Mechanical Constraints as Computational Constraints in Tabletop Tangible Interfaces

James Patten
MIT Media Lab
20 Ames St.
Cambridge, Ma . 02139
jpatten@media.mit.edu

Hiroshi Ishii
MIT Media Lab
20 Ames St.
Cambridge, Ma . 02139
ishii@media.mit.edu

ABSTRACT
This paper presents a new type of human-computer interface called Pico (Physical Intervention in Computational Optimization) 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. The positions of these physical objects represent and control parameters inside a software application, such as a system for optimizing the configuration of radio towers in a cellular telephone network. The computer autonomously attempts to optimize the network, moving the objects on the table as it changes their corresponding parameters in software. As these objects move, the user can constrain their motion with his or her hands, or many other kinds of physical objects. The interface provides ample opportunities for improvisation by allowing the user to employ a rich variety of everyday physical objects as mechanical constraints. This approach leverages the user's mechanical intuition for how objects respond to physical forces. As well, it allows the user to balance the numerical optimization performed by the computer with other goals that are difficult to quantify. 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.

Author Keywords
tangible interfaces, physical interaction, interactive surface, improvisation, actuation.

ACM Classification Keywords
H.5.2 User Interfaces

INTRODUCTION
The physical form of many everyday mechanical systems helps users quickly discover how these systems work and how to use them. One example is the record turntable. Because the mechanism through which this device functions is exposed to the user, some users have developed interaction techniques that the inventors of the device likely had never imagined, such as "scratching" the record as a part of musical performance. In a similar spirit, we have developed a system called Pico (Physical Intervention in Computational Optimization) that simultaneously represents and controls the high level structure of a software process with a mechanical process. The user can leverage his or her mechanical intuition about the way physical objects respond to forces and interact with each other to understand how common objects, such as a rubber band or coffee cup, might be used to constrain the underlying software process.

CHI 2007, April 28–May 3, 2007, San Jose, California, USA.

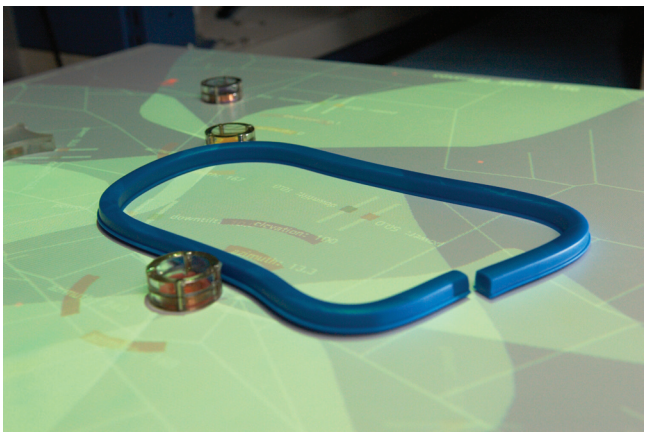
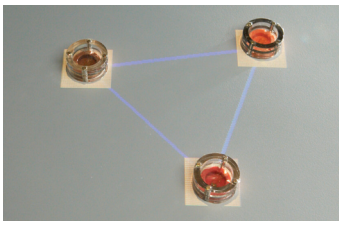


Figure 1: A flexible "artist's curve" constraining the motion of a cellphone tower in the Pico system.

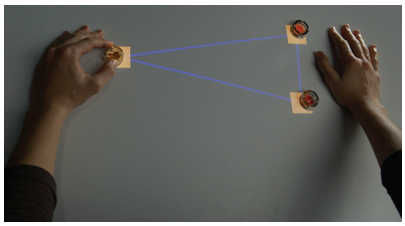
Objects on the Pico table are moved not only under software control using electromagnets but also by users standing around the table. The combination of these interactions, all governed by the friction and mass of the objects themselves directly affects the result of the task being performed. Additional information is graphically projected onto the table from above. In this paper we will show how this technique can be applied to spatial layout problems, and discuss how it could be applied to other types of tabletop interactions. To date we have built Pico applications for factory floor plan layout, CNC toolpath optimization, and cellular telephone tower layout (figure 1).

First we present a simple example of Pico in action, followed by a more complex one. We then present related work and describe the technical details of our implementation. We discuss the variety of interaction techniques that Pico enables and an experiment evaluating Pico. We conclude with a discussion of Pico's implications on future user interface design.

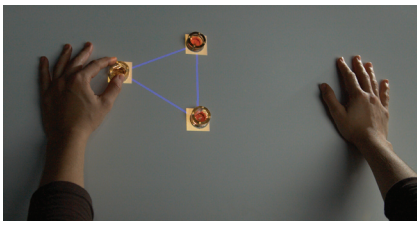
EXAMPLES
Pico works by iteratively attempting to resolve a series of software-defined constraints among a set of pucks on the interaction surface. A simple example of this process is shown in figure 2. A set of software constraints specifies that the distance between each of three objects on the table should be equal. The system iteratively measures the distances be-



1 Software constraints 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 constraints defined in software are again satisfied.

Figure 2: A simple Pico example

tween 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 task. 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 software constraints, the positions of the pucks adjust to conform to both the mechanical constraints applied by the user on the tabletop and the software constraints previously programmed into the application.

While creating equilateral triangles on a tabletop is a simple problem, this approach can also be applied to more complex tasks. It seems particularly well suited to problems that are both computationally complex, and benefit from human input. To develop the Pico concept we implemented three applications, the first involved determining the optimum placement of pieces to be cut out of a sheet of material in a CNC manufacturing context so as to use the raw material in an efficient manner. We also implemented a simple room layout application, where the configuration of rooms in a hospital can be optimized for efficiency based on the levels of traffic flow between the various rooms. Finally, we implemented a cellular telephone tower layout application. We believe this approach is applicable to a large class of spatial optimization problems, such as how to layout components on a printed circuit board, or how to arrange machines on a factory floor. Here we present cellular telephone tower layout as an example of this class.

Cellular Telephone Tower Placement

The goal of the cellular telephone tower layout application is to determine 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 causal relationships present in these changes.

The user adds new radio towers to the map by placing a puck on a "new tower" icon on the corner of the table. A tower appears and moves on the map along with the position of the puck. In software, the puck and the tower are "bound" together by this operation, and a software constraint engine tries to keep these two positions (physical and digital) as consistent as possible. Meanwhile, another software module attempts to move each tower to optimize the overall

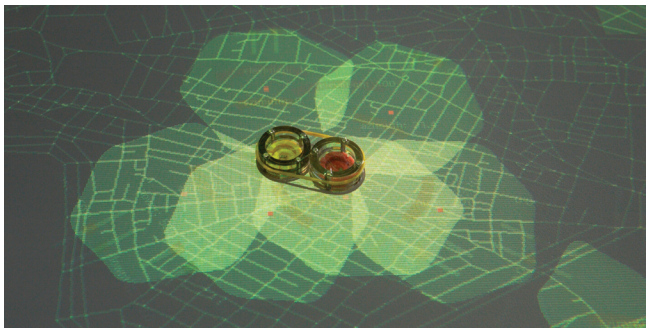


Figure 3a: Two adjacent cellphone towers in the Pico application. The computer is trying to separate these towers to improve the overall coverage, but is unable to because the towers are physically attached by a rubber band.

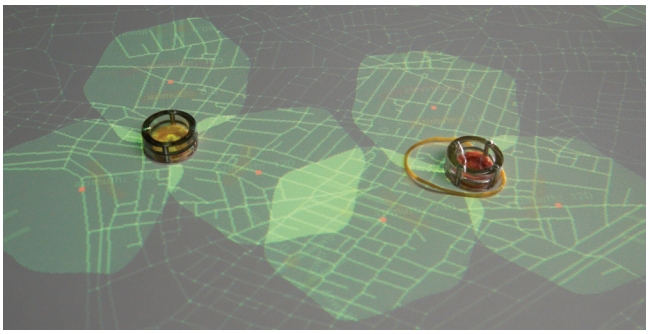


Figure 3b: When the rubber band is removed, the towers move apart in response to the removed constraint as the computer continues searching for a better layout.

coverage score. This movement of the tower influences the position of the puck it is bound to, while the position and physical forces upon the puck in turn influence the position of the tower. The end result is that the puck and tower gradually move on the table toward a location that provides better overall coverage.

The computer searches for the best place to put that tower according to a fitness function based on a variety of factors, including a cost score and a coverage score, using simulated annealing [15]. It scores locations near the current one, and tries to move the tower away from areas that score poorly and closer to ones that score well. If the user moves the tower around the map with his or her hand, he or she feels 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 in figure 3a. 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 as in figure 3b. If the user 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 simply by 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 mechanical 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.

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 software constraints with mechanical 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.

RELATED WORK

Tangible Interfaces

This work builds on a series of tabletop Tangible Interfaces[10] that focus on layout, planning and simulation applications, such as Urp[30], BUILD-IT[6, 7] and Sensetable[18]. These systems employ physical objects as user interface elements to represent and control the computational process. One limitation of these systems is that the physical objects cannot be moved under computer control to reflect changes in the underlying software state. As a result, operations such as sorting and undo are difficult to implement with these systems. While URP features a mode where the table can display a fluid flow simulation around an arbitrary physical object, the tactile nature of this interaction is one directional: While the physical object can affect the software state, the software state cannot affect the physical object. Pico supports a more generalized, bidirectional interaction with everyday physical objects in which the physical objects and software state both influence each other.

Tangible Constraints

Ullmer's work on TUIs explores a rich set of physical constraints to impart structure to physical arrangements in token systems [29]. 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"[29]. 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. In Ullmer's work, 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 [29].

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. We 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"[29]. The general concept is that users can add, remove and manipulate mechanical constraints on the tabletop to influence the way objects on the table move. The computer does not sense these mechanical 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 mathematical 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.

Actuation

Many researchers have explored force feedback in the HCI context in applications such as surgical simulation[23] and other 3D problem solving tasks. In the GROPE system[2],

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.

Tabletop Actuation

In addition to providing haptic feedback through multiple physical objects, tabletop actuation systems allow actuated objects to provide visual feedback even when they are not being touched. Several researchers have explored this idea of physical motion on a tabletop surface as part of a user interface. Fitzmaurice first proposed the idea of graspable interfaces based on "propelled bricks"[5]. The Planar Manipulator Display [24] uses a series of wireless battery powered robots that drive themselves around based on commands from a central computer. In the computer gaming context, the Augmented Coliseum[13] tracks the position of robotic toys on a tabletop and projects graphics on and around them. The Actuated Workbench uses an array of electromagnets below a position sensing antenna to move physical user interface elements on a tabletop [17]. It demonstrated that this approach could be used for haptic feedback and remote collaboration [17]. While the actuation hardware in Pico is similar to that in the Actuated Workbench, the control software is quite different. Actuated Workbench uses anti-aliasing and PID control algorithms to provide very smooth motion on the tabletop surface. In so doing, it abstracts away many of the physical properties of the objects in the interface, such as friction and mass. In contrast, Pico preserves the dynamic behaviors caused by differences in the physical objects themselves. Preserving these dynamic behaviors helps Pico support the concept of mechanical constraints, and offer the user an interaction vocabulary based on the physics of objects in the everyday world.

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" [28], Shieber points out that many types of problems can be thought of as optimization problems, such as "writing a (maximally) convincing memo, determining the (ideal) price for a product, constructing a (maximally) communicative diagram." Computers are unable to perform these kinds of tasks autonomously, so 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" [28]. OpTable [26] is one tabletop system that investigates collaborative optimization. The OpTable uses a turn-taking mechanism to mediate control and support role-division between the user and the computer. In contrast, Pico supports this role division using mechanical constraints to guide the task's global structure. The user and the computer can simultaneously influence the problem solving process, but the user can always overrule or guide the computers actions using mechanical constraints.

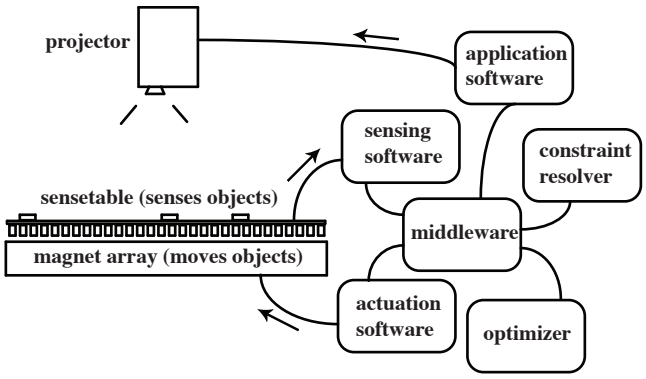


Figure 4: System architecture

While Shieber's term "Collaborative Interfaces" refers to interfaces where the computer is regarded as a collaborator with unique skills, many other interfaces have been developed where the computer instead facilitates collaboration between people. Recent examples of this approach in the context of tabletop systems are MultiSpace[4], ViCAT[3] and LumiSight[11].

Human-in-the-Loop Optimization

A variety of interactive optimization systems have been developed that aim to take advantage of the relative strengths of people and computers to solve problems that are difficult for either people or computers to solve alone. This approach has been applied to vehicle routing problems [31][26][1], graph layout problems [25] and rendering tasks[14] among others. Using a human-in-the-loop approach to optimization has been shown to perform competitively with very sophisticated optimization systems when run on benchmark problems [1]. Pico builds on this work by exploring the use of tangible objects and mechanical constraints as a method of structuring the collaboration between user and computer.

IMPLEMENTATION

The use of interaction techniques based on mechanical constraints on a tabletop user interface requires three components: a system for moving objects on the tabletop, a way to sense their position and software to make these two parts work together as part of an application. A high-level system diagram is shown in figure 4.

Actuation

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 predictable by the user. One aspect of this predictability 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 "holonomic" drives for robots that are able to move equally easily in all directions, however. The magnetic array approach used by the Actuated Workbench is also holonomic, and has the added benefit of using small, passive objects. Based on these considerations we based our actuation system on an array of electromagnets. The fully assembled array, containing 512 magnets, is shown in figure 5, and its components are shown in figure 6.

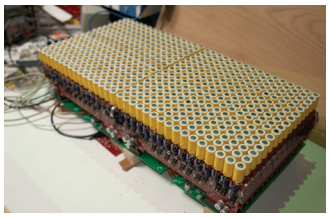


Figure 5: The Pico magnet array

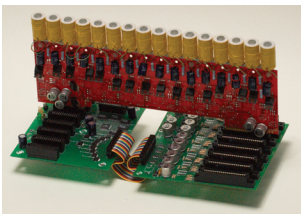


Figure 6: Cards containing magnets and driver chips plug into sockets in the logic board and the high voltage board.

The electromagnets are mounted in groups of 16 on cards, which also contain Power MOSFETs and 16 large bipolar capacitors, critically damped with the electromagnets. Each magnet has an N-channel and a P-channel MOSFET associated with 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. The MOSFETs are controlled by 35V-45V signalling inputs from the high voltage board, which serves to isolate the high voltage circuitry from the logic board.

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 system's 4 Atmel AVR microcontrollers render that frame to the magnets repeatedly until a new frame is received. The AVR refreshes the magnets at a rate of roughly 400 Hz. The overall array measures 30.5 cm by 61cm, and consumes an average of 45W of power. Peak power consumption is 225W. The array can be scaled to a larger size by adding more of the building blocks shown in figure 6. The power consumption scales linearly with the number of objects on the tabletop. It is independent of the overall size of the magnet array. The system is able to position objects with a precision of roughly 1 mm and accuracy of about 2 mm.

Position Sensing

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. Our approach is similar to that used in the Patten and Recht's Audiopad system[19].

Software

Control Loop

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 15mm away from the object's current location. This point is located inside of a square, such that the corners of that square are defined by four adjacent electromagnets. A duty cycle for each of these four magnets is calculated, such that the sum of the forces will draw the puck toward the given point. These duty cycles are sent to the control hardware, which then turns the magnets on and off appropriately. 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.

Figure 7: A rubber band used as a mechanical constraint to keep two pucks in close physical proximity.

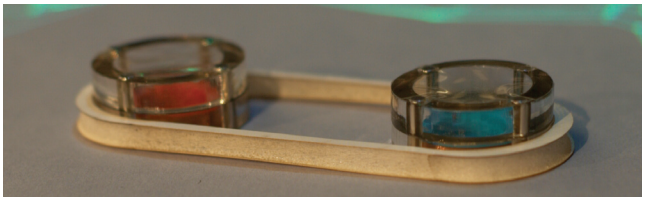


Figure 8: An oval-shaped ring used to keep two pucks within a given minimum distance to each other.

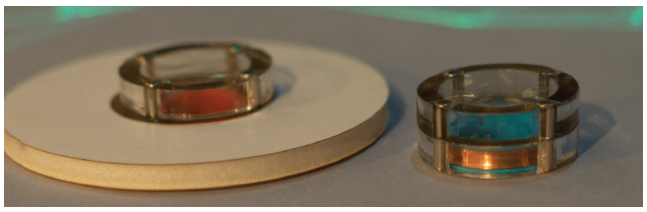


Figure 9: A collar used to enforce a minimum distance constraint.

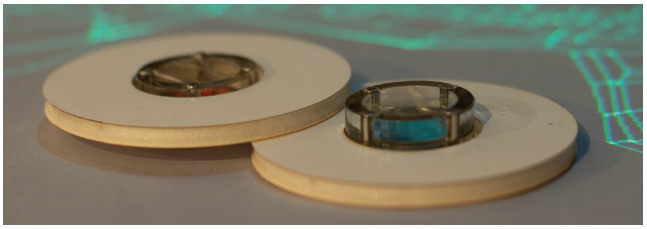


Figure 10. Collars at differing heights can yield different minimum distance constraints for different combinations of pucks.

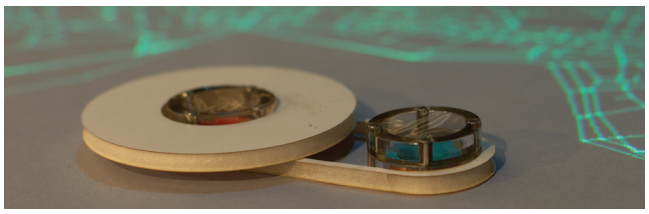


Figure 11: Minimum and maximum distance constraints can be combined.

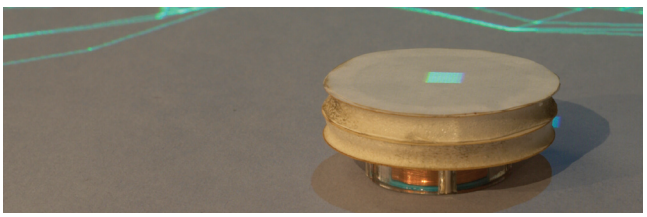


Figure 12: The object placed on top of the puck is filled with sand, preventing the computer from moving the puck autonomously.

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. 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.

Pico's motion control algorithms do not attempt to learn the positions of obstacles on the table, or change a puck's path in order to dodge obstacles, or adapt based on feedback from the sensing circuits. The control algorithms were kept as simple as possible in order to ensure that the cause-and-effect relationships governing the motion of the pucks on the table were as easy as possible for the user to understand. If the computer tries to move a puck in a way that is prevented by an obstacle, it is easy for the user to understand why the motion of the puck has stopped. If the user does not intend for that particular mechanical constraint to be enforced for that particular object, he or she can simply move the puck or the object constraining its motion as desired.

If the ultimate destination of the puck is less than 15 mm 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 that slowly moves it into position.

Other Software Modules

The "optimizer" is an application specific module that runs on a separate, powerful computer. Its performs the simulated annealing process that determines how the computer will try to move the objects on the table to solve a given optimization problem. In contrast to the OpTable[26], the optimizers we have implemented for the Pico platform do not try to learn or change strategy based on the history of user input. We wanted Pico to be useful for quickly comparing alternative problem solving strategies and we were concerned that considering user interaction history when optimizing would confound the comparison of alternative problem solving strategies.

The "constraint resolver" continuously tries to ensure that the position of each physical puck corresponds correctly with the software parameter it represents. If the physical and software representations of a parameter should become inconsistent, the constraint resolver pulls the puck toward the position indicated in software, and adjusts the position indicated in software toward the puck. It does this incrementally until the puck and its associated parameter are within 3 mm of each other. The "application software" draws application specific graphics on the tabletop using an overhead video projector, and the "middleware" provides a message bus over which the other modules communicate.

INTERACTION TECHNIQUES

The primary goal and main contribution of Pico is to make objects 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. Pico's physical feedback loop gives us all of these techniques "for free" without having to explicitly design in each one. In fact, the physical constraints on the table are not sensed by the computer in any way, other than that the computer may try to move a puck and find that it is not able to.

By translating the mathematical "forces" on parameters in an optimization process into physical forces on the corresponding pucks, the system creates the illusion that the pucks are attracted to a better solution, as if by gravity. Other interactions happen in the context of this continuous force that tries to pull the complex system of variables toward the best solution it can find.

Pucks (and their associated parameters) can be kept very close together using a rubber band, as shown in figure 7. Alternatively, if one prefers a larger maximum distance between objects, one can use a larger mechanism such as that shown in figure 8. A minimum distance constraint can be established using collar around one or more pucks, as shown in figure 9.

Collars with different elevations can be used, so that different puck combinations are mechanically constrained to different minimum distances, as in figure 10. These distance constraints can also be combined to establish a minimum and maximum at the same time, shown in figure 11.

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 2), or place some sort of weight on top of it (figure 12), 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 1. One can 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.

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 it if necessary. They serve as short term, ad hoc "jigs" to make the problem solving process easier.

While physical constraints can create a rich and flexible vocabulary, there are some types of constraints that are difficult to represent with physical objects. Pico also supports software constraints for these cases.

EVALUATION

We evaluated Pico using a simplified version of the cellphone tower layout application. 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 we wanted

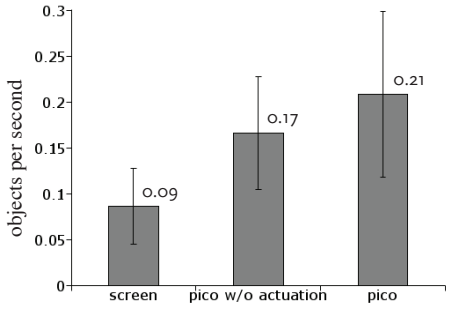


Figure 13: Number of interface objects acquired per second in the three experimental conditions. Mean and standard deviation are shown.

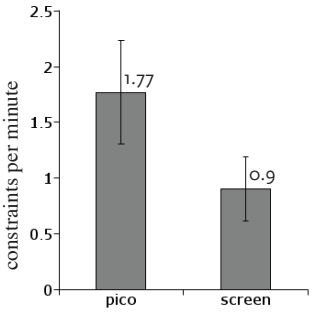


Figure 14: Number of constraints used per minute of interaction in the pico and screen conditions. Constraints were not used in the pico without actuation condition. Mean and standard deviation are shown.

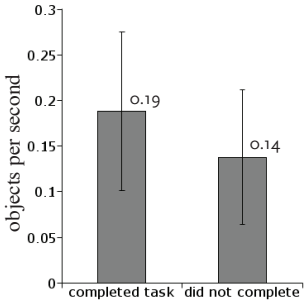


Figure 15: Number of interface objects acquired per second across all experimental conditions. In the "completed task" category are trials in which the subject was able to obtain a total layout score of 400 or greater. The "did not complete" category shows trials in which the score of 400 was not reached. Mean and standard deviation are shown.

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. The current coverage score was displayed to the user on the screen or the table (depending on the experimental condition) and the user was told that the task would terminate when the score of 400 had been reached, or the allotted time had expired. 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. We 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 (PWA), and Screen.

Screen condition: 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 parity 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 condition: 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 (PWA) condition: 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

Our hypotheses for this experiment were framed in the context of Kirsh and Maglio's distinction between "epistemic" and "pragmatic" action [12]. Epistemic action takes place when users change their environment to search for the best solution or strategy to perform a task. Pragmatic action is action taken to actually perform the task. For example, often players of the game Tetris will rapidly rotate the falling bricks while they are at the very top of the screen. This is epistemic action, and players do it to find the rotation that fits best with the bricks below. They rotate the bricks on the screen rather than mentally because rotating them on screen is faster [12]. Others have argued previously that this distinction between epistemic and pragmatic action is particularly relevant to research involving computer interfaces based on physical objects [8, 5]. Building on this idea, Sharlin et al. propose that a key characteristic of a successful TUI is that it supports this "trial-and-error activity" [27].

By the same token, we expected users of the Pico system to plan less and rely more on epistemic action, switching between alternative strategies more frequently than in the other two cases. We measured this switching by monitoring the number of times users shifted their control to a new object or objects in the interface. We 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. This yields the first hypothesis:

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

We also hypothesized that the differences 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:

H2: Users will shift their control between objects more often in the Pico condition than in the PWA condition.

Alternative mechanisms for constraining the motion of the cellphone towers are provided in the screen case and the Pico case. While these provide similar functionality, we expected users will be more likely to use them in the Pico case:

H3: Users will constrain the motion of pucks more in the Pico case than the screen case.

Results

Data was collected using several methods. The application software logged user input to a datafile for later analysis. However, in both Pico conditions (with and without actuation) it was difficult to determine what the user was doing by relying exclusively on the software logging feature. 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 under each experimental condition. The results are shown in figure 13. We found that the number of these cycles in the Pico condition was significantly higher than in the PWA condition ($p < 0.05$) and the screen condition ($p < 0.001$), supporting H1 and H2. The PWA 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 14 ($p < 0.05$), supporting H3.

The mean score on the Likert scale for the screen condition was 4.3 (std. dev. = 1.05). The mean score for the PWA condition was 5.1 (std. dev. = 1.25). The mean score for the Pico (with actuation) condition was 5.3 (std. dev. = 0.72). The only difference between these scores that reached statistical significance was the difference between the screen and pico (with actuation) conditions ($p < 0.05$).

Four subjects were able to complete the task in the screen condition, versus five in the PWA condition, and seven in the Pico condition. To see if there was a relationship between the tendency to switch objects and successfully completing the task, we 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. We 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 different cellphone towers in the interface, shown in figure 15.

Subjective Data

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 and PWA conditions.

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 said that he "wasn't sure how to benefit from the computer's input." 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.

The style of problem solving encouraged by the Pico condition worked well for some users, but was difficult for others. One possible area for future work would be to provide a way to easily disable and enable the computer's ability to actuate the pucks, such as with a foot pedal, and see how people used this pedal. For example, would users tend to leave the actuation enabled all of the time, or would they tend to periodically disable it, leading to a de facto turn-taking style of interaction? Based on the results of the present experiment, we suspect that two usage patterns would emerge: one in which the users let the computer move pucks all of the time, and another in which a turn taking style was dominant.

Video Analysis

Subjects in the Pico and PWA 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 PWA 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 support hypotheses H1, H2 and H3, and suggest that Pico makes it easier to quickly explore various potential solutions to a spatial layout problem. 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, we 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.

DISCUSSION

One of the main contributions of this work is that it supports improvisation with physical objects in the user's environment to help perform a computational 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. For example, in the cellphone tower placement application, 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. This approach 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 [10], in which the use of physical affordances and metaphors is unchangeable by the user.

Improvisation with physical objects can 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. By appropriating everyday physical objects as interface elements, this type of reprogramming is not necessary. 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.

This approach encourages accidental discoveries by making 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.

CONCLUSION

This paper has presented a set of novel interaction techniques that use mechanical constraints as computational constraints. These constraints leverage users' mechanical intuition about the behavior of objects in the physical world. The ability to use mechanical constraints encourages improvisation with common physical objects as tools to quickly try out a problem solving strategy. These constraints allow users to focus on the high-level structure of a problem while the computer optimizes the details. Subjects in our user study preferred Pico and were more successful with it in a complex spatial layout task than with other interfaces.

While the applications implemented on the Pico system to date have been spatial in nature, there are many applications that do not have literal interpretations of space that could be mapped to Pico's interaction space. 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.

Pico points to opportunities for a larger degree of improvisation with everyday physical objects in the context of human-computer interfaces. We believe that Pico is just the beginning, merely a first exploration of this idea within the larger context of interaction design. One can imagine interfaces where mechanical actuation is incorporated into many different types of interfaces that are more three dimensional in nature, creating free-form, improvisational interfaces that encourage experimentation, helping users make discoveries and change perspectives.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their helpful comments. We would also like to thank Rob Jacob, John Maeda, Jason Alonso, Gian Pangaro, Dan Maynes-Aminzade, Vincent Leclerc, the students in the Tangible Media Group, and Phil Fleming for valuable discussions regarding the work. This work was supported by the Things that Think and Digital Life consortia of the MIT Media Lab.

REFERENCES

1. Anderson, D., Anderson, E., Lesh, N., Marks, J., Mirtich, B., Ratajczak, D., and Ryall, K. Human-guided Simple Search. Proc. of AAAI 2000, AAAI Press (2000), pp. 209-216.

2. Brooks, F., Ouh-Young, M., Batter, J., Kilpatrick, P. Project GROPE: Haptic Displays for Scientific Visualization. In Proceedings of SIGGRAPH '90, ACM Press (1990), pp. 177-186.

3. Chen, F., Close, B., Eades, P., Epps, J., Hutterer, P., Lichman, S., Takatsuka, M., Thomas, B., Wu, M. ViCAT: Visualization and Interaction on a Collaborative Access Table. In Proceedings of IEEE Tabletop 2006, IEEE Publisher (2006).

4. Everitt, K., Shen, C., Ryall, K., Forlines, C. MultiSpace: Enabling Electronic Document Micro-Mobility in table-centric, multi-device environments. In Proceedings of IEEE Tabletop 2006, IEEE Publisher (2006).

5. Fitzmaurice G. Graspable User Interfaces. Ph.D. Thesis. Department of Computer Science University of Toronto. 1996.

6. Fjeld, M., F. Voorhorst, M. Bichsel, H. Krueger, and M. Rauterberg (2000): Navigation Methods for an Augmented Reality System (video). In Extended Abstracts of ACM CHI '00. ACM Press (2000), pp. 8–9.

7. Fjeld, M., Lauche, K., Bichsel, M., Voorhorst, F., Krueger, H. and Rauterberg, M. Physical and Virtual Tools: Activity Theory Applied to the Design of Groupware. In B. A. Nardi & D. F. Redmiles (eds.) A Special Issue of Computer Supported Cooperative Work (CSCW): Activity Theory and the Practice of Design (2002), Volume 11 (1-2), pp. 153-180.

8. Fjeld, M., Schär, S., Signorello, D., and Krueger, H. Alternative Tools for Tangible Interaction: A Usability Evaluation. Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR) 2002, pp. 157-166.

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

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

11. Kakehi, Y., Hosomi, T., Iida, M., Naemura, T., Matushita, M., Transparent Tabletop Interface for Multiple Users on Lumisight Table., In Proceedings of IEEE Tabletop 2006, IEEE Publisher (2006).

12. Kirsh, D. and Maglio, P. On Distinguishing Epistemic from Pragmatic Actions. Cognitive Science. (1995).

13. Kojima, M., Sugimoto, M., Nakamura, A., Tomita, M., Nii, H., and Inami, M. Augmented Coliseum: An Augmented Game Environment with Small Vehicles. Horizontal Interactive Human-Computer Systems, 2006. IEEE TableTop 2006. IEEE Publisher (2006).

14. Marks, J. et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation, Proceedings of ACM SIGGRAPH '97, ACM Press (1997) page 389-400.

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

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

17. 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), ACM Press (2002).

18. Patten, J., Ishii, H., Hines, J., and Pangaro, G. Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces. In Proceedings of ACM SIGCHI '01. ACM Press (2001), New York, NY, 253-260.

19. 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), ACM Press (2002).

20. Piper, B., Ratti, C., and Ishii, H. Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis, in Proceedings of ACM CHI '02, ACM Press (2002), pp. 355-362.

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

22. Richard, C., and Cutkosky, M., The Effects of Real and Computer Generated Friction on Human Performance in a Targeting Task, Proceedings of the ASME Dynamic Systems and Control Division, vol. 69-2, (2000), pp. 1101-1108.

23. Rosen, J., Hannaford, B., MacFarlane, M., and Sinanan, M. Force Controlled and Teleoperated Endoscopic Grasper for Minimally Invasive Surgery - Experimental Performance Evaluation, IEEE Transactions on Biomedical Engineering, vol. 46, no. 10, IEEE Publisher (1999). pp. 1212-1221.

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

25. Ryall, K., Marks, J., Shieber, S., An Interactive Constraint-Based System for Drawing Graphs. Mitsubishi Electric Research Laboratory technical report TR-97-15.

26. Scott, S.D., Lesh, N., and Klau, G.W. Investigating human-computer optimization. Proceedings of CHI'02, pp. 155 - 162.

27. Sharlin, E., Watson, B.A., Kitamura, Y., Kishino, F., and Itoh, Y. On humans, spatiality and tangible user interfaces. Pervasive and Ubiquitous Computing, 8, 338-346. Theme issue on Tangible Interfaces in Perspective. 2004.

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

29. Ullmer, B., Ishii, H., Jacob, R. J. K., Token and Constraint Systems for Tangible Interaction with Digital Information. ACM Transactions on Computer-Human Interaction, Vol. 12, No. 1, March 2005, ACM Press (2005), pp. 81–118.

30. Underkoffler, J. and Ishii, H. Urp: a luminous-tangible workbench for urban planning and design. Proceedings of CHI'99, ACM Press (1999), pp. 386 - 393.

31. Waters, C.D.J. Interactive Vehicle Routing. Journal of the Operational Research Society, 35(9), (1984), pp. 821-826.