CubeQuery: Tangible Interface for Creating and Manipulating Database Queries

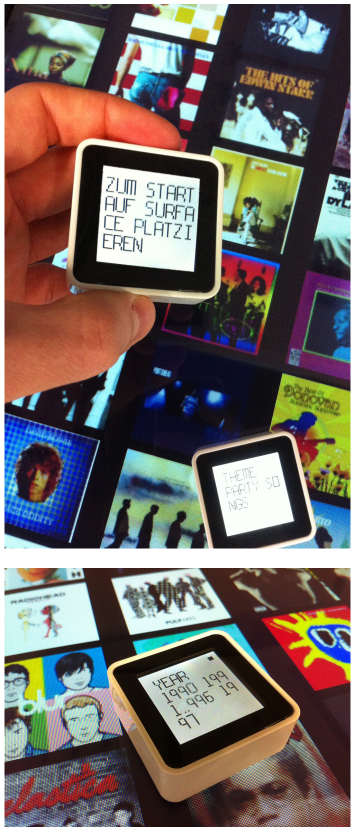

Figure 1: Small tangibles allow users to create and manipulate search parameters of a database query. (↑ top) Initially every token is unbound, i.e., without a selected facet. (↓ bottom) Token with selected facet (i.e., *year*) and values.

Ricardo Langner
Interactive Media Lab
Technische Universität Dresden
Dresden, Germany
langner@acm.org

Anton Augsburg
Interactive Media Lab
Technische Universität Dresden
Dresden, Germany

Raimund Dachselt
Interactive Media Lab
Technische Universität Dresden
Dresden, Germany
dachselt@acm.org

Abstract
We demonstrate *CubeQuery*, a tangible user interface providing a physical way to both create and manipulate basic database queries. This interactive installation is designed for individual faceted browsing and allows users to explore contents of a database, i.e., a music library. While each tangible represents an individual search parameter of a search request, the physical arrangement of multiple tangibles permits the combination of search parameters by utilizing basic logical operators. Goal of this research is to explore the practicality of spatial arrangement of tangibles to ease the process of faceted browsing.

Author Keywords
Tangible UIs; Faceted Browsing & Search; Database Queries; Sifteo Cubes; Microsoft Surface

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation: User Interfaces].

Introduction
Using a tangible query interface to physically create and manipulate search requests has been investigated in several research projects [1, 2, 6]. However, using active

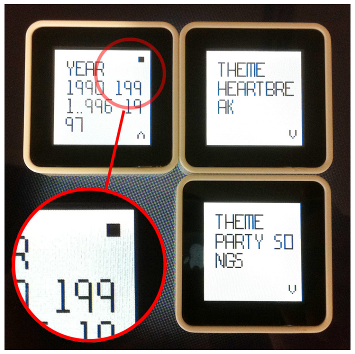

Figure 2: Spatial arrangement: tokens can be placed side by side to create combined queries. (close-up) An icon highlights the main token and active query.

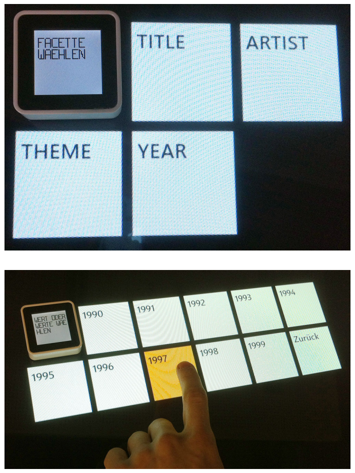

Figure 3: Available values. (↑ top) Users can selects facets, such as *title*, *artist*, *theme*, or *year*. (↓ bottom) Selection of facet-specific values.

tangibles – tangibles equipped with, e.g., a display, a number of sensors, or wireless communication – for such search tasks had previously received more attention [7]. Therefore, our *CubeQuery* concept uses Sifteo cubes [3, 4], a commercially available interactive game system that provides a SDK to create custom applications. Our interactive demonstration allows users to explore a dataset (i.e., music library) by spatially arranging tangibles on an interactive surface.

The general idea is to provide users with a set of tokens (cf. [2]), i.e., tangibles representing individual search parameters. A search parameter is defined as a facet (e.g., categorical, numeric) and one or more facet-specific values. In contrast to *Stackables* [2], where tangibles are vertically stacked, *CubeQuery* explores the spatial alignment (side by side) of tokens.

CubeQueries: Design and Concept

The *CubeQuery* concept utilizes various features of a rich interaction space. Based on the capabilities of the deployed hardware, these features can be classified into two categories: output attributes and input attributes.

Output Attributes. Output is provided both on the interactive surface as well as on the tokens. While the surface features visual and audio feedback, tokens use visual output only. In terms of an application state, tokens supply local (personalized) feedback, whereas surface feedback represents both local and global.

Input Attributes. The interaction vocabulary develops from the use of the interactive surface, tokens, and the combination of the devices. The surface provides direct touch input only, allowing users to tap, drag, or swipe visuals. Since *CubeQuery* makes use of the full functionality provided by the Sifteo cubes, features of tokens directly correspond to the hardware capabilities. Unique identifiers allow tokens to be distinguished. Tokens can be tilted, flipped, and shaken by making use of the build-in motion sensor. They also allow users to press the display. Furthermore, tokens can be positioned on the surface. This results in a token-specific location, i.e., a 2D position and orientation. Finally, tokens can be placed side by side relating to each of the four sides of a tangible (see Figure 2). Considering the interactive surface, this placement can be performed either off-screen or on-screen.

To explore a dataset, users can define search parameters and logically combine them in queries, i.e., search requests. A typical work flow of binding a search parameter involves (1) assigning a facet to a token, (2) selecting facet-specific values, (3) optionally combining parameters by using logical operators, and (4) optionally putting a request by activating the query. In this context, *CubeQuery* enables users to perform various operations.

Facet selection. An individual search parameter related to a specific facet can be bound to a specific token. Initially, a token is unbound, i.e., neither a value nor a facet is set up. If an unbound token is placed down on the surface, the application shows available facets. Users can select (bind) a facet by tapping it (see Figure 3 top). The application shows corresponding facet values immediately after a the selection of a facet (cf. value selection). To unbind a token again, i.e., clear both the values and the facet users simply grab and shake the tangible.

Value selection. Depending on the data type and value range of a selected facet, users can choose single or multiple values. Available values are shown right beside (or around) the corresponding token (see Figure 3 bottom). Exactly like the facet selection, a *single-value selection* requires the user to tap the specific value.

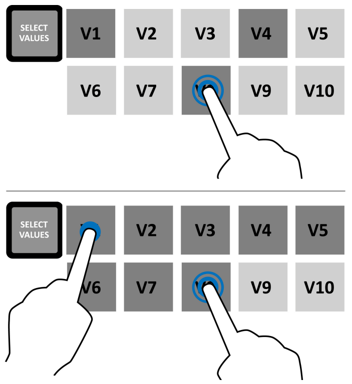

Figure 4: Multi-value selection. (↑ top) Selection of individual values separated from each other. (↓ bottom) Selection of a continuous value range.

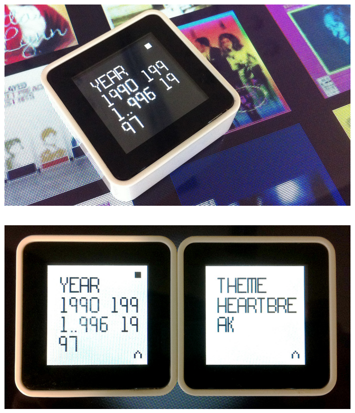

Figure 5: (↑ top) Negation of a search parameter.
(↓ bottom) Horizontal alignment represents AND operator.

Tapping a value again allows users to undo the selection (toggle behavior). Selecting several values in sequence results in a *multi-value selection*. Such a selection can represent both individual values separated from each other as well as continuous value ranges (see Figure 4). The selection of a continuous value range can be achieved in various ways. *CubeQuery* allows a successive selection of values step by step. Although this might be obvious, it is not effective. Hence, our design considers the following alternative methods: Either touching and swiping values (uni-manual) or touching and holding the first value with one finger and tapping the last value with another finger (bi-manual), see Figure 4 bottom. Besides a common value selection, users might want to exclude specific values without selecting every other value. In contrast to value selections involving touch input, a negation of previously selected values can be achieved by physically flipping a token back and forth. To both provide adequate feedback and help users to identify such negated tokens, the visual output is inverted, i.e., white text on dark background (see Figure 5).

Activation. As the interactive surfaces shows the results of one main search request (cf. visualization of query results), one main token has to be declared. Only if a main token is present a search query will be processed. The main token can be set by pressing the display of the token. A small icon visually highlights the main token (see Figure 2 close-up).

Combining parameters. Multiple tokens can be combined to formulate more complex queries. By placing tokens side by side, users can combine search parameters using simple boolean AND or OR operators. Here, the alignment affects the boolean operator. Tokens need to be combined horizontally to apply an AND operator (see Figure 5 bottom). A vertical alignment results in a logical OR combination. The order of execution of a combined query is time-dependent, i.e., corresponds to the chronological order of the token combinations. Again, icons visualize the logical combination and indicate the type of operator.

Visualization of Query Results
The result of a query is shown on the display of the interactive surface. Similar to common media library applications, the visualization is presented as a grid view containing result elements (see Figure 1, 6). If the sample space contains more elements than the grid view can display, users can pan or swipe the grid view horizontally to scroll. The grid visualization immediately responds to query changes, e.g., adding or removing a token. This way users can dynamically modify a query and explore a dataset in an interactive way.

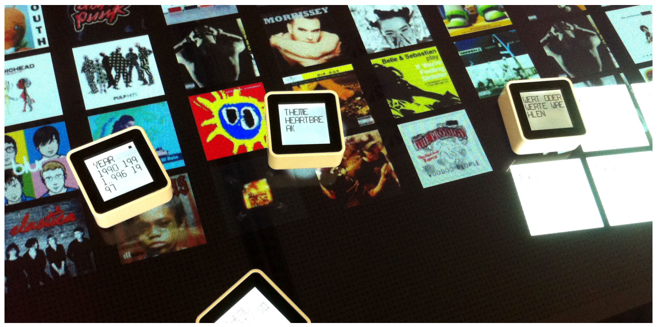

Figure 6: A grid view shows the result of a query. The visualization immediately responds to query changes.

Implementation

CubeQuery makes use of the first version of the Sifteo cubes [4] and a Microsoft Surface SUR40, a touch-sensitive interactive tabletop. The cubes communicate wirelessly via a proprietary 2.4GHz protocol with the surface (host PC). To receive the location of a token, each cube is tagged with a fiducial marker, i.e., a 4x4 cm infrared reflective marker with a unique 8-bit identification pattern. The prototype consists of two components. The first component, written in C# and using the official Sifteo SDK, provides application logic and a communication interface to the Sifteo cubes. The second component, written in C# and using WPF, features the grid visualization and handles touch and marker input. Both components run on the host PC (i.e., Microsoft Surface SUR40) and communicate through a TCP socket connection. The *CubeQuery* prototype enables users to explore a musical dataset of approx. 600 songs [5] where users can select various facets, such as theme, title, artist, and year.

Conclusion

We developed *CubeQuery*, a tangible user interface providing a physical way to both create and manipulate basic database queries. We wanted to ease the process of faceted browsing and make information seeking more effective. Unlike the majority of previous work, we utilized active tangibles. With our *CubeQuery* prototype we explore the practicality of creating simple boolean query constructions by aligning these devices side by side.

Acknowledgements

We thank Mathias Frisch, Ulrike Kister, André Viergutz, and Wolfgang Büschel for productive discussions and their valuable feedback.

References

[1] Jetter, H.-C., Gerken, J., Zöllner, M., Reiterer, H., and Milic-Frayling, N. Materializing the query with facet-streams: A hybrid surface for collaborative search on tabletops. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 3013–3022.

[2] Klum, S., Isenberg, P., Langner, R., Fekete, J.-D., and Dachselt, R. Stackables: Combining tangibles for faceted browsing. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, ACM (New York, NY, USA, 2012), 241–248.

[3] Merrill, D., Sun, E., and Kalanithi, J. Sifteo cubes. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*, CHI EA '12, ACM (New York, NY, USA, 2012), 1015–1018.

[4] Sifteo, Inc. Sifteo cubes. `https://www.sifteo.com/cubes`.

[5] The Guardian. The 1000 best songs ever. `https://opendata.socrata.com/dataset/1000-Songs/y5f9-excn`.

[6] Ullmer, B., Ishii, H., and Jacob, R. J. Tangible query interfaces: Physically constrained tokens for manipulating database queries. In *Proceedings of the INTERACT '03*, vol. 2003 (2003), 279–286.

[7] Valdes, C., Eastman, D., Grote, C., Thatte, S., Shaer, O., Mazalek, A., Ullmer, B., and Konkel, M. K. Exploring the design space of gestural interaction with active tokens through user-defined gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, ACM (New York, NY, USA, 2014), 4107–4116.