a different type of data to be displayed, while one side has a button to perform the clutching operations described above. When the user flips the puck, the graphical representation of the tree flips as well. The items that were being displayed disappear, and new ones appear in place, as shown in figure 5.

With the first version of this interaction technique, users understood the notion of flipping the puck to see a different type of data, but they rarely used the technique when interacting with the SCVis system. While each node in this system technically could have any number of three different types of children, in practice most interaction focused on the 'subclass' type. At first users often tried flipping the puck several times, for example to look for 'navigation' nodes, only to find none. After this initial exploration users often stopped flipping the puck because it was tedious to repeatedly do this during an otherwise rapid process of navigating the tree. A possible solution is to add graphical feedback around the puck to

indicate when the puck could be flipped to reveal extra nodes of a different type. This feedback might consist of a small colored area to one side of the puck. The side would indicate which way the puck should be flipped to reveal the nodes, the color might indicate the type of nodes, and the size could represent an estimate of the number of nodes.

Two notable examples of flipping physical objects in the context of a user interface are the Flipbrick [Fitzmaurice 1995] and the Toolstone [Rekimoto 2000]. Both objects are cordless devices with six faces tracked by Wacom tablets. The Flipbrick can be used in a graspable interface to select items from a menu. The Toolstone is designed for use with a GUI in a user's non-dominant hand. The position of the Toolstone can switch between tools that are used by the dominant hand, or modify actions performed by it.

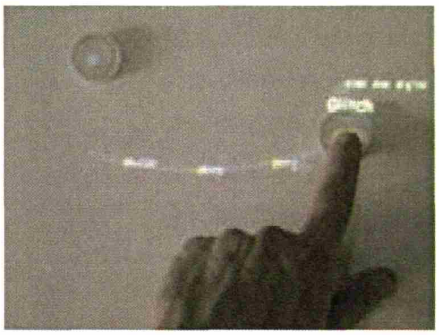
The technique using the Sensetable described above is used to see a different view of data, rather than to select actions, as with Flipbrick and Toolstone. Just as one might flip an everyday physical object to see it from a different perspective, it makes sense to flip a physical representation of computational data to see a different perspective on the data itself. Software reinforces this metaphor by graphically flipping the data in the same way the user flips the physical puck.

Floating menus

As discussed above, users of Audiopad in performance found it tedious to repeatedly select samples from several levels deep within the sample tree. To address this issue, I developed a floating menu that can follow objects around as they move on the table. The menu is shown in figure 6. To select an item from the menu, one simply moves the object on top of the desired selection. In the context of Audiopad, these menu items represent audio samples that are related to the sample currently being played. As the user moves the object around the table, the menu follows it, so that the user can easily select something from the menu with a quick gesture.

The important design issue in this interaction is when the menu should move, and when it should be stationary. If the menu moves too much, it can be difficult to select something from it, while if it moves too little, it will usually be far from the object it corresponds to. To determine when the menu should move and when it should be still, I define an area surrounding the icons called the selection area, as shown in figure 7. When the puck is inside of this area, the menu stays still to make selection easier. If the puck moves outside of this area for more than 3 seconds, the menu recenters around the puck, such that the currently selected choice from the menu is underneath the puck. When the puck moves, the menu lags behind it slightly. This gives the user freedom of movement in case he or she would like to move a puck to a specific area on the table without accidentally selecting an item from the menu. In the original version of this technique, the menu would move toward the puck whenever the puck left the selection area surrounding the icons. This approach sometimes caused problems, because a user would accidentally move the puck outside of this area while trying to select a menu item.

The floating menu presents a list of choices to the user.



The user selects one by moving the puck along the arc.

The user can move the puck anywhere else once the desired item is selected.

After a brief time delay, the floating menu moves back under the puck.

Figure 6: Floating menus in the Audiopad application. One selects an item from the menu by placing the puck on top of it. When the user moves the puck away from the menu, the menu follows it.

This would cause the desired menu item to move, making it difficult to select. I experimented with increasing the size of the selection area to make menu selection easier, but this caused the menus not to follow the pucks when users thought they should because the puck was still inside of the selection zone. The time-based approach works well because users can stray outside of the selection zone when moving the puck toward an item in the menu without having the menu move in response. This time-based tolerance means that the selection zone around the icons can be small, ensuring that the menu will follow the puck as the user moves the puck around on the table.

Another issue with the design of this technique was how the menu should recenter around the puck. In the initial design, the menu recentered by moving toward the puck until the puck was once again in the selection zone. This approach occasionally led to items in the menu being inadvertently selected after the menu had recentered itself several times. Recentering the menu by moving the currently selected menu item underneath the puck resolves this problem.

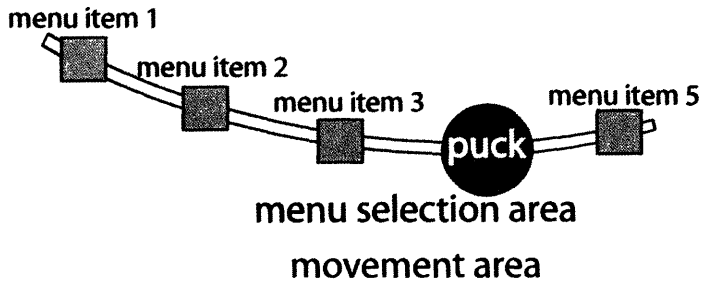


Figure 7: The selection area around a floating menu. When the puck is in this area, the menu will not move.

Changing Continuous Parameters

Many applications involve the manipulation of continuous parameters. For example in Audiopad, each audio track has a volume parameter. One early approach to this problem was to rotate pucks on the table to change their volume. Graphical feedback, in the form of an arrow and a bar graph were displayed beside the puck to indicate the current setting. I avoided using a physical dial as was used with the Sensetable system [Patten 2001] because this approach must deal with what Buxton calls the 'nulling problem' [Buxton 1986]: a condition resulting when the physical state of a dial and its computational state are inconsistent. There were several problems with this approach to parameter control. First was a tradeoff between precision and speed when adjusting a parameter. The software could be configured such that several revolutions of the puck were needed to fully traverse the range of possible parameters. In this case it was possible to set the puck to a value with several digits of precision, but it took a lot of rotating to reach a desired value. Alternatively, with the entire parameter space accessible in one revolution of the puck (or less), parameter changes could be made quickly but it was difficult to make them precisely. Another issue with this approach was that it was difficult to change multiple parameters at the same time. Any more than two parameters was essentially impossible with two hands.

A more subtle issue was that when a user would rotate a puck, their hand often obscured it from the view of others. This made it difficult for others to observe the manipulation being performed and understand its effect in the context of the application. In the context of Audiopad, this was a concern because we wanted the audience of a musical performance to see the causal relationships between the performers actions and the music they were hearing.

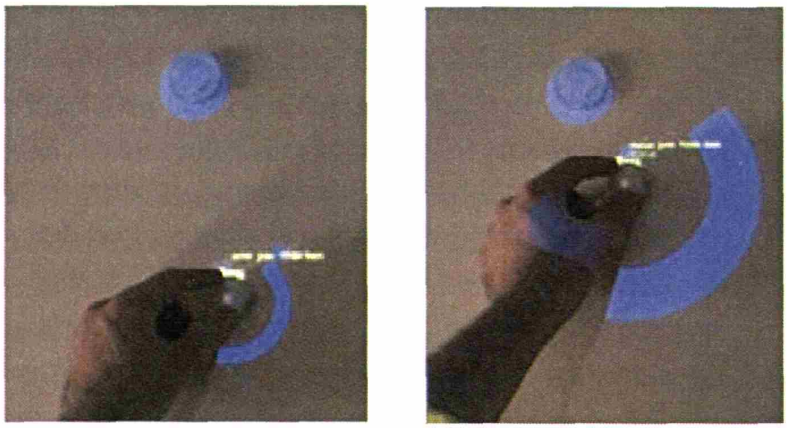


Figure 8: In Audiopad, the distance between the microphone (top puck) and an audio track (bottom puck) determines the current volume of that track. The size of the colored arc in the photos represents the current volume of the track it surrounds.

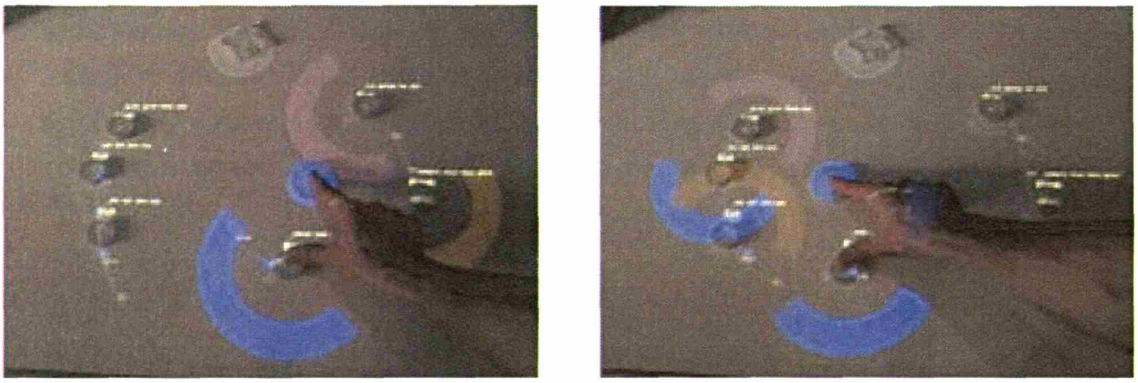


Figure 9: Using the microphone to adjust the volume of many tracks at one time, as one might do when transitioning between songs. The user is moving one audio track with his thumb to keep its volume constant while he adjusts the volume of the other tracks. The blue circle underneath the user's index finger is the microphone.

I believe this difficulty in seeing causal relationships could be of concern in face-to-face collaborative applications as well, where the TUI becomes a shared medium for expressing ideas.

Based on these observations I developed a technique that allows one to manipulate multiple parameters simultaneously with coarse motor movements. The value of the parameter is determined by the distance between the puck and another master puck. In the Audiopad application, this technique is used to control the volume of all of the tracks. The distance between each track and a special puck, called the microphone, determines all of the volumes: tracks that are closer to the microphone are louder than those that are farther away. To change the volume of a particular track, one simply moves it closer or farther away from the microphone as shown in figure 8. One can grab several tracks with each hand and move them simultaneously, or move the microphone itself to change the volume of all tracks together. If the user wants to change the volume of most tracks while leaving a few of the volumes constant, he or she can move the microphone with one hand, while moving the other tracks with the other hand, so as to maintain a constant distance between them and the microphone. (figure 9)

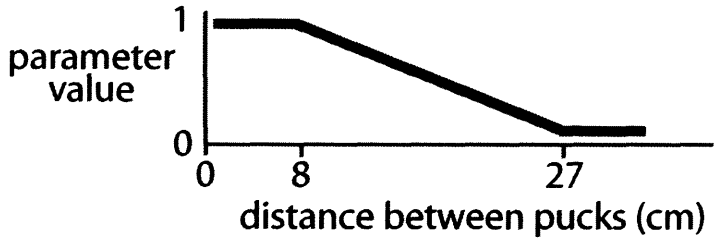


Figure 10: The volume parameter as a function of distance from the microphone puck.

One detail important for making this technique work well is the function mapping distance to the parameter being controlled. After some experimentation in the context of Audiopad we arrived at a transfer function shown in figure 10. Within the range of 8 cm. of the microphone, the volume is at its maximum level. From this point the volume decreases linearly until a distance of 27 cm, where the volume reaches zero. This mapping means that there are always some areas of the table where the movement of a puck has no effect on it's volume. For parameters that are changed infrequently, one might want to use a mapping in which the active area was smaller. This would give the user maximum flexibility in how objects in the rest of the space were organized.

Arcs

Another method for changing one-dimensional continuous parameters is with the use of a "parameter arc" around the corresponding puck, as shown in figure 11. To change this parameter, one simply moves the modifier puck over the parameter, and moves it in the desired direction. The arc is constrained to only move along a circle of fixed radius with the puck at its center, so deviating too far from this path with the modifier puck will cause the parameter to stop changing. So, when one wants to stop changing a particular parameter one simply slides the puck away, perpendicular to the arc. This method can be used with multiple parameters per puck, each rotating around a circle of a different diameter with the puck at the center.

Setting two dimensional parameters

While the technique above works well for controlling a one-dimensional parameter such as volume, there is no clear way to apply it to a two dimensional continuous parameter. In the context of Audiopad, we explored two techniques for modifying two dimen-

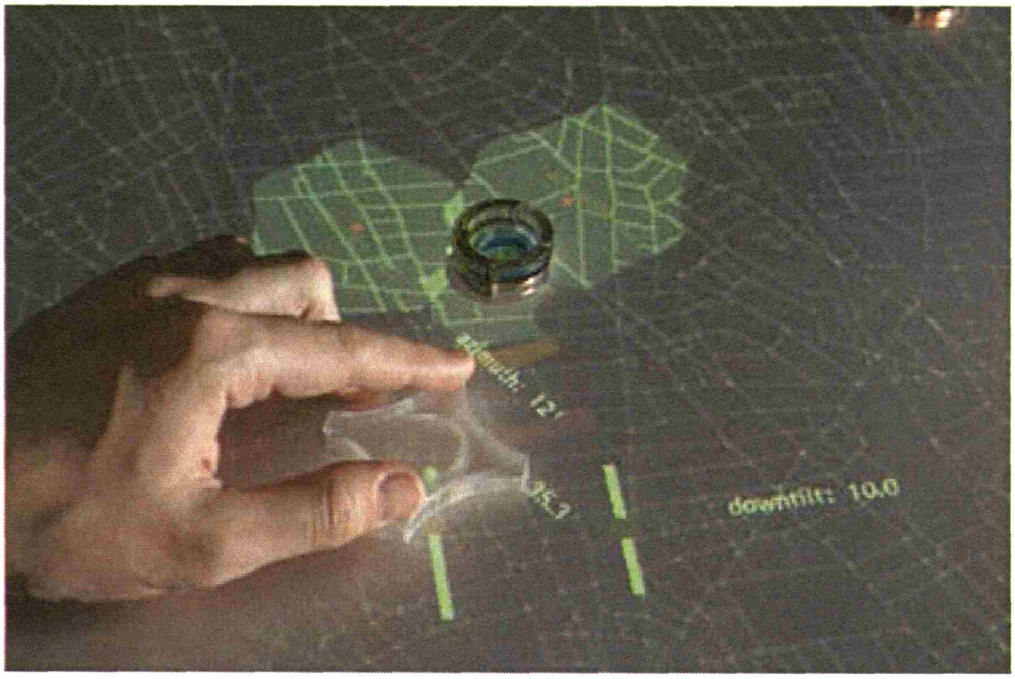


Figure 11: Using the selector puck to modify a parameter arc in the cellphone tower layout application.

sional continuous parameters. Users employed these techniques to change digital effect parameters on a track-by-track basis, for example the high frequency and low frequency cutoff of an audio filter.

The first technique was the use of effect "zones" on the table where the two dimensional motion of the puck controlled the two parameters. In this case, the absolute position of a puck in the effect zone determined the value of the parameter. Figure 12 shows a picture of this technique. With this approach, the direct mapping of a particular point on the table to a particular setting of effects parameters reduced flexibility in terms of where pucks could be on

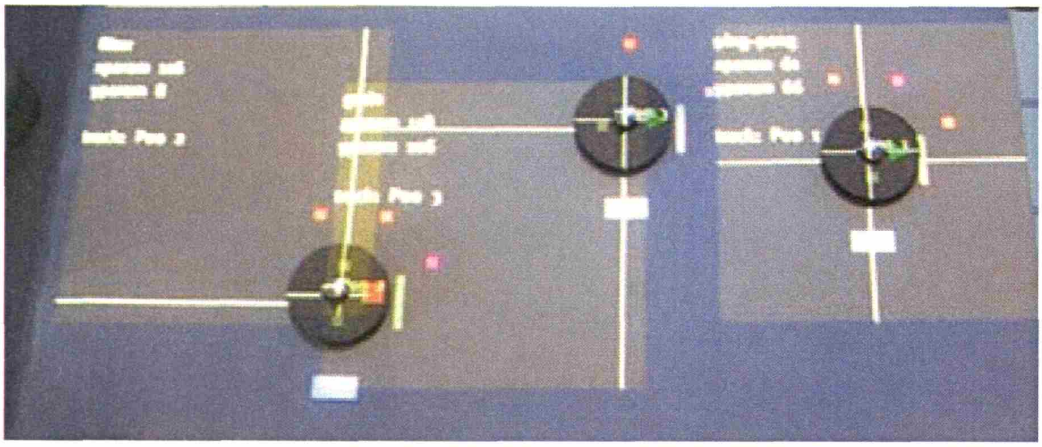


Figure 12: An early method of controlling audio effect parameters in Audiopad using "effect zones" where effect settings corresponded to absolute positions on the table.

the table. This rigidity made it difficult for users to arrange objects in other ways, for example to line tracks up in a row according to the order in which they were to be played. Second, this interaction technique did not give feedback about how the parameters were changed over time. If a musician wanted to gradually change a parameter a certain amount, the interface made it difficult to know when that change was complete. To address these issues we explored a technique for making relative adjustments to two-dimensional parameters. The user places the modifier puck on a hotspot toward the bottom of the puck. Then, the two-dimensional motions of the modifier puck control the two-dimensional parameter setting. Graphical feedback shows how the setting has changed since the modification started, as well as the current absolute setting of the parameter, as shown in figure 13. One can move either puck to change the parameter. What matters is their relative position. If the parameter has a bounded range of possible input values, the graphical feedback indicates this as shown in right picture of figure 13. The colored area stops following the modifier puck, and

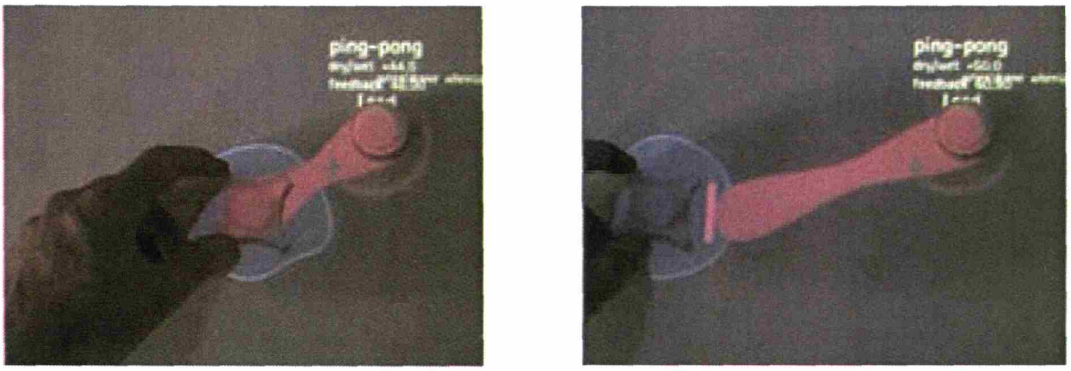
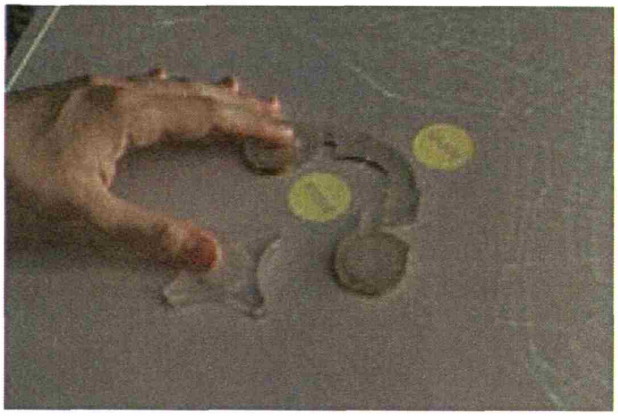


Figure 13: Using the modifier puck to change the effect parameters of an audio track. Here the effect setting is determined by the relative position of the two pucks. In the right picture, the user has exceed the bounds of the parameter, so the red colored area stops following the modifier puck.

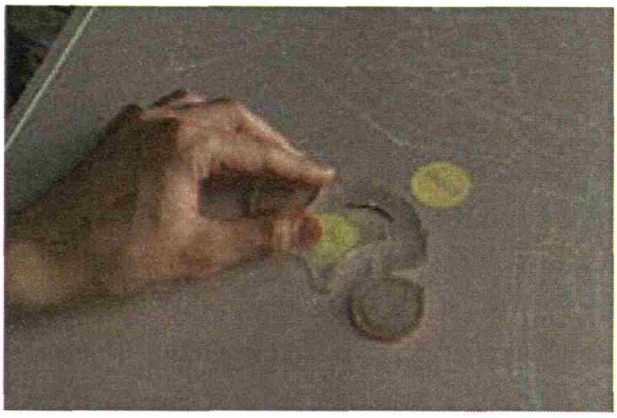
remains at the edge of the area of valid input. A red line indicates that the modifier puck has moved past the limit of the parameter setting. Once the user has set the parameter to the desired value, he or she can lift the modifier puck off of the table to deactivate the parameter modification mode.

Navigation

A common requirement in spatial applications is the ability to navigate within a spatial reference frame. For this task, I use two pucks, without buttons or other functionality, together to pan and zoom a map. The two objects are shown in figure 14. The first is the same star-shaped "modifier" puck used in other contexts, while the other has two slots in which the modifier can fit. One of these slots is for panning the map, while the other is for zooming. When one wants to move the map, one slides these two pucks together, as in figure 14, "pinching" the map with the two objects. As long as these two objects are touching in this configuration, the map will move along with the pucks as they are moved together across the table. This action can be quickly and easily repeated with a single hand to travel a larger distance. Alternatively, one can zoom out, move to the desired map location, and then zoom back in.



The selector puck and navigator puck are used to perform the move.



When the user brings them together as shown to the left, they "grab" the position on the map directly underneath.

The user can then move the map freely around the interaction surface, and spread the pucks apart when the desired position is achieved. Larger movements can be performed by repeating this gesture, or zooming out before moving. (see next page)

Figure 14: Moving a map using a pinching gesture

Zooming the map is also performed with the navigator puck and the selector puck.

The user brings the two pucks together so the star-shaped selector puck is touching the convex side of the navigator puck. This causes the zoom bar to appear.

The user moves the selector puck to the handle at the end of the zoom bar to change the zoom.

The user can now move the selector puck along the zoom bar to stretch or shrink the map. Once the desired zoom has been achieved, one simply moves the selector puck off of the zoom bar, and it disappears.

Figure 15: Zooming the map using a stretching gesture.

Zooming is performed using a slot on the other side of the navigation puck. Touching the two pucks together, as shown in figure 15, enables a "zoom bar" that one can slide closer to the navigation puck to zoom out, or farther away from the navigation puck to zoom in. During the zooming process, the map is constantly rescaled so that points on the map that are underneath one of the pucks remain so until the zooming operation is complete. Here the metaphor is that one is "stretching" the map with both hands. A similar metaphor was used in the "Metadesk" [Ullmer 1997] project by Hiroshi Ishii and Brygg Ullmer. However the Metadesk did not include the clutching and mode-switching functionality described here.

Design Principles

The lessons learned while creating and testing applications in musical performance and business simulation suggest three design principles for use with tabletop tangible interfaces.

Interactions should be legible for observers

An important issue to consider in the design of these systems is the legibility of the interaction from the perspective of an observer. I first observed the importance of this principle when testing the Audiopad in a performance situation. The initial iteration of the system used the rotation of objects on the table to control the volumes of individual tracks. One of the limitations of this approach was that observers could not easily tell that a performer was rotating an object on the table because the performer's hand usually obscured the object. One of the reasons that linear movements of the objects on the table worked better for changing parameters was that audience members could see them more easily. They could observe the correlation in time between certain motions on the table, and corresponding changes in the sound produced by Audiopad, and thus begin to understand what the performers were doing.

The idea of legibility of interaction from the perspective of an observer is relevant for systems involving collocated collaboration as well. For example, in case where multiple users are interacting with a simulation, such as a business supply chain or computer network simulation, this work suggests that observers would more quickly understand the causal relationships present in the simulation if rotating gestures to change simulation parameters were replaced with linear movements of pucks on the table.

Show possible interactions between objects

The use of a mouse in a GUI is typically based on a one-to-many relationship of tool (mouse pointer) to targets (buttons, menus, sliders etc.) In contrast, a tabletop TUI usually has a many-to-many relationship of pucks to targets (other pucks, graphical hotspots on the table). This many-to-many relationship means that an interface must convey more information about what interactions are valid between pucks. One approach to this need is using animation when there is a possible interaction between two pucks that are close to each other, as shown in figures 16 and 17.

Relative versus absolute mappings

Two possibilities for setting continuous parameters in a tabletop tangible interface are to use a relative mapping based on the positions of other pucks, or an absolute mapping based on a puck's position on the table itself. In application domains such as urban planning [15], an obvious mapping exists between the positions of buildings on the table and hypothetical buildings in the real world. In these types of applications, a direct spatial mapping of physical objects in the interface to a hypothetical urban site makes it easy for a user to understand and participate in the interaction. However, many applications have no such obvious direct spatial mapping. Using spatial mappings based on objects' positions relative to each

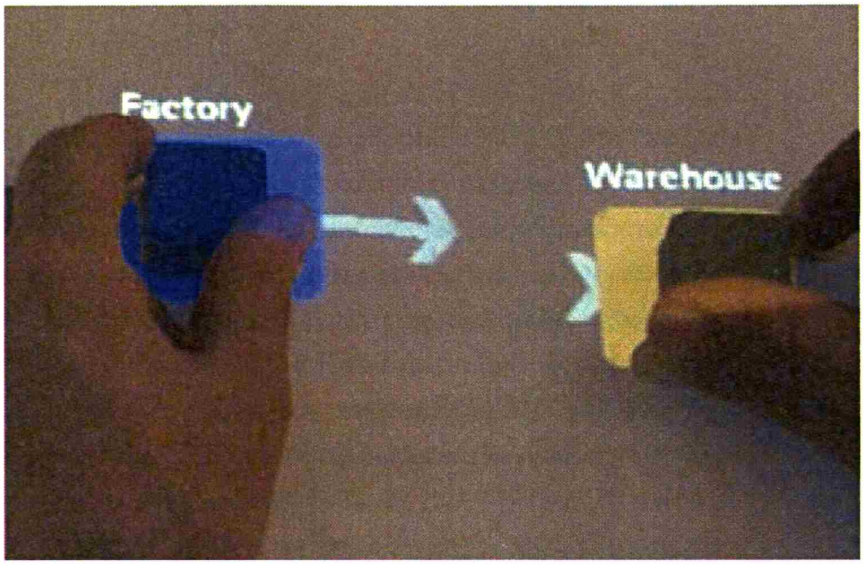


Figure 16: Arrows from two pucks on the table point to each other, indicating that these objects can be connected.

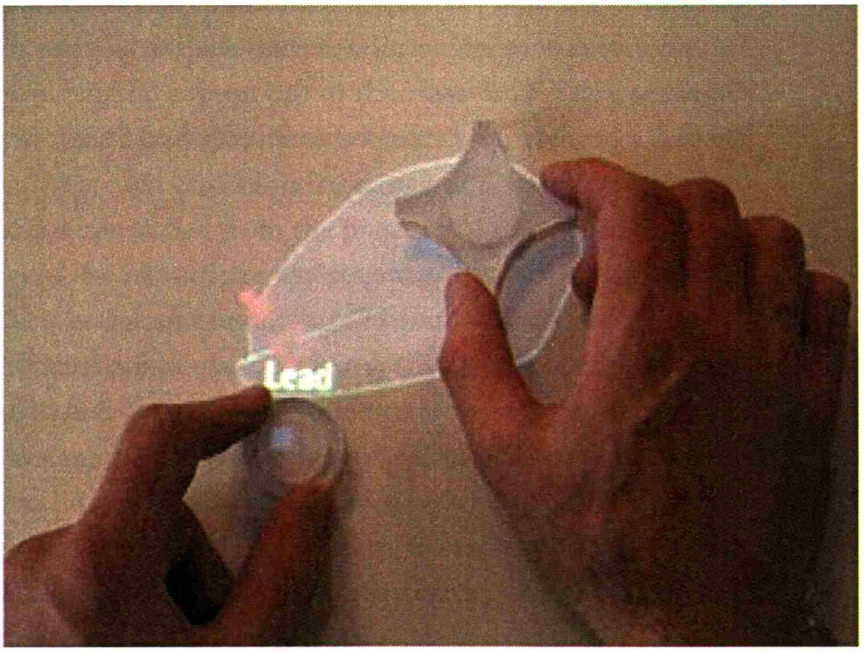


Figure 17: The grey circle underneath the star-shaped modifier puck points to the hotspot above the puck in the user's left hand, indicating that this object can be modified.

other seems to work better in these cases. For example, the technique shown in figure 13 worked better than that shown in figure 12. Research by Kirsh [Kirsh 1995] illustrates a variety of ways that people can use physical objects to offload computation from their brains to their environments. The use of absolute mappings in a tabletop TUI can prevent the user from moving the pucks on the table to employ these types of techniques. Relative mappings can also better afford multiuser collaboration, as users standing around the tabletop interaction surface can define their own reference frame within the context of their body by orienting their pucks appropriately. For many applications, it seems better to leave some degrees of freedom open to interpretation by the user.

Interaction Techniques for Actuated Interfaces

While the interaction techniques described above are applicable to tabletop interfaces with actuation, there is another group of techniques that are specifically designed for interfaces incorporating actuation. One of the primary goals of incorporating actuation in Pico is to make pucks in the interface behave more like objects in the everyday world, so that many of the "interaction techniques" we use with everyday things (such as putting a paperweight on top of a stack of papers to keep any of them from moving) can have an intuitive and easily discoverable analog in the interface.

The pucks used with Pico can be bound to the positions of objects projected on the interaction surface, and positions of these graphical objects in turn represent some parameters in what is at its most abstract level a complicated math problem. By translating the mathematical "forces" on these parameters into physical forces, the system creates the illusion that the parameters are attracted to a better solution, as if by gravity. Other interactions happen in the context of this constant force that tries to pull the complex system of variables toward the best solution it can find.

Within that context, users can guide, constrain and move objects (thus changing parameters) using a rich physical vocabulary. The richness of this physical vocabulary comes from the fact that of the assumptions we make about how everyday objects behave are still valid here. Here I present some interactions that are possible within that space.

Objects can be kept very close together using a rubber band, as shown in figure 18. Alternatively, if one prefers a larger maximum distance between objects, one can use a larger mechanism such as that shown in figure 19. A minimum distance constraint can be established using collar around one or more pucks, as shown in figure 20.

Figure 18: A rubber band used as a mechanical constraint to keep two pucks in close physical proximity.

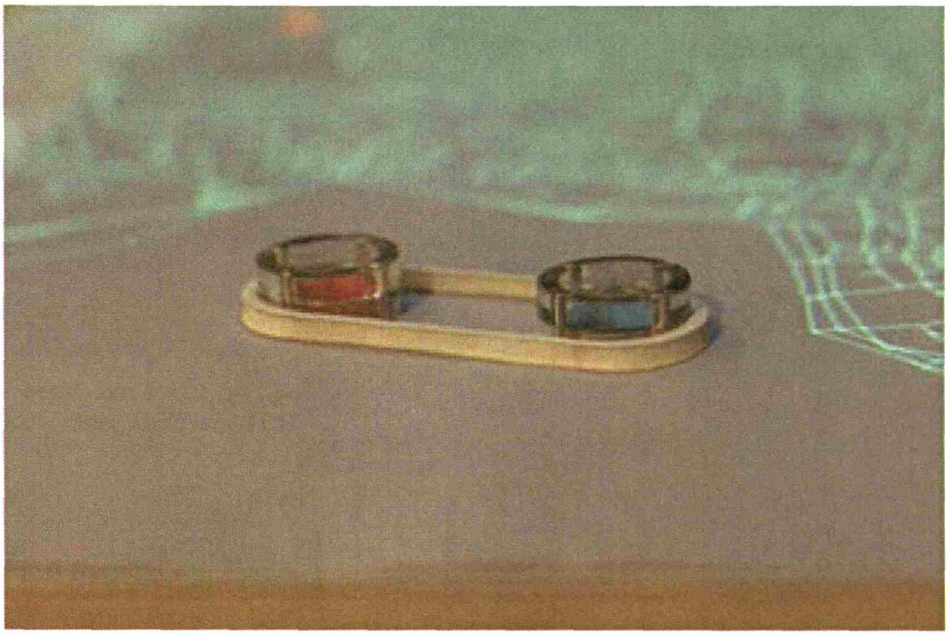


Figure 19: An oval-shaped ring used to keep two pucks within a given minimum distance to each other.

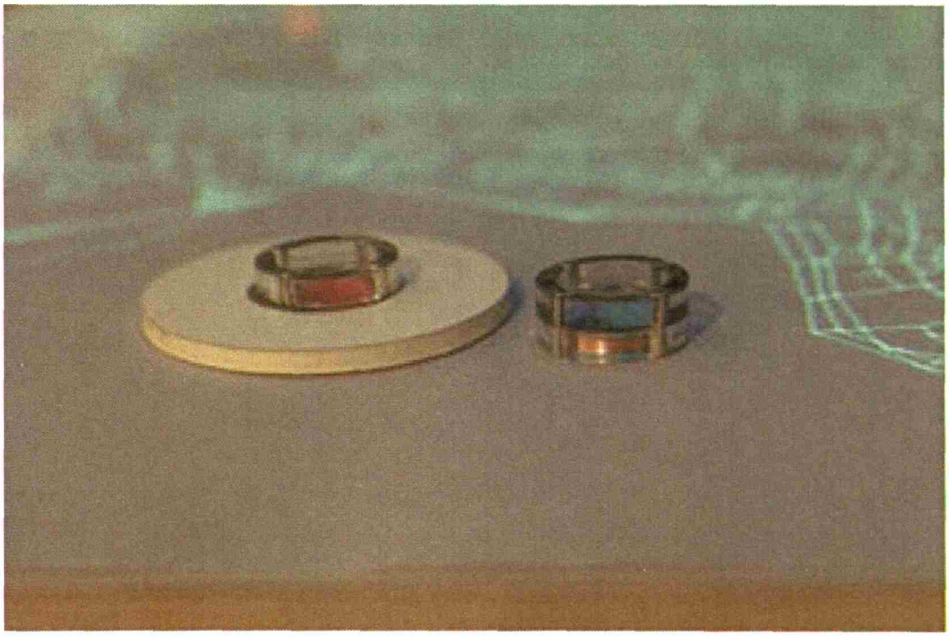


Figure 20: A collar used to enforce a minimum distance constraint.

Collars with different elevations can be used, so that different puck combinations are mechanically constrained to different minimum distances, as in figure 21. These distance constraints can also be combined to establish a minimum and maximum at the same time, shown in figure 22.

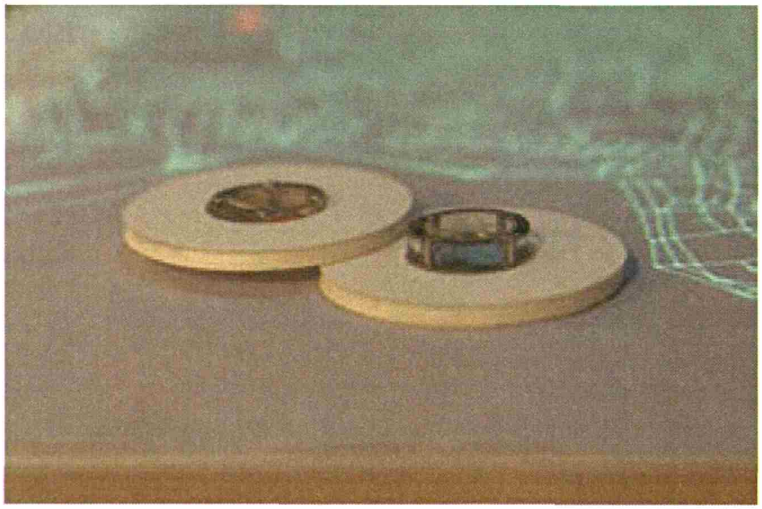


Figure 21. Collars at differing heights can yield different minimum distance constraints for different combinations of pucks.

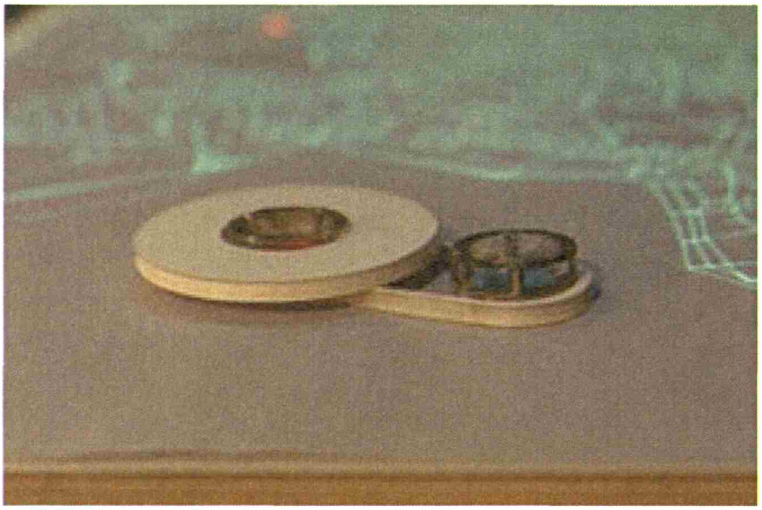


Figure 22: Minimum and maximum distance constraints can be combined.

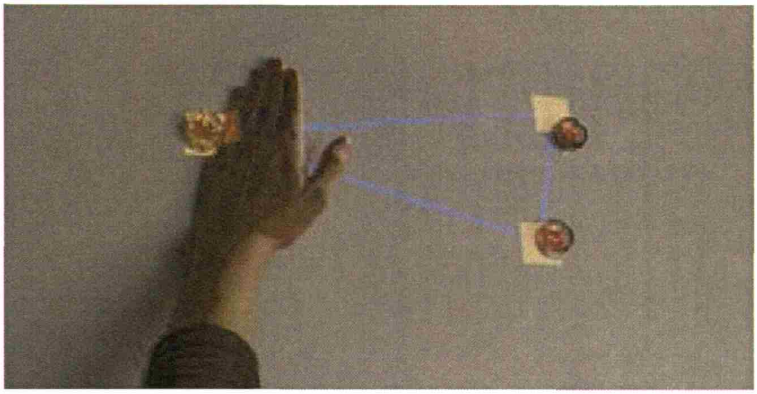


Figure 23: A temporary mechanical constraint of one puck's motion established by the user's hand.

Physical barriers can be used to constrain puck motion in a variety of ways. For example, if one wants to keep an object in its current position, one can simply hold it in place with one's hand (figure 23), or place some sort of weight on top of it (figure 24), or fix it in place with tape. To keep an object or objects inside of or outside of a given area, one can use a flexible curve such as the one shown in figure 25. One could place the pucks on small pads with different types of bottom materials, such as Teflon or sandpaper, to make it easier to move some parameters than others, changing the "weight" of these parameters within the mathematical optimization.

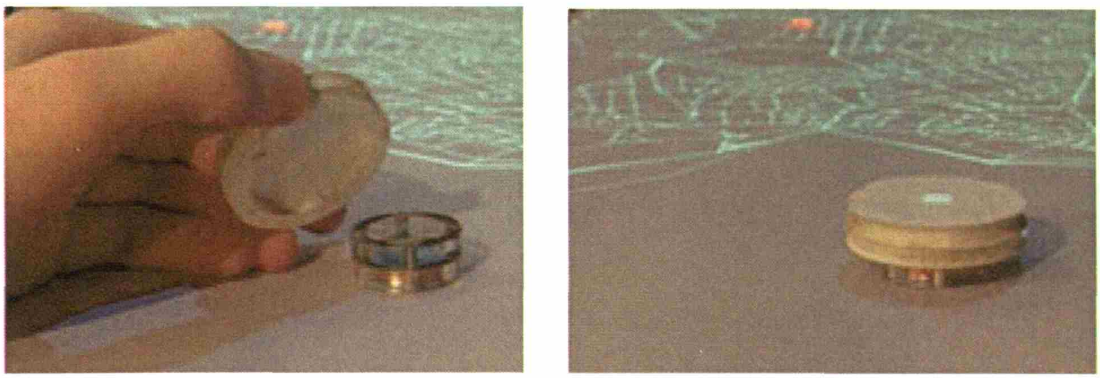


Figure 24: The object placed on top of the puck is filled with sand, preventing the computer from moving the puck autonomously.

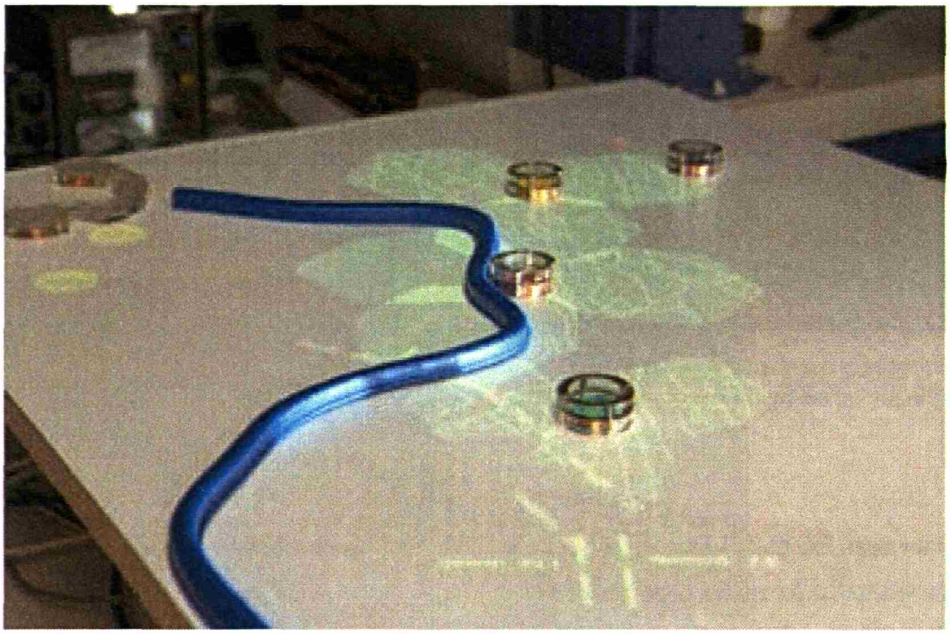


Figure 25: A barrier limiting the motion of pucks on the table.

While the set of constraints above illustrate some of what is possible with mechanical constraints on actuated surfaces, it is not intended as an exhaustive list. One of the goals of this work is that users will be able to improvise new mechanical constraints to meet their needs, because these constraints build on users' existing knowledge of the world. Because users can see and understand the causal relationships between the pucks and constraints on the table, a constraint need not perfectly describe the desired computational behavior perfectly, because users can easily change or override them if necessary. They serve as short term, ad hoc "jigs" to make the problem solving process easier.

Sound feedback

The electromagnets used by Pico are mounted on strips of stainless steel, which moves slightly when the magnets are electrified as a result of the magnetic field. While I was building Pico, I calibrated the control loop on the Pico embedded microcontroller so that while the magnets were being driven, the vibration of the steel beams would occur in the audible range. This sound proved to be a useful debugging tool, in the same way that sounds from a car's engine can help a mechanic determine its health. Once the construction of Pico was complete, I placed a microphone near the electromagnets to amplify this same sound. This audio is processed using audio software called Pure Data [Puckette 2001] and played over speakers near the table. The audio provides an added channel of feedback about how the computer is responding to the user's actions. This can be helpful, for example, in cases where one is watching another user move objects, and the computer is resisting the movements. The audio can help onlookers understand at which points the computer is resisting the user's motions, without touching the table.

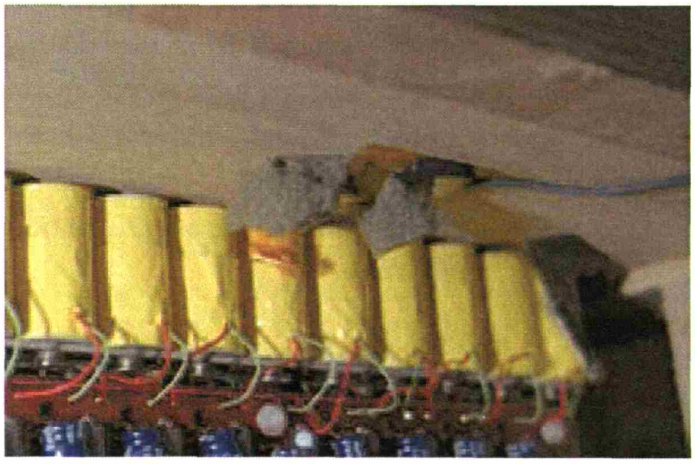


Figure 26: A contact microphone attached to the bottom of the interaction surface to detect the vibrations caused by moving pucks and magnets.

6 Evaluation

Pico differs from most graphical user interfaces in two fundamental ways: First, objects inside the software are represented and controlled by pucks on a tabletop, rather than with graphical icons and a pointing device. Second, Pico uses actuation to represent mathematical "forces" inside an optimization using mechanical forces on the table. To explore these differences I conducted a two-part experimental evaluation asking the following questions:

1 Are tabletop interfaces based on tracked physical objects any better than touch screens for object manipulation tasks?

2 Does actuation in a tabletop tangible interface like Pico help users solve complex problems?

Object Manipulation on Horizontal Interactive Surfaces

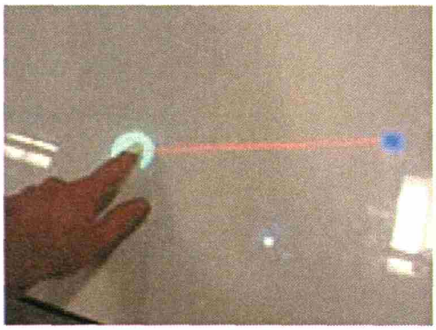
In this first experiment I compared performance times for a simple object manipulation task performed on a horizontal surface with video projection. In task A, users moved graphical objects with their hands on a touchscreen. In task B, users moved graphical objects on the tabletop using a tracked object held in the hand. In task C, users move four physical objects (one at a time) which are bound to projected graphical objects. While typical graphical and tangible interfaces differ in a variety of ways, in this experiment I chose to focus on the use of multiple physical controllers and compare that to the use of a single input device.

Task

Subjects were seated in front of a desk with a video projector mounted above it, pointing down onto the desk. Using two different types of sensing apparatus, subject performed three different tasks on the tabletop in front of them. The three tasks are described below.

Touchscreen task

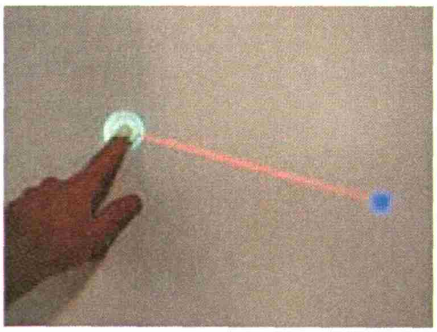
Subjects were presented with a colored circle, graphically projected on the tabletop. Subjects were asked to drag this circle using a touchscreen to another, smaller circle located at a random location on the tabletop. A graphically projected straight line connected the two points. As soon as this task was completed, the user would immediately be presented with another instance of the same task.



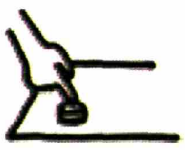
The touchscreen task: The user's finger is directly touching a glass sensing surface

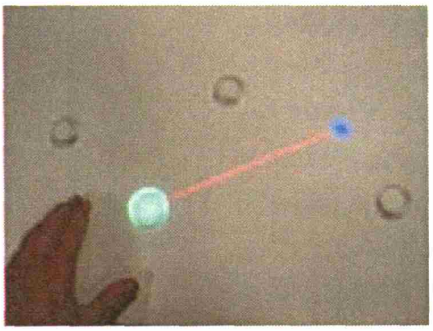
One puck task

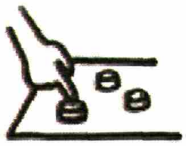
This task is similar to the touchscreen task, in that subjects were to drag a circle to designated a location on the table. However, instead of using a finger for this task, subjects were given a cylindrical object made of translucent acrylic measuring approximately 1 cm in height and 3 cm in diameter. The puck was embedded with an LC tag that allowed it to be tracked inductively using an antenna placed on the tabletop.



The one puck task: The user moves a Sensetable puck with his hand, using it as a pointing device







The four pucks task: The user moves one of four Sensetable pucks to the target

Four pucks task

Four pucks, identical to that used in the one puck task, were placed on a tracking surface in front of the subject. A randomly selected one of these pucks is then highlighted, and a line is drawn a location where it must be moved. As soon as the user accomplishes this, another puck is highlighted to perform another iteration of the task. The next puck highlighted is always different from the one the user just moved.

In each of these tasks, the user must first acquire an object with his or her hand, and then move it to a specified target. The difference between the experimental tasks lies in how the objects are acquired. In the touchscreen task, the user must touch the graphical object with his or her finger. While the user can feel when he or she touches the screen, the user must rely on visual confirmation, in the form of graphics displayed by the projector, to be sure that the object has been successfully acquired. In the One Puck task, the user still receives some useful tactile feedback from the object in his or her hand, but must rely on graphical feedback for confirmation that an object has been successfully acquired. In the Four Puck task, the graphical objects to be moved are permanently associated with physical objects on the table. In this case, the user can rely purely on tactile feedback from his or her hand to know that the object has been successfully acquired.

Experimental Hypotheses

H1 Subjects will complete the Four Pucks task faster than the other two.

The tactile feedback provided by the physical objects in the Four Pucks task should allow the user to begin moving the target toward its destination without waiting for visual confirmation that the object has been successfully acquired. In the other two cases, having to rely in graphical feedback will delay the user.

H2 The acquisition time in the Four Pucks task will be smaller than in the other two tasks

By acquisition time I mean the time between when the task is displayed to the user and when the user begins moving the target toward its destination. Even though the target diameter in all experimental conditions is the same, I expect the two-dimensionality and lack of haptic feedback during target acquisition will require more fine motor control in the One Puck and Touchscreen tasks than in the Four Pucks task, forcing the user to perform it more slowly.

H3 There will be no statistically significant difference between overall performance times for the Touchscreen task and the One puck task

In both of these tasks the user must rely on graphical feedback to know that an object has been successfully acquired. While there may be some difference in average times for these tasks, perhaps in part due to differences in friction between the RF tracking surface and the touchscreen, I do not expect the difference to be significant.

H4 If one compensates for acquisition time, there will be no statistically significant difference between the three tasks.

Once the object has been acquired, all three tasks are quite similar, so I do not expect to see significant differences in performance times when acquisition time is subtracted from total task performance time.

Subjects

Twelve subjects were recruited using flyers posted around the university campus. Each subject completed all three conditions of the experiment. The order in which subjects completed the three conditions was varied such that each of the six possible orderings was experienced by two subjects.

Experimental Procedure and Design

The One Puck and Four Puck tasks were performed with a custom built sensing system using a matrix of antenna elements to inductively track the position of analog LC tags embedded in the objects on top of it. The antenna was embedded in a wooden case and covered with a white, high pressure laminate sheet similar to what one might find on a kitchen countertop.

The Touchscreen task was performed using a NextWindow 2401 touchscreen. [NextWindow] This system tracks finger positions using two cameras located directly above a glass pane inside a metal frame. This system can only reliably track one point of contact at a time.

For each task, subjects were asked to perform a training block of 50 iterations of the task. They were told that this block was just for practice. After this block of 50 trials, subjects were asked if the understood the task, and any questions were answered. Then subjects were given another block of 50 tasks, and were asked to complete the group of tasks as quickly as possible. Subjects were given an oppor-

tunity to rest, and were then asked to complete another block of 50 tasks, which was also timed. The times from these final 100 trials for each task were analyzed. Raw sensor data from the touchscreen and antenna array were collected with custom software for further analysis.

Limitations

This experiment does not deal with cases where users move multiple objects to multiple targets at the same time. We omitted this case for lack of a touchscreen interface of sufficient size that could reliably track multiple points of contact from the same person.

Results

We performed a two factor ANOVA (task x subject) on the average task performance times and found that users were able to perform the four puck task faster than the one puck and Touchscreen tasks, as shown in figure 4. The difference between the four puck and one puck conditions was found to be significant ($p < 0.05$). The difference between the four puck and touchscreen conditions was also significant ($p < 0.05$). However, the difference between the touchscreen and one puck conditions was not found to be significant ($p = 0.59$).

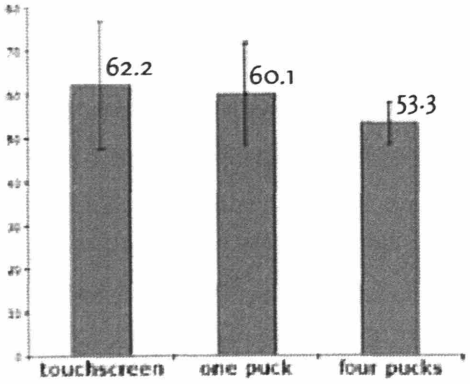


Figure 4: Task performance time in seconds for the three experimental conditions. Mean and standard deviation are shown.

By analyzing the data log generated by software as each condition was running, we determined the time delay between when the subject was presented with each task, and when he or she began moving the object to complete the task. We performed a one-way ANOVA on this data and found that subjects were able to acquire objects more quickly in the four pucks task than in the one puck task and the touchscreen task. The difference between the four and one puck tasks was significant ($p < 0.001$) as was the difference between the four pucks and touchscreen tasks ($p < 0.001$). Acquisition times for the one puck task were also significantly faster than for the touchscreen task ($p < 0.001$). This result was surprising, as both tasks require the user to rely on graphical feedback to know when an object has been successfully acquired.

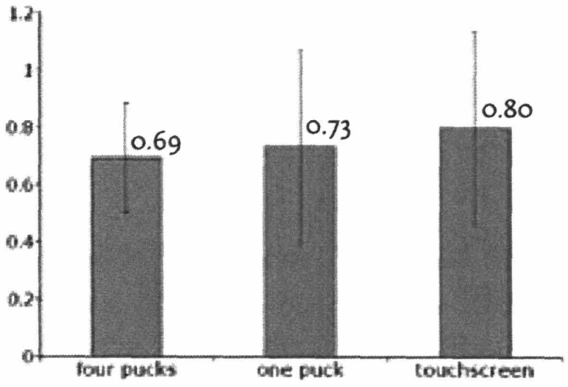


Figure 5: Object acquisition time in seconds for the three experimental conditions. Mean and standard deviation are shown.

Multiplying the average acquisition time in each experimental condition times the number of movement tasks per trial (50) reveals that faster acquisition of the objects in the four puck case only accounts for 4.5 seconds of the overall difference in performance time. In the interest of learning what was causing the rest of the performance difference, I analyzed the velocity data of the object in each condition during the period when it was moving to the target. The objects in the four puck condition have a much higher velocity at the beginning of their path than in the other conditions. Figure 6 shows the average speed of the objects in each condition 0.2 seconds after the user has begun moving it. The speed of the object in the four pucks condition is significantly higher than in the one puck ($p < 0.001$) and touchscreen conditions ($p < 0.001$). There was no significant difference between the one puck and touchscreen conditions ($p = 0.71$).

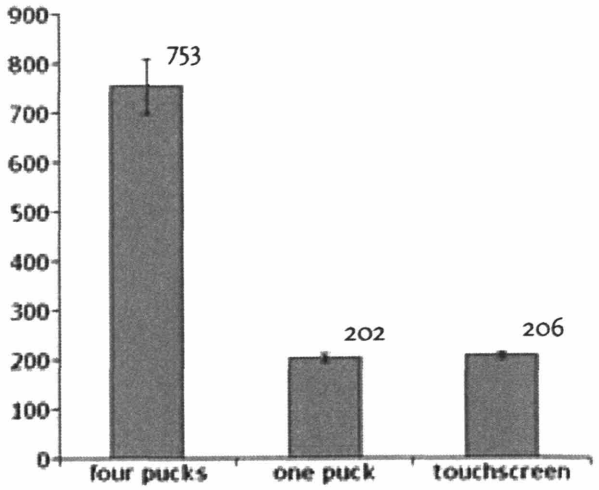


Figure 6: Speed of object (mm/s) 0.2 seconds after user has begun moving it. Mean and standard error are shown.

User Comments

Two subjects commented that it was difficult to move one's finger around on the touchscreen surface. However, these same two subjects also discovered during the practice period of the experiment that by using their fingernail, or a different portion of their finger, they could move their finger on the screen with greater ease. While the friction of the touchscreen was greater than that of the laminate surface users touched in the other two experimental conditions, we do not believe this factor influenced our primary results in a significant way, as there was no overall significant difference found in the completion times for the Touchscreen task and the One Puck task.

Three subjects noted that in the Four Puck task, other objects often served as obstructions along the path of the object to be moved. The most common strategy used in response to this was to move the puck along a curved path, though some users also experimented with lifting the puck up off of the surface and placing it down onto the target. One subject tried sliding pucks along the surface of the table from one hand to the other but abandoned this technique after a few attempts. Finally, one subject moved pucks with his dominant hand while placing his non-dominant hand just past the target, so that he could rapidly approach the target without overshooting it.

Discussion

The experimental data supports hypotheses H1, H2 and H3. Subjects complete the task more quickly in the four puck condition, and acquire the objects to be moved more rapidly in that condition as well. One unexpected finding is that subjects also acquire the objects to be moved more quickly in the one puck condition than the touchscreen condition. One possible explanation for this difference in performance is the tendency of many users to slide

the puck along the sensing surface rather than pick it up and put it down again. This sliding causes users to approach the target with a horizontal motion rather than a vertical one. The horizontal motion may make it easier to recover quickly if the original attempt to acquire the puck fails and must be repeated. In contrast, many subjects used a stabbing gesture to acquire an object in the touch-screen condition, which may require more time to recover if the initial attempt fails, due to the increased friction with the glass pane due to the momentum of the finger.

Once subjects had acquired the object to be moved, they were faster to move it in the four pucks condition than in the other two, disproving hypothesis H4. It appears this speed advantage comes from the fact that one can combine the acts of grasping and moving the puck into one continuous sweeping gesture. As subject's hands often do not fully stop when acquiring an object in the four pucks condition, the initial speed of travel toward the target is higher. This is an additional advantage in terms of overall performance time in the four pucks condition.

The results suggest that in applications where users must manipulate a variety of different objects and must switch between manipulating different graphical objects frequently, users would be better off having physical objects directly mapped to those graphical objects. However, as the number of objects increased, at some point the physical objects on the table would probably become a hindrance more than a help.

In pilot experiments, we informally observed that as the number of objects on the table increased, the performance in the Multiple Pucks condition degraded. We believe this is because with increasing numbers of pucks, grasping the correct one out of a crowded group on the table becomes more of a fine motor task than a coarse

one. As well, the need to avoid many objects during the movement process also adds a fine grained motor control element to the movement task.

The results suggest that latency in interactive systems similar to the Multiple Pucks condition of our experiment is less of an issue than in interfaces based on Touchscreens. In applications where physical objects are bound to graphical content, passive haptic feedback from the physical objects can help users interact more quickly.

Evaluation of Actuation in Pico

I this second experiment, evaluated Pico using a simplified version of the cellphone tower layout application discussed earlier in this thesis. The aim was to understand how mechanical actuation would affect the users' problem solving strategies, and how users would react to an interface involving actuation. 15 subjects were asked to use the interface to lay out a group of four cellular telephone towers to maximize a "coverage score" displayed on the screen or table. With each different interface, subjects were given a chance to try the interface and ask any questions they had before the timed portion of the interaction. Because I wanted to understand how subjects would interact with the systems when the underlying mathematics were opaque, they were not given an explanation of how cellular radio propagation works. They were simply asked to position the towers to try to reach a coverage score of 400, and given 4 minutes and 30 seconds to complete the task. To focus the users on the task of positioning the towers in space, the manipulation of other tower parameters normally available with Pico was disabled. I chose the task of cellphone tower layout for this evaluation because it was mathematically complex enough to benefit from computer augmentation, yet the goal of the task was conceptually simple enough to be understood by a novice user.

The experiment had three conditions: **Pico**, **Pico without actuation**, and **Screen**.

Screen: Subjects used a three button mouse to move the towers. The user could click on a tower with the left mouse button and drag it to a desired position, and could right click on the tower to lock it in place. With the middle mouse button, subjects could draw a line on the screen that towers could not cross. These middle and right mouse button features were added to ensure feature par-

ity with the other experiment conditions. The software running the screen-based condition was identical to that in the other two conditions, save a small change to the tracking code to track mouse clicks instead of Pico pucks. As in the other conditions, the computer used a simulated annealing process to attempt to maximize the coverage score on its own by moving the towers.

Pico: The experimental task was performed with four Pico pucks, each associated with a cellphone tower. In addition, subjects were provided with a flexible barrier and three hollow discs filled with sand. The experimenter explained that a disc could be placed on top of an object to stop it from moving, and the barrier could be bent into any shape to constrain the motion of the pucks.

Pico without actuation: This case was the same as the Pico condition, except that the power supply to the magnet array was turned off, preventing the Pico software from moving any pucks on its own.

Three conditions were used in order to separate the effects of being able to use two hands at the same time and interact directly with physical objects, and the effects of using actuation. The 15 subjects ranged from 19 to 55 years old (median 33) and consisted of 5 females and 10 males. The order of presentation of the conditions was randomized to counteract ordering effects. After subjects had used all three interfaces, they were asked to rate each interface on a 7-point Likert scale, and were asked a series of open-ended questions about what they liked and disliked about each interface, and if they found any aspect particularly surprising or frustrating.

Hypotheses

I believed that subjects would find it easier to move their hands between objects on the table, than to move between towers on

the screen with the mouse, and that the tactile feedback from Pico would cause subjects to develop an impression more quickly about whether a given problem solving strategy would be effective or not. As a result, I expected subjects to favor a series of quick manipulations of different groups of pucks, rather than more prolonged periods of interaction with a single object, when compared to behavior in the screen-based condition. This yields the first hypothesis:

H1: Users will shift their control between objects more often in the actuation condition than with the screen based condition.

Alternative mechanisms for constraining the motion of the cell-phone towers are provided in the screen and Pico cases. While these provide similar functionality, I expect users will be more likely to use them in the Pico case:

H2: Users will constrain the motion of pucks more in the Pico case than the screen case.

Finally, I hypothesized that the differences I expected to see between the screen condition and the Pico condition would not be fully explained by the use of physical objects alone. Actuation would play a significant role as well:

H3: Users will shift their control between objects more often in the actuation condition than with the non-actuation condition.

Results

Data was collected using several methods. The application software logged user input to a datafile for later analysis. However, in the Pico conditions (with and without actuation) it was difficult to determine what the user was doing by relying exclusively on the software logging feature. One reason is that occasional "hops" in

the puck location data might be mistaken for user interactions. As a result, for these conditions a videocamera was pointed at the interaction surface such that the user's hand motions could later be analyzed.

We compared the number of times each subject switched objects on the tabletop conditions, and compared this to the number of times each subject switched objects in the screen based condition. The results are shown in figure 7. We found that the number of these cycles in the Pico condition was significantly higher than in the Pico without actuation condition ($p < 0.05$) and the screen condition ($p < 0.001$). The Pico without actuation condition also involved more switching between objects than the screen condition ($p < 0.001$). Subjects also used constraints more often in the Pico condition than the screen condition as shown in figure 8 ($p < 0.05$).

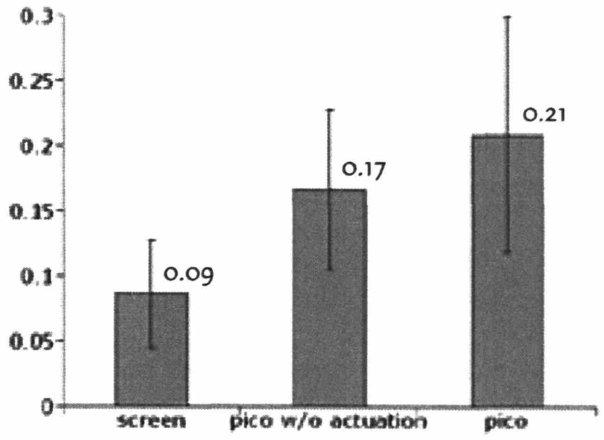


Figure 7: Number of interface objects acquired per second in the three experimental conditions. Mean and standard deviation are shown.

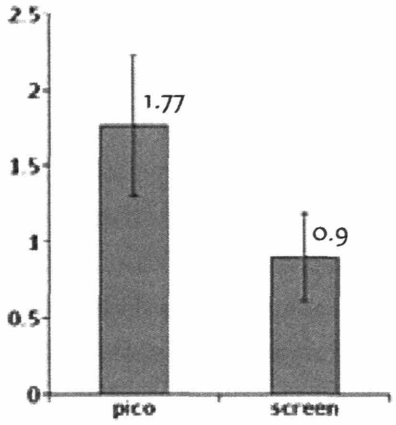


Figure 8: Number of constraints used per minute of interaction in the pico and screen conditions. Constraints were not used in the pico without actuation condition. Mean and standard deviation are shown.

Four subjects were able to complete the task in the screen condition, versus five in the Pico without actuation condition, and seven in the Pico condition. In the interest of seeing if there was a relationship between the tendency to switch objects and successfully completing the task, I grouped the data across all tasks into two groups, trials in which the subject completed the task, and trials in which the subject did not. I compared these two groups with a one factor ANOVA and found that in trials where subjects completed the task successfully, they tended to switch significantly ($p < 0.05$) more frequently between moving different cellphone towers in the interface, shown in figure 9.

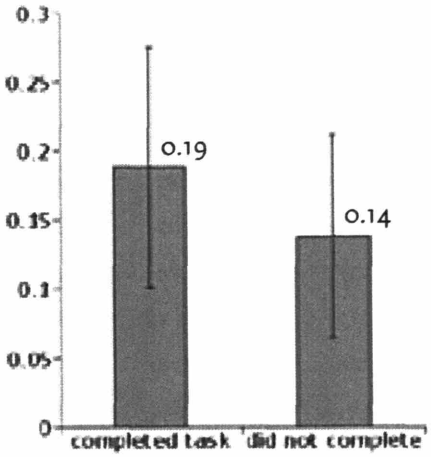


Figure 9: Number of interface objects acquired per second across all experimental conditions. In the "completed task" category are trials in which the subject was able to obtain a total layout score of 400 or greater. The "did not complete" category shows trials in which the score of 400 was not reached. Mean and standard deviation are shown.

Subjects also ranked each interface on a 7-point Likert scale regarding its ease of use. These results are summarized in figure 10. The only difference between these scores that reached statistical significance was the difference between the Screen and Pico interfaces ($p < 0.05$).

Users' qualitative reactions to Pico were more divided than the quantitative data might suggest. In response to Pico one subject said "I felt like if I moved one thing the computer was trying to balance it by moving the others." Another said "I got the feeling of where they [the towers] wanted to go... It was better than seeing." A third subject said "I felt like I was collaborating with the computer to solve the problem" and that in the Pico case it "feels like the computer wants to help more." Some subjects also appreciated the ability to move more than one object at a time in the Pico conditions, both with and without actuation.

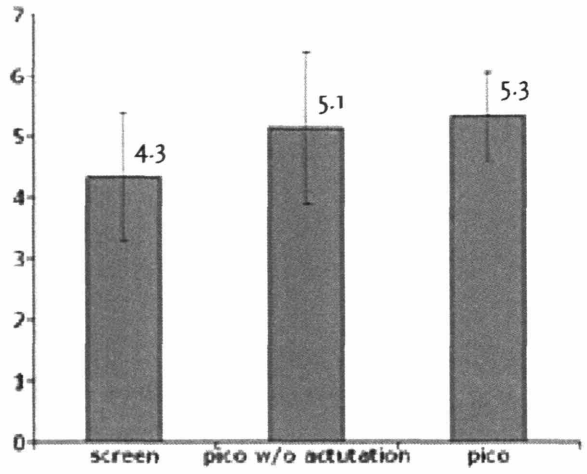


Figure 10: Subjects' average ranking of ease of use of the three experimental conditions on a 7-point Likert scale. Error bars represent standard deviation.

At the same time, some users seemed to find the system a bit frustrating. One user commented that he would prefer if the user and computer "take turns" while moving the towers, rather than moving them at the same time. He felt that the movement was imprecise, and commented that "I like to have really precise control when I'm interacting." Another commented that he "wasn't sure how to benefit from the computer's input." A third expressed a preference for moving only one object in the interface at a time. In his interaction with Pico, this subject placed weights on top of three of the objects, effectively limiting the computer's actuation to just one object at a time. Two subjects commented that it was very easy to establish constraints with the mouse because, in the words of one, "all I have to do is click." They seemed to prefer this approach to the Pico condition because it required less physical movement. Users uniformly found it frustrating when the computer occasionally moved an object on its own in a way that decreased the overall score. They seemed less tolerant of computer error than they might be of human error.

Subjects in the two Pico conditions used a variety of hand gestures to manipulate multiple objects at the same time. These included using both hands, using separate fingers on a hand to independently manipulate distinct pucks and pinning pucks to the table to constrain their motion. Often in the case of constraining motion, a user would begin by holding a puck in place with one hand, while reaching for a weight to place on top of the puck with the other hand, which he or she would then quickly substitute for the hand pinning the object, freeing both hands to interact with other objects. Constraints were used primarily in two ways. One was to facilitate a "step by step" problem solving process, where users would try to find the best place for a particular tower, and then lock it down, and move to the next one. The other was in response to motions the computer was causing. Users would employ a barrier or weight to prevent an action from happening again.

Another interesting strategy was a repeated "poking" gesture that subjects used on the Pico condition. Subjects would push a puck with an extended index finger about an inch or two on the table, and then see how the computer responded and the coverage score changed. Depending on the result, they might poke the same object again or switch to another one, at times moving an object from one side of the table to the other using a series of short pushes.

The results indicate that subjects switch between manipulating different objects more frequently using Pico than with the other two conditions. This more rapid switching between objects suggests that users iterate more rapidly among alternative problem solving strategies (e.g. moving puck A, versus puck B, versus A and B together etc.) with Pico using actuation than with the other two conditions. This difference appears partially due to the ease of grasping and manipulating objects on the table (also seen in the difference seen

between the screen and Pico without actuation conditions) and partially due to the Actuation in the Pico condition. The data also suggests that in the tasks presented to the subjects, switching between multiple problem solving strategies (a breadth-first search) was more effective than exploring fewer strategies for a longer period of time (a depth-first search).

In summary, the results suggest that Pico makes it easier to quickly explore various potential solutions to a spatial layout problem, when compared to Pico without actuation, and the screen-based interface. As a result subjects are more inclined toward many brief interactions with multiple pucks rather than longer periods of sustained exploration with a single object. In the Pico condition, subjects seemed to more readily reject approaches that did not seem promising, and were more likely to successfully complete the task. Due to the prevalence of brief interactions with different pucks in the system, and the faster decision making with Pico's tactile feedback, I believe Pico is a step toward interaction more like what we experience with purely mechanical systems. Of course, the feedback from a mechanical system such as a bicycle is immediate; it is not delayed by the computer as in Pico. However, as the sensing technology gets faster, and computers increase in speed, we can expect the tactile dialog that happens between Pico and the user to occur at an increasing rate, which should provide further usability benefits.

7 Discussion and Future Work

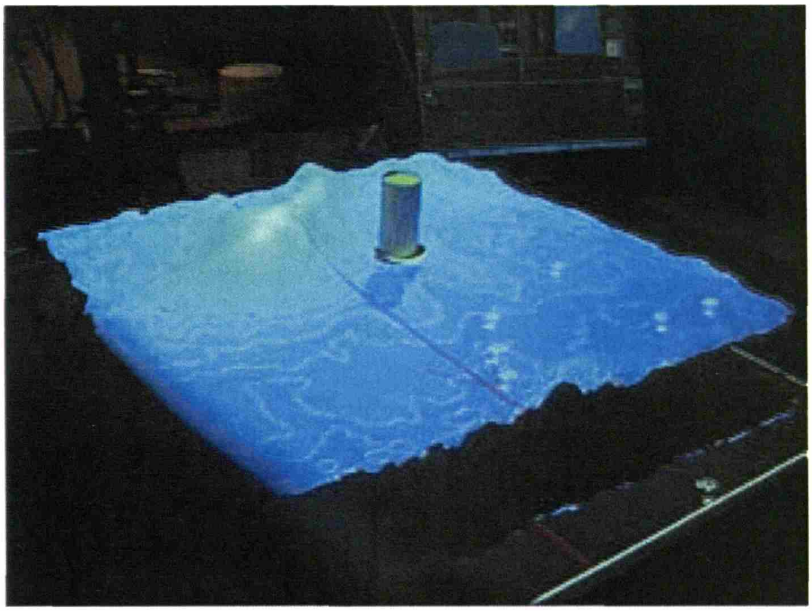
One of the main accomplishments of this thesis is that it supports improvisation with physical objects in the user's environment to help perform the user's task. While many of the objects that surround us daily are designed for a specific function, we often appropriate them for tasks other than their intended function when needed. For example, a chair can also be used as a doorstop or a stool, or as a place to hang one's jacket. Our mechanical intuition about how objects in the world interact with each other makes it easy to think about how to adapt familiar objects to new kinds of problems. Because Pico translates aspects of a computational system into a mechanical one, we suddenly have at our disposal the rich variety of physical objects in our environment to help us interact with it. In an improvisational manner, one can experiment with using different objects on the tabletop interaction surface until finding one that behaves as desired. For example, in the cellphone tower placement application discussed throughout this thesis, one might use a coffee cup to keep a tower out of a certain area. Later, one might want to increase the radius of that forbidden area by replacing the coffee cup with a larger diameter object such as a roll of tape.

Interfaces that support this temporary repurposing of physical objects to help perform a task could be called "improvisational interfaces." This improvisation can take place on several different levels:

Object as symbol - Users may associate an object that is meaningful to them with an interface element such as an RFID tag to give that object an identity. One example of this approach is in the Sensetable system, where one might place a nearby object on top of a generic puck as a symbol to give that object a distinct meaning in the context of the application. This object would serve as a mental reminder to the users and would not be sensed by the computer.

A battery taped to the top of a Sensetable puck, placed there by the user to represent that pucks role as an energy source in a physics simulation.

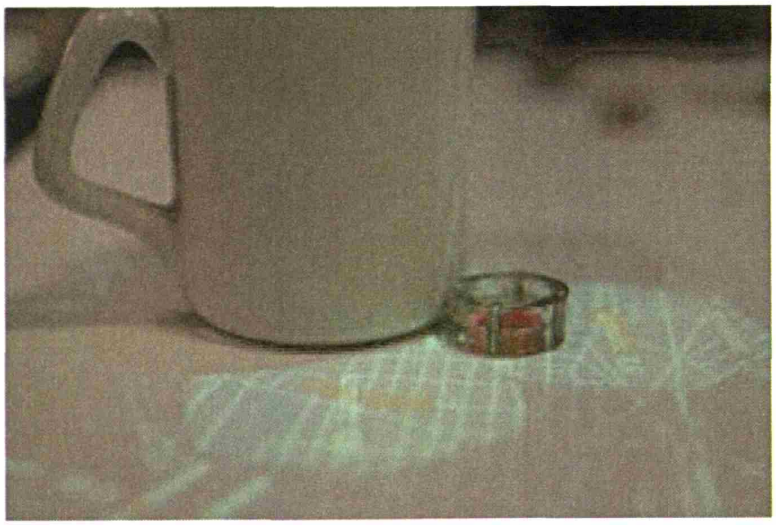
Object as form - A richer level of improvisation with physical objects involves selecting and using objects based on their physical form. This interaction can be seen in the Illuminating Clay [Piper 2002] landscape planning tool. The system uses a laser scanner to create a topographic model of the terrain on the tabletop. Any object can be used to create that terrain, ranging from clay, wooden blocks and cardboard to more unconventional repurposed modeling materials that may happen to be nearby, such as office supplies.



A wooden cylinder used as a stand-in for the physical form of a building in the Illuminating Clay landscape planning system.

Object as part of mechanical system - A still richer level of improvisation is possible when the mechanical properties of an object, such as friction, mass and flexibility, are incorporated into an interface. This occurs through the integration of a physical feedback loop into the interface. This level of integration is present in Pico, where physical objects such as rubber bands can guide software pa-

rameters toward similar values, and physical barriers, such as a ruler or a book, can control the range of values a software parameter can obtain. My favorite example of this repurposing is the use of soap and water to lubricate part of the tabletop surface so that software parameters associated with that part of the table would change more readily than others. In this level, the true improvisational interface, the interface designer relinquishes responsibility for the incorporation of physical affordances and metaphors, and empowers the user to appropriate any of a huge variety of objects at his or her disposal in a way that is useful in the context of the task at hand. This shift is in contrast with most tangible interfaces, in which the use of physical affordances and metaphors is the job of the designer, and is unchangeable by the user.



A coffee cup, preventing a cellular telephone tower in the Pico system from entering a certain area of the map.

Why Improvisational Interfaces?

The advantages of improvisational interfaces center around the idea that immediate, imprecise interaction and feedback is often more desirable than the rigid, precise and slower interaction we have become so accustomed to when using computes. Some of the specific advantages of the improvisational approach are:

- Improvisational interfaces provide the user with a **rich, familiar vocabulary** with which to interact with the computer. The vocabulary is familiar because the user can employ objects he or she has daily experience with and have reasonable assumptions about how these objects will behave in the context of the interface. The richness of the interaction vocabulary comes from the diversity of physical objects surrounding us every day, and the resulting diversity of mechanical interactions that are possible on the tabletop interface. For example, a common construct in GUI interaction is that of the "lock", an attribute that one sets on an object on the screen to prevent it from being changed. This object may be a file, part of an image, etc. In a GUI, these locks are almost always completely on or off: something can either be changed or not. With Pico, there are infinite possibilities between "locked" and "unlocked" because there are an infinite number of ways to constrain the motion of a puck. One could place a heavy weight on top of a puck, making it impossible for the computer to move it (but still possible for the user), or one could place a small weight on a puck, slowing down its motion only slightly due to increased friction. There are many possibilities in between these two extremes, limited only by the user's own hands and the objects at his or her disposal.

- Improvisational interfaces change application behavior **faster than programming**. All of the interactions presented in this paper could be simulated on-screen using custom-developed computer software. However, each of these possible interactions would have to be foreseen to be included, and many would likely take a talented programmer hours if not days to accurately simulate. With the improvisational approach, this type of reprogramming is not necessary. One does not even have to restart the program. If one wants to interact with an application in a way not considered by the developers, one simply adjusts the constraints of the system by manipulating objects on the tabletop. This manipulation of physical objects takes seconds, rather than hours, to do.

- Improvisational interfaces **encourage accidental discoveries**. They make it so easy to experiment that people are bound to make mistakes. As the history of scientific discovery shows, these mistakes are often critical in helping people solve complex problems, or approach them in a new way. When a mistake is made, the ease of understanding the cause and effect relationships between the different parts of the mechanical system make it easy to understand the implications of the error.

The vast array of possible interactions present with an interface that can incorporate everyday physical objects can be described by what Prof. John Maeda once called Patten's Law of Opportunity. Specifically, that:

"The opportunities for improvisation within an interface increase proportionately to the square of the number of types of physical objects one can use within the interface."

Each new physical object added to the tabletop can interact mechanically with all of the other objects already on the table, leading to a number of possible interactions that grows with the square of the number of available objects, whether the objects are right next to the user, or waiting on local store shelves. With an improvisational interface like Pico, each of these objects becomes a potential improvised physical tool to solve computational problems.

What Next?

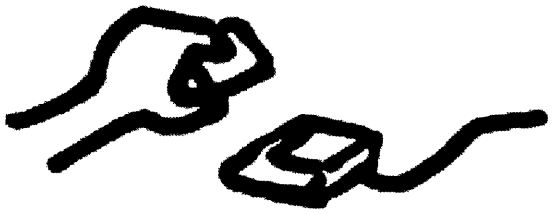
The applications implemented on the Pico system to date have been spatial in nature. However, there are a wide variety of applications that do not have literal interpretations of space that could be mapped to the interaction space Pico provides. For example, in a business supply chain simulation, distance between objects on the table could represent shipping time between those locations. One might change the allocations of shipping resources to the various parts of the supply chain by moving the objects, while the computer ensured that the total shipping budget was not exceeded.

Another area to be explored is the use of actuation as an expressive medium. When collaborating with Roberto Aimi to explore how sound could be integrated with Pico, we found that the sound emphasized the playfulness of the interaction. It was as if the computer was trying to tell us something. When we would move an object, and the computer would respond be immediately moving it back to where it was before, it created a call and response dialog with an interesting sonic rhythm. We are working on developing this further, creating a distant sibling of the Audiopad that incorporates physical forces as an expressive musical and visual element in a tabletop composition.

Pico points to opportunities for a larger degree of improvisation with everyday physical objects in the context of human-computer interfaces. My hope is that Pico is just the beginning, merely the first exploration of this idea within the larger context of interaction design. We can easily imagine interfaces where mechanical actuation is incorporated into many different types of interfaces off of the tabletop, creating free-form, improvisational interfaces that encourage experimentation, helping users make discoveries and change perspectives. Some ideas for these next steps follow:

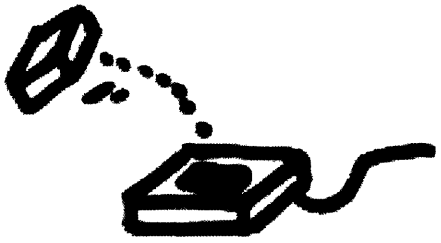
Jumping objects

A user places a physical token into a reader.



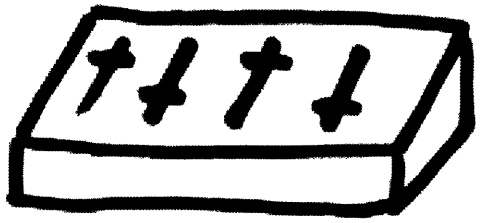
The software registers this action, and determines that this action is forbidden according to a rule defined in software.

So, the reader ejects the block!

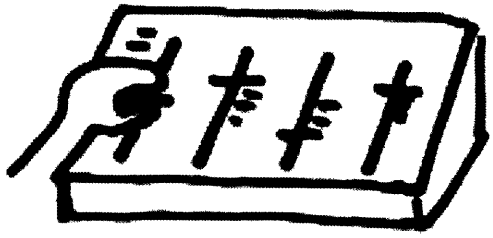


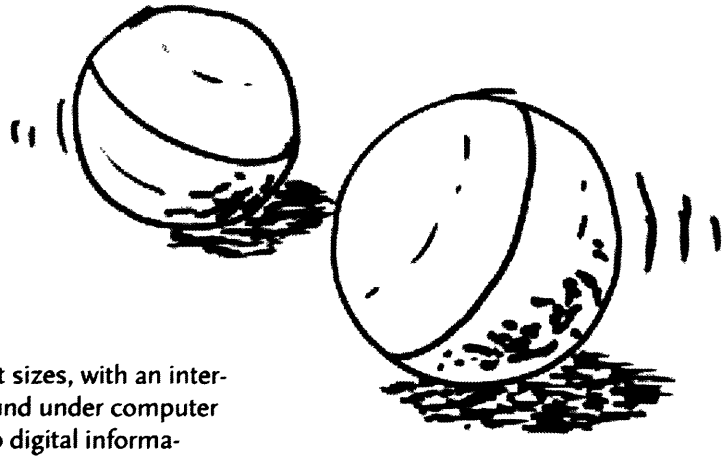
Actuated Sliders

An array of linear potentiometers with motors attached, as used in some high end audio equipment, could be used in the same way as is Pico in this thesis, for a different set of problems.



The user's movement of one slider changes internal software state, reflected in the positions of other sliders. These movements could be physically constrained in a variety of ways with everyday physical objects, for example to set an upper and lower bound for a parameter.





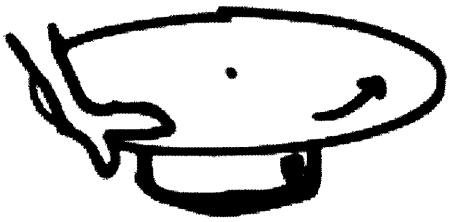
Actuated Beachballs

Spheres, potentially of different sizes, with an internal motors and CPUs, roll around under computer control to act as an interface to digital information. These objects are perhaps better suited to the three dimensionality of the physical world than are cylindrical Pico pucks. These could roll up and down hills, be placed on shelves of varying heights, and be moved by users to expand the concepts in this thesis to interaction with 3D data.

Turntable

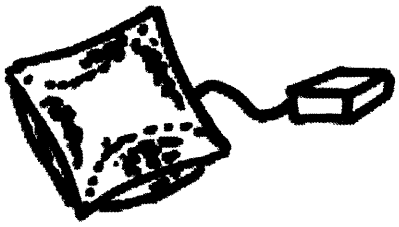
A turntable rotates under computer control to represent a changing parameter, such as the passage of time in an interactive simulation or musical performance.

The user can grab this wheel and spin it as desired to move forward or backward along the timeline. When the user releases the wheel, it keeps spinning a its current velocity until the user touches it again.



Inflatables

The volume of air inside an inflatable reservoir represents and controls a parameter in software.



The computer's inflation of the object can be countered by the user squeezing, or by placing the inflatable inside a small box, for example.

8 References

Badros, G., Borning, A., and Stuckey, P., "The Cassowary Linear Arithmetic Constraint Solving Algorithm," ACM Transactions on Computer Human Interaction, Vol. 8 No. 4, December 2001, pages 267-306. (pdf; ps.gz)

Ballard, D.H., Hayhoe, M.M., Pook, P.K., and Rao, R.P.N., Deictic codes for the embodiment of cognition. Behavioral and Brain Sciences, 20(4), 723-742, 1997.

Ballard, D.H., Hayhoe, M.M., and Pelz, J.B., Memory representations in natural tasks. Journal of Cognitive Neuroscience, 7(1), 66-80, 1995.

Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T., "Toolglass and Magic Lenses: A See-Trough Interface," Proceedings of ACM SIGGRAPH 1993, p. 73-80.

Brave, S. and Dahley, A., inTouch: A Medium for Haptic Interpersonal Communication (short paper), in Extended Abstracts of Conference on Human Factors in Computing Systems CHI '97), (Atlanta, March 1997), ACM Press, pp. 363-364. wpe27.jpg (1103 bytes)

Brooks, F., Ouh-Young, M., Batter, J., Kilpatrick, P., Project GROPE: Haptic Displays for Scientific Visualization, in Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 90), ACM Press, pp. 177-186, 1990.

Burdea, G.,"Haptics Issues in Virtual Environments," invited paper, Proceedings of Computer Graphics International 2000, Geneva, pp. 295-302, June 18-23, 2000.

Buxton, W. (1986). "There's More to Interaction than Meets the Eye: Some Issues in Manual Input." in User Centered System Design., pp. 319 - 337., 1986.

Dahley, A., "Expressive Kinetic Objects", http://tangible.media.mit.edu/projects/exp_kin_obj/ Referenced January 3, 2006.

Dietz, P., Leigh, D., DiamondTouch: a multi-user touch technology, Proceedings of the 14th annual ACM symposium on User interface software and technology, November 11-14, 2001, Orlando, Florida

Donald Norman, The Design of Everyday Things, Currency, 1990. ISBN 0465067107

Fitts, P.M., The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology, 47, 381-391, 1954.

Fitzmaurice, G., Graspable User Interfaces. Ph.D. Thesis, University of Toronto, 1996.

Fitzmaurice, G., Ishii, H., Buxton, W., Bricks: laying the foundations for graspable user interfaces, Proceedings of the SIGCHI conference on Human factors in computing systems, p.442-449, May 07-11, 1995, Denver, Colorado, United States

Fitzmaurice, G., "Situated information spaces and spatially aware palmtop computers," Communications of the ACM, July 1993, 36(7), p.38-49.

Fjeld, M., Bichsel, M., And Rauterberg, M. (1998). BUILD-IT: An Intuitive Design Tool Based on Direct Object Manipulation. In Gesture and Sign Language in Human-Computer Interaction, Lecture Notes in Artificial Intelligence, v.1371, Wachsmut and Fröhlich, eds. Berlin: SpringerVerlag, pp. 297-308.

Frei, P., Su, V., Mikhak, B., and Ishii, H., curlybot: Designing a New Class of Computational Toys, in Proceedings of Conference on Human Factors in Computing Systems (CHI '00), (The Hague, The Netherlands, April 1-6, 2000), ACM Press, pp.129-136

Gray, W.D., and Boehm-Davis, D.A., Milliseconds Matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. Journal of Experiment Psychology: Applied, 6(4), 322-335, 2000.

Greenberg, S., and Fitchett, C. (2001). Phidgets: Easy Development of Physical Interfaces through Physical Widgets. In Proceedings of UIST 2001, pp. 209-218.

Goldstein, H., Poole. C., Safko., J., Classical Mechanics. Addison Wesley, 2002. ISBN: 0201657023

Guiard, Y., "Asymmetric Division of Labor in Human Skilled Bimanual Action: ?e Kinematic Chain as a Model," J. Motor Behavior, 19 (4), 1987, pp. 486-517.

Hinckley, K., Pausch, R., Proffitt, D., Patten, J., Kassell, N., Cooperative Bimanual Action, in Proceedings of Conference on Human Factors in Computing Systems, (CHI 97), Pages 27 - 34, 1997.

Hopkins, D., "Directional Selection is Easy as Pie Menus!", in ;login: The Usenix Association Newsletter, Vol. 12(5), 1987.

Ishii, H. and Ullmer, B., Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms, in Proceedings of Conference on Human Factors in Computing Systems (CHI 97), ACM Press, pp. 234-241, 1997.

Ishii, H. and Ullmer, B., "Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms," in Proceedings of ACM CHI '97, pp. 234-241, 1997.

Ishii, H., Mazalek, A., Lee, J., Bottles as a Minimal Interface to Access Digital Information (short paper), in Extended Abstracts of Conference on Human Factors in Computing Systems (CHI '01), (Seattle, Washington, USA, March 31 - April 5, 2001), ACM Press, pp.187-188

Ishii, H., Ren, S. and Frei, P., Pinwheels: Visualizing Information Flow in an Architectural Space (short paper), in Extended Abstracts of Conference on Human Factors in Computing Systems (CHI '01), (Seattle, Washington, USA, March 31 - April 5, 2001), ACM Press, pp.111-112

Ishii, H., Ullmer, B., Tangible bits: towards seamless interfaces between people, bits and atoms, Proceedings of the SIGCHI conference on Human factors in computing systems, p.234-241, March 22-27, 1997, Atlanta, Georgia, United States

Jacob, R., Reality-based Interaction: Understanding the Next Generation of User Interfaces (draft), retrieved from: http://www.eecs.tufts.edu/~jacob/theory on: January 10, 2006.

Jacob, R., Ishii, H., Pangaro, G., Patten, J., A Tangible Interface for Organizing Information Using a Grid, in Proceedings of Conference on Human Factors in Computing Systems (CHI '02), pp. 339-346.

Johnson, J., Roberts, T., Verplank, W., Smith, D., Irby, C., Beard, M., Mackey, K. The Xerox Star: A retro- spective. IEEE Comput. 22, 9 (Sept. 1989).

Katz, M., Capturing Sound: How Technology Has Changed Music, University of California Press. 2004.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., Optimization by simulated annealing, Science, vol. 220, pp. 671-680, 1983.

Kirsh, D., The intelligent use of space, Journal of Artificial Intelligence, 73(1-2), 31-68, 1995.

Madhani, A., G. Niemeyer, and J. K. Salisbury, The Black Falcon: A Teleoperated Surgical Instrument for Minimally Invasive Surgery," in Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Victoria, BC, Canada, 1998.

Malone, T. et al., Tools for Inventing Organizations. In Management Science 45(3) pp 425-443, March, 1999.

Marks, J. et al. 1997. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In Proceedings of the 24th Annual Conference on Computer Graphics and interactive Techniques ACM Press/Addison-Wesley Publishing Co., New York, NY, 389-400.

Metropolis,N., A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, "Equation of State Calculations by Fast Computing Machines", J. Chem. Phys.,21, 6, 1087-1092, 1953.

Motorola, Netplan RF engineering user's manual version 6.1, Tech. Rep., Motorola, May 2002.

NextWindow, NextWindow Touch Interactive Display. site: http://www.nextwindow.com Accessed January 13th, 2006.

Nielsen, J., Usability Engineering, Academic Press, Boston, 1993.

O'Hara, K.P., and Payne, S.J., The effects of operator implementation cost on planfulness of problem solving and learning. Cognitive Psychology, 35, 34-70, 1998.

Pangaro, G., Maynes-Aminzade, D., Ishii, H. The Actuated Workbench: Computer-Controlled Actuation in Tabletop Tangible Interfaces, in Proceedings of Symposium on User Interface Software and Technology (UIST '02), 2002.

Paradiso, J., et al., "Sensor Systems for Interactive Surfaces," IBM Systems Journal, 39(3-4), 892-914,2000.

Patten, J., Ishii, H., Hines, J., Pangaro, G., Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces, in Proceedings of Conference on Human Factors in Computing Systems (CHI '01), ACM Press, pp.253-260, 2001.

Patten, J., Recht, B., Ishii, H., Audiopad: A Tag-based Interface for Musical Performance, in Proceedings of Conference on New Interface for Musical Expression (NIME '02), 2002.

Pausch, R., Crea, T., Conway, M., A Literature Survey for Virtual Environments: Military Flight Simulator Visual Systems and Simulator Sickness, PRESENCE: Teleoperators and Virtual Environments 1:3, January, 1993.

Piper, B., Ratti, C., and Ishii, H., Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis, in Proceedings of Conference on Human Factors in Computing Systems (CHI '02), pp. 355-362.

Potter, R., , L. J. Weldon, B. Shneiderman, Improving the accuracy of touch screens: an experimental evaluation of three strategies, Proceedings of the SIGCHI conference on Human factors in computing systems, p.27-32, May 15-19, 1988, Washington, D.C., United States

Raffle, H., Parkes, A., Ishii, H. Topobo: A Constructive Assembly System with Kinetic Memory, in Proceedings of Conference on Human Factors in Computing Systems (CHI '04), (Vienna, Austria, April 24 - April 29, 2004)

Rekimoto, J. and Sciammarella, E., ToolStone: Effective Use of the Physical Manipulation Vocabularies of Input Devices, in Proceedings of UIST 2000, ACM Press.

Rekimoto, J., Saitoh, M., Augmented surfaces: a spatially continuous work space for hybrid computing environments, Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, p.378-385, May 15-20, 1999, Pittsburgh, Pennsylvania, United States

Rekimoto, J., SmartSkin: an infrastructure for freehand manipulation on interactive surfaces, Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves, April 20-25, 2002, Minneapolis, Minnesota, USA

Reznik, D., Canny, J., "C'mon Part, do the Local Motion!", 2001 IEEE International Conference on Robotics & Automation (ICRA), Seoul, Korea, May 2001.

Rosenfeld, D., Zawadzki, M., Sudol, J., Perlin., K.,"Physical Objects as Bidirectional User Interface Elements," IEEE Computer Graphics and Applications, vol. 24, no. 1, pp. 44-49, January/February, 2004.

Rosenfeld, D., et al., Planar Manipulator Display, http://cat.nyu.edu/PMD

Scarlatos, L.L., An Application of Tangible Interfaces in Collaborative Learning Environments, in ACM SIGGRAPH 2002 Conference Abstracts and Applications, 125-126.

Sears, A., Shneiderman, B., High precision touchscreens: design strategies and comparisons with a mouse, International Journal of Man-Machine Studies, v.34 n.4, p.593-613, April 1991

Shen, C., Lesh, N., Vernier, F., Forlines, C., Frost, J., Sharing and building digital group histories, Proceedings of the 2002 ACM conference on Computer supported cooperative work, November 16-20, 2002, New Orleans, Louisiana, USA

Shieber, S., A Call for Collaborative Interfaces, ACM Computing Surveys, volume 28A (electronic), December, 1996.

Snibbe, S., Maclean, K, Shaw, R., Roderick, J., Verplank, W., and Scheeff, M. (2001). Haptic Techniques for Media Control. In Proceedings of UIST 01.

Svendsen, G.B., The influence of interface style on problem solving. International Journal of Man-Machine Studies, 35(3), 379-397, 1991.

TSpaces, IBM Systems Journal, August 1998.

Ullmer, B., Tangible Interfaces for Manipulating Aggregates of Digital Information. Phd Thesis. Massachusetts Institute of Technology. August 2002.

Ullmer, B., Ishii, H. The metaDESK: models and prototypes for tangible user interfaces, Proceedings of the 10th annual ACM symposium on User interface software and technology, p.223-232, October 14-17, 1997, Banff, Alberta, Canada

Ullmer, B., Ishii, H., Glas, D., mediaBlocks: physical containers, transports, and controls for online media, Proceedings of the 25th annual conference on Computer graphics and interactive techniques, p.379-386, July 1998

Underkoffler, J., and Ishii, H., Urp: A Luminous-Tangible Workbench for Urban Planning and Design, in Proceedings of Conference on Human Factors in Computing Systems CHI '99), ACM Press, pp. 386-393, 1999.

Weiser, M. The Computer for the 21st Century. Scientific American, 265 (3), pp. 94-104, 1991.

Wellner, P., Interacting with paper on the DigitalDesk, Communications of the ACM, v.36 n.7, p.87-96, July 1993

Wellner, P., Mackay, W., and Gold, R. Computer Augmented Environments: Back to the Real World. Commun. ACM, Vol. 36, No. 7, July 1993.

Wisneski, C., Ishii, H., Dahley, A., Gorbet, M., Brave, S., Ullmer, B. and Yarin, P., Ambient Displays: Turning Architectual Space into an Interface between People and Digital Information, in Proceedings of International Workshop on Cooperative Buildings CoBuild '98), (Darmstadt, Germany, February 1998), Springer Press, pp. 22-32.

Wu, M., Balakrishnan, R., Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays, Proceedings of the 16th annual ACM symposium on User interface software and technology, November 2003

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