Tangible Query Interfaces:
Physically Constrained Tokens for Manipulating Database Queries

Brygg Ullmer[†], Hiroshi Ishii, and Robert J.K. Jacob[††]
MIT Media Laboratory, One Cambridge Center, 5FL, Cambridge, MA USA
{ullmer, ishii, rjacob}@media.mit.edu

Abstract: We present a new approach for using physically constrained tokens to express, manipulate, and visualize parameterized database queries. This method extends tangible interfaces to enable interaction with large aggregates of information. We describe two interface prototypes that use physical tokens to represent database parameters. These tokens are manipulated upon physical constraints, which map compositions of tokens onto interpretations including database queries, views, and Boolean operations. We propose a framework for "token + constraint" interfaces, and compare one of our prototypes with a comparable graphical interface in a preliminary user study.

Keywords: tangible interfaces, dynamic queries, databases

1 Introduction

A growing number of tangible user interfaces (TUIs) have worked to give physical form to digital information. Most TUIs make a direct "one-to-one" mapping between physical objects and elements of digital information. However, physical world pragmatics can limit the scalability of this approach in several respects.

First, this approach can limit the number of information elements a TUI can practically be used to manipulate. While spreadsheets and databases of hundreds, thousands, or more digital elements are common, manipulating such numbers of discrete physical elements might often become burdensome. Second, one-to-one mappings of physical objects to data elements can also limit the kinds of operations a TUI can support. While digital operations over large aggregates of information are common – e.g., query, sort, group, etc. – such operations may be difficult to express, view, and build upon if data elements are individually embodied.

Instead of using physical objects to directly represent individual information elements, we use physical objects to indirectly reference information by representing expressions that hold over large aggregates of information. Specifically, we use physical tokens to represent database parameters. Placing these tokens within "query racks" expresses queries composed of the corresponding parameters, and invokes visualizations of the parameter distributions. The physical manipulation of these tokens modifies parameter thresholds, expresses Boolean relationships, and controls visualizations of query results.

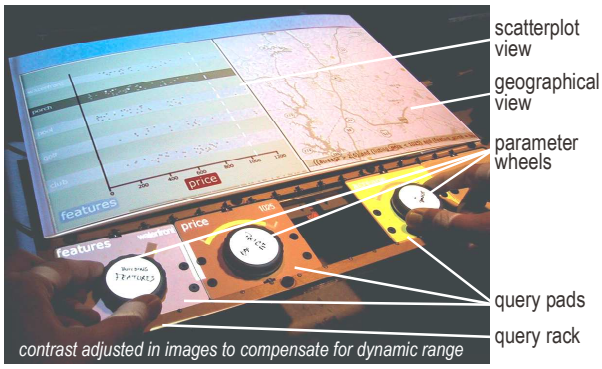

Figure 1: Parameter wheels and visualizations.

We have implemented two prototypes of these "tangible query interfaces." We believe our approach extends tangible interfaces to leverage computers' capabilities for processing large aggregates of digital information, while preserving the benefits of TUIs such as support for two-handed interaction, colocated collaboration, and provision of strong physical and cognitive affordances.

We begin with an overview of our interfaces' functions. We propose a framework for "token + constraint" TUIs that describes both our interface and earlier systems, and compare one of our prototypes alongside a GUI in a preliminary user study.

[†] Current affiliation: Visualization Dept., Zuse Institute Berlin (ZIB), ullmer@zib.de
[††] Current affiliation: Computer Science Dept., Tufts University, jacob@cs.tufts.edu

2 Functionality Overview

We begin by describing the use of our query interface prototypes for a real estate application. These build upon ideas first developed in the GUI "Dynamic Homefinder" prototype illustrating dynamic queries (Williamson and Shneiderman, 1992).

2.1 Parameter wheels

In the first example, "parameter wheels" are used to explore homes in a real estate database using our system (Figures 1, 2, 3). These wheels are small cylindrical tokens embedded with RFID tags and faced with cardstock labels. Nine parameter wheels are present, each representing fields of the database. Six wheels represent continuous parameters like price and acreage (hectares). Three wheels represent discrete parameters like building types and features.

These parameter wheels are used within a "query rack" made up of a series of "query pads," each with a receptacle for a parameter wheel (Figures 1, 2, 3). Placing a wheel upon a pad expresses the associated parameter as part of the active query.

A display surface is located adjacent to the query rack. Two visualizations appear on this surface: geographical and scatterplot views. A projector illuminates both the display surface and the query rack, including its embedded query pads. The two query pads on the left are adjacent to and associated with the 'X' and 'Y' axes of the scatterplot. The right pads offer space for additional parameters.

Figure 2: Query rack and pads.
The two leftmost pads map to the scatterplot's Y and X axes, respectively. Here, the Y axis pad is empty, while the X axis pad contains a parameter wheel.

An example interaction might begin by picking up the "price" parameter wheel and placing it upon the "X axis" query pad. In response, the "price" label and value range are illuminated on the query pad surrounding the wheel, and a 1D plot of price appears on the scatterplot (Figure 3a). The locations of all of the homes meeting these criteria are also displayed on the geographical view. The "price" wheel initially specifies the parameter range spanning from the least expensive homes to the middle-cost homes. The upper bound can be adjusted by rotating the wheel within the query pad. The query pad, scatterplot, and geographical views update correspondingly. The two leftmost pads map to the scatterplot's Y and X axes. Here, the Y axis pad is empty, while the X axis pad contains a parameter wheel.

To add a second parameter criterion to the query, an "acreage" wheel is placed upon the Y axis query pad. The Y axis of the scatterplot updates accordingly, yielding a 2D plot of acreage against price. To identify (e.g.) low-priced houses sited upon relatively large properties, the user can manipulate both wheels using his two hands. The scatterplot indicates available prospects, while the geographical view indicates their corresponding locations (here, on the periphery of the city). These parameters are implicitly joined with a Boolean AND relation.

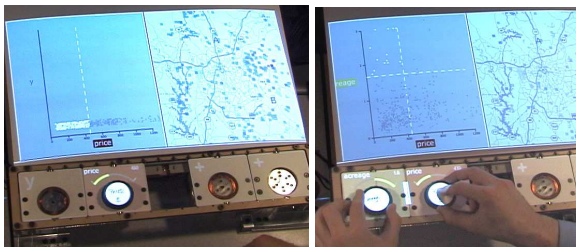

Figure 3a,b: Manipulation of parameter wheels.

In some cases it is desirable to spot trends in the data. Such patterns are not immediately visible with the price and acreage pairing. Replacing the acreage token with the "square footage" wheel, a clear correlation becomes visible in the scatterplot view.

Parameter wheels remain persistently bound to their associated value ranges when moved to or from the query rack. This eases change of view (e.g., swapping scatterplot axes), and simplifies queries involving a third or fourth parameter. For example, the acreage wheel can be returned to the query rack alongside the price and square footage wheels. While the third token is not separately represented on the scatterplot, its impact is shown through highlighting within the geographical and scatterplot views.

Several wheels are associated with discrete parameters. For instance, one wheel is associated with different building types. Turning this wheel to select "patio homes" shows clustering in certain areas of the city. Similarly, selecting "mobile homes" exposes locations on the city's periphery. The discrete-valued parameter wheels can easily be combined with continuous-valued wheels. For example, placing the "building type" and "price" wheels upon the X and Y axes shows clusterings of prices associated with different housing types (Figure 4a).

When price is replaced with acreage or square footage, different patterns are visible.

A second discrete-valued parameter wheel selects for area high schools. As expected, this parameter shows strong clustering corresponding to different school districts. A final discrete-valued wheel selects for building "features;" e.g., waterfront proximity. This parameter also illustrates clustering around lakes and other geographical features.

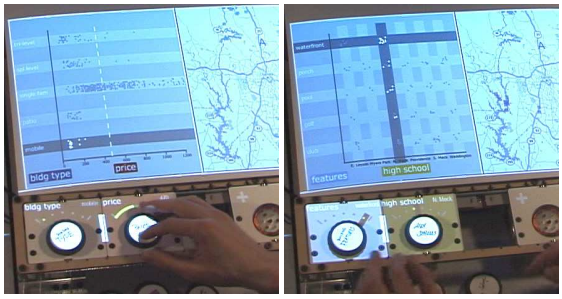

Figure 4a,b: Scatterplot views composed of discrete + continuous and two discrete parameter wheels.

Finally, discrete-valued parameter wheels may be combined with each other. Figure 4a illustrates the intersection of building features with school districts. Here, districts with waterfront homes are visible.

2.2 Parameter bars

Our second prototype uses "parameter bars" to provide another approach for expressing queries. While parameter wheels are faced with passive labels, parameter bars are embedded with active displays. These displays indicate the identity of the active parameter and a value histogram. Parameter bars can be dynamically bound to new parameters by placing them near binding points on a GUI monitor, with their internal displays updating accordingly.

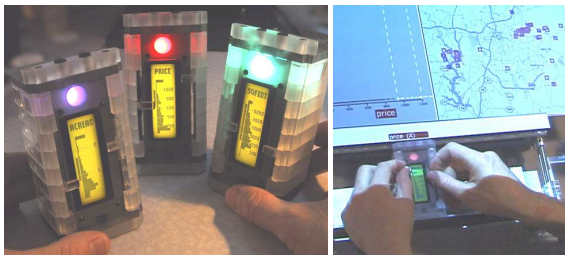

Figure 5: Parameter bars; clustering of costly homes.

Parameter bars are embedded with double sliders, allowing the modification of both the upper and lower bounds of a target parameter range. The combination of embedded displays and manipulators allows parameter wheels to be reconfigured while away from the query rack. This is potentially useful in group meetings, among other contexts.

As an example, a parameter bar representing the price of real estate properties can be placed onto a query rack. As with the parameter wheels, corresponding scatterplot and geographical results are displayed. Unlike parameter wheels, both the lower and upper bounds of the price distribution can be controlled. This supports the identification of patterns such as spatial clusterings of high-priced homes.

A second parameter bar can be added to the query rack. When these bars are adjacent, a Boolean "AND" operation is applied, as in the case of parameter wheels. When the parameter bars are spatially separated on the rack (which is detented to support stable positioning and haptic feedback), an "OR" operation is instead applied (Figure 6).

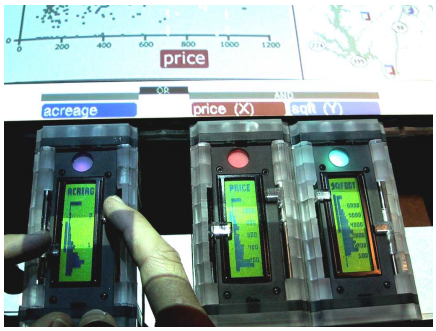

Figure 6: Boolean relations between parameter bars.

The "OR" operation has special value for comparing the distributions of different parameters. For example, when an "OR" relation between high-priced and high-acreage homes is displayed, the original "price" clustering is visible alongside the distribution associated with the acreage parameter.

Where this visualization has meaning in the real estate domain, it takes on special value for other kinds of datasets; e.g., to mutual fund databases. Here, it is valuable to compare the one-year and ten-year returns for multiple funds. It is desirable to simultaneously view both distributions, which is facilitated by the "OR" relation and visualization. As particular funds are identified, the "AND" conjunction aids identification of this relationship with other variables (e.g., risk assessment).

3 Token + Constraint Approach

We now consider how these new interfaces fit into the larger space of tangible interfaces. Much of the TUI design space can be divided into several high level approaches. In the interactive surfaces paradigm, physical objects are manipulated upon an augmented workbench or wall (e.g., Wellner, 1993; Rauterberg et al., 1997; Underkoffler and Ishii 1999). The constructive assemblies approach draws inspiration from LEGO™ and building blocks, building upon the interconnection of modular physical elements (e.g., Aish et al., 2001). These are illustrated in Figure 7.

A third, less populated approach can be described as "tokens + constraints." In our interpretation, tokens are discrete, spatially reconfigurable physical objects that represent digital information or operations. Constraints are confining regions within which tokens can be placed. Constraints are mapped to digital operations or properties that are applied to tokens placed within their confines. Constraints are often embodied as physical structures that mechanically channel how tokens can be manipulated, often limiting their movement to a single physical dimension. Alternately, constraints can be visually expressed without a mechanically defining perimeter, as with the cells found in many board games.

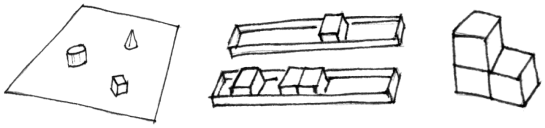

Figure 7a,b,c: Major TUI approaches: interactive surfaces, tokens+constraints, constructive assemblies.

This paper focuses on the use of physical, mechanically confining constraints within tangible interfaces. The manipulation of tokens within these constraints – token entrance, exit, translation, and rotation – is mapped to a variety of computational interpretations. Taken separately, tokens and constraints are not individually "actionable." Combined together, tokens and constraints represent fully formed, manipulable computational expressions.

Token+constraint TUIs offer a kind of middle ground between interactive surfaces and constructive assemblies (Figure 7b). As discussed in related terms within (Maclean et al. 2000), token + constraint interfaces offer a balance between continuous and discrete styles of manipulation. Interactive surfaces TUIs have usually emphasized continuous styles of interaction, framing interaction in terms of continuous positions and orientations of physical tokens. Alternately, TUIs employing constructive assemblies emphasize discrete relationships between physical objects, generally framed in terms of connection, disconnection, and topology. In contrast, token + constraint systems lend themselves to supporting both continuous and discrete forms of manipulation.

These discrete and continuous forms of manipulation occur within two distinct phases of interaction: *associate* and *manipulate* (Figure 8). In the associate phase, tokens are associated with specific constraints. This is done by placing the token within the physical confines of the constraint. This action establishes a (discrete) physical relationship between the token and constraint, and a computational relationship between the associated digital mappings. In the second phase, tokens are continuously manipulated within the constraints' confines, and interpreted with respect to the constraint and/or other tokens.

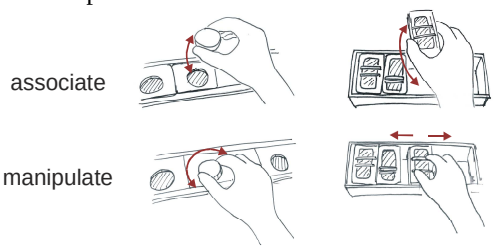

Figure 8: Phases of interaction with tokens + constraints.

4 Token + Constraint Mappings

The token+constraint approach gives physical form not only to digital information, but also to aspects of the "syntax" for combining these physical/digital elements together. Physical constraints help to enforce consistency by mechanically restricting the physical relationships that objects can express. While not eliminating the possibility of meaningless expressions, token+constraint systems physically express to users something about the kinds of interactions the interface can (and cannot) support.

In the tangible query interfaces, four kinds of operations are associated with the query rack constraints: query, view, selection+assignment, and Boolean operations. These correspond to the following physical/digital mappings:

1) physical presence → query parameter assertion
2) physical placement → view selection
3) physical rotation → parameter value selection
4) physical adjacency → Boolean operation

The query and view operations are both invoked during the associate phase of interaction, while the selection and Boolean operations are invoked during the manipulate phase. To consider the query operation, the act of placing a token upon a query rack invokes a "select… where…" operation in SQL (the most common database query language). For example, if a price token is placed on a query rack that is associated with a real estate database, a query like:

```
select bldg_id where (price > [price.min] AND price < [price.max])
```

is evaluated. If multiple tokens are on the rack, associated parameters are used as "where" operands.

The query interface mappings are specific instances of a broader family of possible token+constraint mappings. These are summarized in Figure 9. The presence relationship is usually expressed in the associate phase of interaction, while other relation-

ships are often expressed in the manipulate phase. Shaded elements are used in the query interfaces.

These relationships and mappings illustrate the range of digital operations that can be expressed by

Physical relationships	Interaction Event	Digital interpretations
Presence	Add/Remove	Logical assertion; activation; binding
Position	Translate/Rotate	Geometric; indexing; scalar
Sequence	Order change	Sequencing; query ordering
Proximity	Prox. change	Relationship strength (e.g., fuzzy set)
├Connection	Connect/Discon.	Logical flow; scope of influence
└Adjacency	Adjacent/NAdj.	Booleans; axes; other paired relations

Figure 9: Grammars for mapping token+constraint compositions to digital interpretations.

token+constraint approaches. The same relationships also can be expressed upon interactive surface TUIs, which usually possess a superset of the physical degrees of freedom of physically structured approaches. However, the use of physical constraints offers a number of benefits, including:

1) increased passive haptic feedback;
2) increased prospects for active force feedback;
3) decreased demands for visual attention;
4) increased kinesthetic awareness;
5) increased prospects for embedded uses; and
6) flexible, widely accessible sensing technologies.

5 Related Work

Tangible query interfaces broadly involve the physical modeling of logical relationships. We share some of the goals of architectural interfaces begun by Aish and Frazer in the late 1970s (Aish et al., 2001). Aish believed that physical/digital tools might help people to communicate, negotiate, and explore alternatives in face-to-face contexts. We share this optimism, and extend support for abstract information.

Several systems have developed interfaces for physically expressing software programs; e.g., (Perlman, 1976; Suzuki and Kato, 1993). Where these systems physically represented elements of procedural or functional languages, we have followed a declarative model, mapping object configurations to SQL expressions that are continuously evaluated. We believe this extends the expressiveness achievable with a small number of objects.

Among recent tangible interface research, we build upon the mediaBlocks (Ullmer et al., 1998), LogJam (Cohen et al., 1999), and ToonTown (Singer et al., 1999) systems, which all drew inspiration from the work of Bishop (Polynor, 1995). The mediaBlocks authors also suggested (but did not develop) the application of adjacency-based mappings to database queries and Booleans in (Ullmer et al., 1998).

The Navigational Blocks (Camerata et al., 2002) also developed a TUI for interaction with databases. The system represents categories of a history application with the faces of physical cubes. Our query interfaces offer new support for continuous parameters, view descriptions, Boolean 'OR' operations, and dynamic binding, among other advantages.

Our interface also has similarities to the DataTiles system (Rekimoto et al., 2001). DataTiles used transparent tiles to represent modular software elements, including a parameter tile for simple queries. DataTiles relied upon pen-based interaction with underlying GUI applets, which contrasts with our emphasis on physical representation and manipulation.

Parameter wheels also share common ground with the "tagged handles" of (Maclean et al., 2000). Here, RFID-tagged objects represent content such as digital video sequences, and mate with force feedback docks to provide haptic cues. This effort is highly complementary to our query interfaces.

Our work builds directly on the dynamic query techniques of (Williamson and Shneiderman, 1992). Several other GUIs have introduced techniques for expressing Boolean relations within database queries (e.g., Fishkin and Stone, 1995; Jones, 1998). Our approach is also related to research on visual query systems (or VQS), and has similarities to icon-based VQS systems (Catarci et al., 1997).

More broadly, our query interfaces relate to the area of visual programming. Our interfaces provide workspaces where each physical action brings an immediate interpretation and response by the system. In this respect, our approach closely follows Shneiderman's principles of "direct manipulation."

6 Implementation

Most of the systems' mechanical fabrication was executed on a Universal Laser Systems 100 watt CO_2 laser cutter, controlled by the CorelDRAW™ drawing program. Circuit boards were fabricated in-house with a Roland Modela mini-mill, and designed with TechSoft's PCB software.

The parameter wheels are embedded with Philips HiTag2 RFID tags, and sensed with an IB Technology reader multiplexed across four sensing coils. Wheel rotation is sensed by a rotary potentiometer. Sensing of the parameter bar levers is monitored by slide potentiometers. The parameter bar displays use tricolor LEDs and 120x32-pixel Seetron backlit LCD displays. These components are controlled with embedded Microchip PIC 16F876 microcontrollers and programmed with the CCS C compiler. Power is provided by rechargeable NiMH batteries. Query racks are linked by RS232 serial cable to a host PC.

The parameter bars used a custom near-field inductive communication scheme inspired by the "Beads" of (Resnick et al., 1998). However, relia-

bility and speed were problematic. Dual-ported RFID transponders would likely have been a better solution.

Projection was via a small 1024x768 pixel video projector, oriented via a desk-mounted mirror jig. The main software was written in Java and run on a two-processor PC. The database was hosted on a Linux-based PostgreSQL server.

7 Preliminary User Study

To gain user feedback on our approach, we conducted a preliminary user study comparing the parameter wheels query interface with a GUI-based dynamic queries interface. The study domain and task were loosely modeled after the "HomeFinder" experiment of (Williamson and Shneiderman, 1992), although were conducted more informally. This earlier study compared dynamic queries with text-based query interfaces, and found both speed increases and user preferences for the GUI technique.

We believe that tangible query interfaces can provide strong support for exploratory interaction with data. To help support this claim, our experiment explored three (informally framed) hypotheses:

1) *Tangible interfaces using physically constrained tokens can provide a feasible approach for expressing simple queries.* Since tangible query interfaces are a new querying approach, usability claims benefit from verification through the user experience.

2) *TUIs elicit parallel two-handed interactions within querying tasks.* While support for two handed interactions has been a frequent claim for TUIs, it was not clear whether people would in practice use both hands to control the TUI querying task.

3) *TUI is faster than GUI for a range of querying tasks.* We also believed that our TUI would be quantitatively faster than comparable GUIs based upon the dynamic queries approach. While we were more interested in TUIs potential for contexts such as colocated collaboration, single-user performance seemed the cleanest metric for an initial comparison.

6.1 Experimental Setup and Task

Figures 10 and 11 illustrate the experimental setup. At the top of the display surfaces, we asked users to express queries involving two to four continuous parameters, drawing from a pool of six parameters. These were manipulated using both the four-cell parameter wheels query rack and a range slider-based GUI (Williamson and Shneiderman, 1992). For the TUI, this ensured the need for users to spatially reconfigure parameter wheels.

The experimental tasks required users to balance between multiple competing criteria. The satisfaction of these criteria was quantified with a simple scoring algorithm, and compared with a "target score" that must be satisfied to complete the task. Current and target scores were displayed as graphical bars in the upper right of the display surfaces.

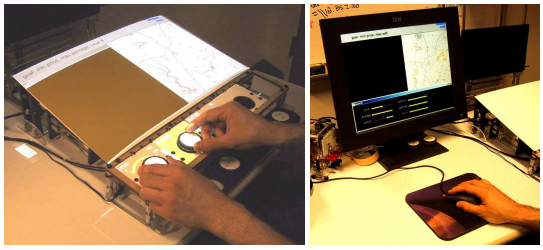

Figure 10: TUI, GUI task setups.

Figure 11: Parameter wheel, GUI range slider settings.

We decided to remove the "scatterplot" feature for our study. Since this was the primary TUI visualization, this was a difficult decision. However, it was unclear how to provide similar support for axis selection using existing GUI techniques.

Our experiment included 24 individual tasks, with the TUI and GUI reset after each task. We staggered interaction with the TUI and GUI in each session, and conducted the first half of the experiment as training tasks. We had 16 subjects, 9 male and 7 female, and used a counterbalanced, within-subjects design. Subjects were from outside our department.

6.2 Experimental Results

The tangible interface performed well in our experiment. However, we encountered two unexpected issues. First, we observed that users interacted with the TUI and GUI interfaces in qualitatively different ways. With the TUI, users almost always began with a "setup phase" (corresponding to the "associate" phase), bringing all necessary tokens onto the query rack before manipulating individual thresholds. This "setup time" made a major performance impact – on the order of 30% of overall task completion time.

Secondly, to our surprise, a few tasks appeared more difficult to complete on the TUI than the GUI. In these tasks, the data distributions required users to configure parameters to values substantially outside of the requested range. TUI users tended to keep their eyes on the score bar, and were often trapped in "local minima." In contrast, GUI users were forced to constantly look at the parameter values. While many users complained about this GUI aspect, it appeared beneficial in this case. We believe that our

scatterplot visualization, and also haptic feedback indicating parameter bounds and density, would have helped TUI users with this problem.

In raw results, the average GUI task completion time was faster than for TUI, but without statistical significance. TUI performance was substantially slowed by the setup times and by the two unusual tasks. Setup time slowed TUI performance by 30%, while the two unusual tasks slowed the cumulative average time by 40%.

In responses from the user surveys, user preferences were split: 8 users preferred the TUI, and 7 preferred the GUI. The average preference was 4.5 on a 7 point scale, weakly favoring the TUI. The preference histogram followed a bimodal distribution. Seven of the sixteen users had a moderate or strong preference for the tangible interface, while all but one users favoring the GUI had a weak preference. Also, subjects used only the geographical visualization; several were shown the scatterplot visualization after the study, and all responded with enthusiasm.

Subjects who preferred the tangible interface felt that the TUI was faster, and vice versa. The users ranged from 19 to 45 years of age, averaging roughly 27. The TUI was more popular with younger users (averaging 23, vs. 30 for users preferring GUIs). Interestingly, of the subjects who rated themselves in the top two tiers of computer expertise, more than half preferred the TUI.

We also asked users about the interfaces' ease of learning, ease of use, and their likelihood to support effective interaction with real databases. On average, users rated the TUI more highly on each count. We were pleased by the "effectiveness" result, given previous success of the GUI method.

Returning to the original study hypotheses:

(1) The feasibility hypothesis was confirmed. Sixteen users completed 189 of the 192 TUI tasks.
(2) The two-handed hypothesis was confirmed. 80% of the users used both hands. More than 40% made unprompted mention of simultaneous two-handed manipulation as a major strength.
(3) The performance hypothesis was not confirmed. But on average, users preferred using the TUI.

8 Discussion

8.1 Comparison with graphical interfaces

One of the most basic questions about our approach is "why not use a GUI?" Graphical interfaces can support all of our system's abstract functionality. Also, textual and graphical query interfaces are clearly preferable in many contexts.

We believed the TUI would help users focus upon the "objects of interest" – in this case, the parameters of the query task. The TUI arguably offers more "direct" manipulation than the GUI, allowing better use of kinesthesia, with eyes focused on the scoring results. We also expected that parallel two-handed manipulation of parameter wheels would contribute toward a TUI performance increase.

In practice, while the TUI met with positive user feedback, the quantitative performance comparison with the GUI was inconclusive. The TUI performance shortcomings in the two "unusual tasks" may reflect the importance of tighter integration of parameter tokens with the results display, including use of haptic feedback techniques.

The impact of the setup/associate phase has several implications. First, inclusion of the scatterplot within the study would likely have benefited TUI performance. Second, most GUIs also require a "setup phase" in which active parameters are determined and views are defined. GUIs often afford representation of more parameters than TUIs, leading to our study design. However, including a setup phase for GUI tasks would be a more equal comparison, and would lead to stronger TUI results.

8.2 Mapping and integration alternatives

One of the largest challenges for TUIs is the design of strong physical/digital mappings. For the query interfaces, three such issues were particularly evident: view composition, query composition, and the integration of physical and graphical elements.

Our parameter wheels used a fixed mapping between pads and scatterplot axes, while the parameter bars' mapping was based on token ordering. We felt the fixed mappings worked quite well. While the order-based mapping functioned, we felt it was a weaker approach. This was partly because the wheel rack's fixed pads simplified rapid composition of views; and partly because the parameter bar mapping was overloaded with the Boolean interpretation.

We believe the adjacency-based Boolean mapping is potentially valuable, especially for contexts like our mutual fund example. However, its utility depends upon the supporting visualizations. Often, the simpler AND-only mapping may be preferable.

Our current results display is visually biased and somewhat segregated from the rest of the TUI. One path for improvement could be to combine our interface with other TUIs with strong inherent visual mappings. For example, our interfaces could query census information in combination with the Urp urban planning simulator (Underkoffler and Ishii, 1999), displaying query results directly onto Urp's graphical workbench. Here, our racks could offer a kind of TUI "widget" as an element of more complex TUIs.

8.3 Scope of database functionality

Our query interfaces support the common "select-from-where" form of SQL queries, including parameter selection, range selection, Boolean operations, and view description. While a small subset of the full SQL language, we believe this is sufficient for meaningful interaction in a number of content domains. It is also a superset of dynamic queries and other prior query approaches.

We have also developed a "dataset container" that is used to reference source datasets, and to save and compare the results of database queries. These functions seem important for further development.

Another important database operation is the "join" operation. Our system internally joins parameters from different tables following the "universal relations" approach. We have also considered alternatives for interactive joins by embodying "views" as physical objects, and expressing visual joins through the stacking of view objects.

Parameter tokens can also be encoded with cryptographic IDs, giving them interesting potential for interactions involving sensitive information (e.g., in meetings between competing organizations).

8 Conclusion

We have presented a system for physically expressing and manipulating parameterized database queries. Our approach builds upon physical objects that represent digital parameters, rather than specific data. We have shown how these tokens can be manipulated to bind them to parameters; to assert these parameters as parts of queries; to change parameter value ranges; to express Boolean relations; and to describe views of query results.

Where previous tangible interfaces have developed "containers" for specific data elements, our parameter tokens describe relations and logical constraints that are computed over large sets of information. We believe this physical embodiment of declarative expressions scales to support interaction with large aggregates of information.

We believe these approaches are relevant not only to the broad space of database applications, but also to other tasks that involve the manipulation of information aggregates and the modeling of abstract relationships. These include physical and behavioral simulations, the configuration of complex systems, and information visualization.

Acknowledgements

We thank many people for their support, including Zachary Malchano, Anna Lee, James Patten, Jennifer Yoon, Nicholas Fahey, Axel Kilian, Dan Maynes-Aminzade, Gian Pangaro, Hannes Vilhjálmsson, Lisa Lieberson, and the study subjects. We thank the MIT Media Lab's TTT and Digital Life for support of this research, and ZIB and the EC GridLab project for subsequent support.

References

Aish, R., Frankel, J., Frazer, J., Patera, A., and Marks, J. (2001). Computational Construction Kits for Geometric Modeling and Design. In *Proc. of I3D'01*, pp. 125-128.

Camerata, K., Do, E., et al. 2002. Navigational Blocks: Tangible Navigation of Digital Information. In *Extended Abstracts of CHI'02*, pp. 752-753.

Catarci, T., Costabile, M., et al. 1997. Visual Query Systems for Databases: A Survey. In *Journal of Visual Languages and Computing*, 8(2), 1997, pp. 215-260.

Cohen, J., Withgott, M., and Piernot, P. 1999. Logjam: A Tangible Multi-Person Interface for Video Logging. In *Proc. of CHI'99*, pp. 128-135.

Fishkin, K. and Stone, M. 1995. Enhanced Dynamic Queries via Movable Filters. In *Proc. of CHI'95*, pp. 415-420.

Jones, S. 1998. Graphical Query Specification and Dynamic Results Preview for a Digital Library. In *Proc. of UIST'98*, pp. 143-151.

MacLean, K., Snibbe, S., and Levin, G. 2000. Tagged Handles: Merging Discrete and Continuous Manual Control. In *Proc. of CHI'00*, pp. 225-232.

Perlman, R. 1976. Using Computer Technology to Provide a Creative Learning Environment for Preschool Children. MIT Logo Memo #24, 1976.

Polynor, R. 1995. The Hand That Rocks the Cradle. *I.D.*, May/June 1995, pp. 60-65.

Rauterberg, M., Fjeld, M., et al. 1997. BUILD-IT: a computer vision-based interaction technique for a planning tool. In *Proc. of HCI'97*, pp. 303-314.

Rekimoto, J., Ullmer, B., and Oba, H. 2001. DataTiles: A Modular Platform for Mixed Physical and Graphical Interactions. In *Proc. of CHI'01*, pp. 269-276.

Resnick, M., Martin, F., et al. 1998. Digital Manipulatives: New Toys to Think With. In *Proc. of CHI'98.*

Singer, A., Hindus, D., et al. 1999. Tangible Progress: Less is More in Somewire Audio Spaces. In *Proc. of CHI'99*, pp.104-111.

Suzuki, H. and Kato, H. 1993. AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning. In *Proc. of 4th ELC*, 1993, pp. 297-303.

Ullmer, B., Ishii, H., and Glas, D. 1998. mediaBlocks: Physical Containers, Transports, and Controls for Online Media. In *Proc. of SIGGRAPH'98*, pp.379-386.

Underkoffler, J., and Ishii, H. 1999. Urp: A Luminous-Tangible Workbench for Urban Planning and Design. In *Proc. of CHI'99*, pp. 386-393.

Wellner, P. 1993. Interacting with paper on the Digital Desk. In *Comm. of the ACM.* July 1993, pp. 86-96.

Williamson, C., and Shneiderman, B. 1992. The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. In *Proc. of SIGIR'92*, pp. 339-346.