IVEE: An Information Visualization & Exploration Environment

Christopher Ahlberg & Erik Wistrand

Department of Computer Science and SSKKII
Chalmers University of Technology
S-412 96 Göteborg
Phone: +46 31 772 5410
Email: {ahlberg,wistrand}@cs.chalmers.se

To be published in Proceedings of IEEE Viz'95

SSKKII
c/o Department of Linguistics
Göteborg University
S-412 96 Göteborg
Sweden

IVEE: An Information Visualization & Exploration Environment

Christopher Ahlberg & Erik Wistrand
Department of Computer Science and SSKKII
Chalmers University of Technology
S-412 96 Göteborg
Phone: +46 31 772 5410
Email: {ahlberg,wistrand}@cs.chalmers.se

Abstract

The Information Visualization and Exploration Environment (IVEE) is a system for automatic creation of dynamic queries applications. IVEE imports database relations and automatically creates environments holding visualizations and query devices. IVEE offers multiple visualizations such as maps and starfields, and multiple query devices, such as sliders, alphasliders, and toggles. Arbitrary graphical objects can be attached to database objects in visualizations. Multiple visualizations may be active simultaneously. Users can interactively lay out and change between types of query devices. Users may retrieve details-on-demand by clicking on visualization objects. An HTML file may be provided along with the database, specifying how details-on-demand information should be presented, allowing for presentation of multimedia information in database objects. Finally, multiple IVEE clients running on separate workstations on a network can communicate by letting one user´s actions affect the visualization in an another IVEE client.

1 Introduction

The cognitive load while performing information retrieval tasks is usually high. Finding the right query formulation which will deliver a query result with high result precision is cognitively difficult. The benefit of static visualizations of data sets has long been known, visualizations of for example demographic, geographic, and economic data sets are commonplace today [25][9]. Allowing users to incrementally control animated visualizations of databases, queries, and query results can minimize the mental effort needed for:

- Finding the right query formulations
- Grasping relations between queries and query results
- Judging relations between individual query results.

Benefits of animated visualizations partially stem from the fact that we can perform many perceptual tasks such as detection of patterns and anomalies in pictures with little conscious mental effort. Not until recently has the use of these remarkable perceptual abilities found its way into information retrieval systems [7][1][12][19].

Visual representations of databases allows for presentation of many more database elements in single screens than traditional text based methods. This in turn allows for overviews in a way which is not possible with text based query systems. A visualization might present thousands of elements in one single screen, while a text based system is limited to the magnitude of tens of elements.

1.1 Dynamic queries

Dynamic queries is a concept for information exploration and database querying [1], defined as:

”a user controlled animated visualization of a query process, including databases, queries, and query results.”

Using dynamic queries, users manipulate query devices (e.g. rangesliders, toggles, and alphasliders) of various kinds to construct queries to incrementally update (with near immediate updates) a visualization of the current query result. Users might increase the value of a query parameter to see how the increase affects the size (recall) of the result set. The relevance (precision) of the result set can be judged quickly from the color of the visualization objects (i.e. the result set), or from the location of the result set in the visualization (indicating for example geographic proximity or some other high level semantic property). Users might also manipulate query devices to explore trends and patterns in data, or to detect anomalies. This is usually done by manipulating for example a rangeslider to observe if excluding a certain part of a query range correlates with visualization objects disappearing in some particular pattern. The power of dynamic queries compared to other query methods, such as form fill-in and natural language has been confirmed in controlled experiments [1][27].

1.2 Other visual query techniques

The Information Visualizer from Xerox Parc utilizes visualizations such as Cone Trees and the Perspective Wall [16][18]. 3D visualizations allow users to detect clusters in data sets and search for information in context of the whole database. Hjemme *et.al* [12] presents the LyberWorld system which holds a visualization controlling a full text retrieval process, partially based on Cone Trees. They emphasize the use of visualizations for perceiving hidden information carried by relations in complex data structures. Spoerri [22] introduced the InfoCrystal, with a visualization of the subresults from a complex query, and their relations to the query terms. By inspecting the visualization, users can quickly judge the relevance of the subqueries. Other interesting examples of visualization systems are XGobi [23] and the AT&T Data Visualization Sliders [11].

1.3 Dynamic queries examples

A number of interesting prototype dynamic queries applications has been built for experimental purposes, such as a dynamic periodic table [1] and a dynamic homefinder [27]. These prototypes depended on domain specific visualizations such as geographic maps and the periodic table of elements. The FilmFinder [2] extended the scope of dynamic queries by introducing the starfield display where two ordinal variables from a database relation are plotted against each other and used as an interactive visualization.

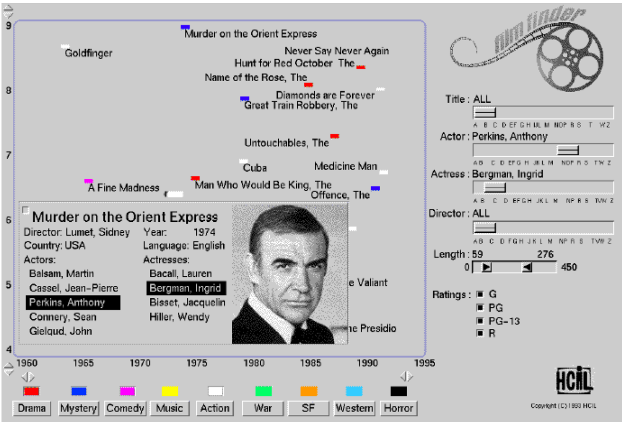


Figure 1: The FilmFinder dynamic queries prototype [2].

2 IVEE

Today, a form fill-in module is provided with most commercial database management systems (DBMS). The form fill-in module allows users to easily create forms for both entering data into and querying a database. Similar to a form fill-in module, a dynamic queries system could be provided with a DBMS. The dynamic queries system would allow for automatic creation of a visual query environment from a database relation.

The Information Visualization and Exploration Environment (IVEE) is an attempt at such a system [1]. IVEE can automatically create dynamic queries environments holding query devices and visualizations (Figure 3). IVEE imports relations with named attributes, given on a straightforward text format. The relations might originate from database systems such as Oracle or spreadsheet programs such as Microsoft Excel.

An important challenge for us was to design IVEE so that users could get started with their visualization and exploration tasks with as little effort as possible. Obviously this is a generic goal for most software developed today. However, visualization software is often quite complex and requires users to go through many steps before the actual exploration process can start. We want to demonstrate how a visualization system can be designed to automate the task of creating both visualizations and manipulation objects and let users start working directly with their high level exploration tasks.

Users of IVEE may use the tool at different levels:

- IVEE is used to explore a data set only using the standard visualizations (starfields) and the automatically created query devices.
- IVEE is used with modified visualizations and query devices, with the configuration provided along with the database (possibly created by another user).
- IVEE is used both to explore a data set and to interactively update visualizations (attaching graphical objects to database objects, varying color schemes) and query devices (changing query devices used for database querying, changing the layout).

IVEE examines the data in a user specified relation and classifies it into datatypes (integers and strings) and size (total number of values held in the attribute and number of distinct values held in the attribute). The data is stored in an internal IVEE database object (Figure 2). Attributes are stored as straightforward vectors, with multiple indexing to increase performance for various search tasks.

The user interface of IVEE holds two main components, a query area holding a number of query devices (right area in Figure 3) and a visualization area (left area in Figure 3), holding a number of visualization. These user interface objects are reflected in the architecture of IVEE in Figure 2. Notice that IVEE can hold several instantiations of visualizations, databases, and query areas.

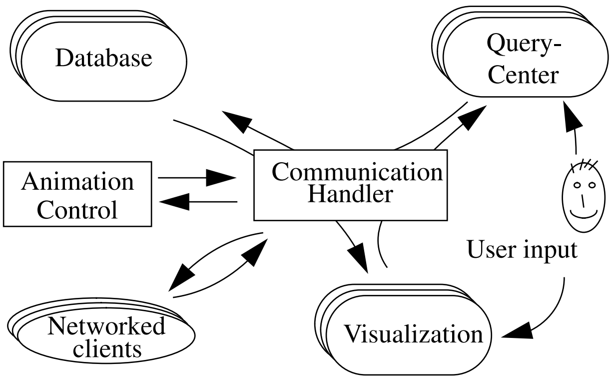


Figure 2: The IVEE architecture.

3 Visualizations in IVEE

Another design goal for IVEE has been to provide users with a rich collection of visualizations. Ours and many others' work on interactive visualization seem to point to that successful visualization environments does not depend on one single powerful visualization, quite contrary a whole smörgåsbord of visualizations appropriate for various tasks and datatypes is closer to a successful solution. Our approach depends on a simple, yet powerful architecture. To each object in a database relation a (possibly user defined) graphical object is attached. This graphical object may be a simple point of light or square as used in starfield visualizations [2], a glyph from a predefined library to create visualizations like those in [21], or arbitrary graphical

1. IVEE is available from ftp.cs.chalmers.se:/pub/IVEE. See also the WWW-page http://www.cs.chalmers.se/SSKKII/ivee.html

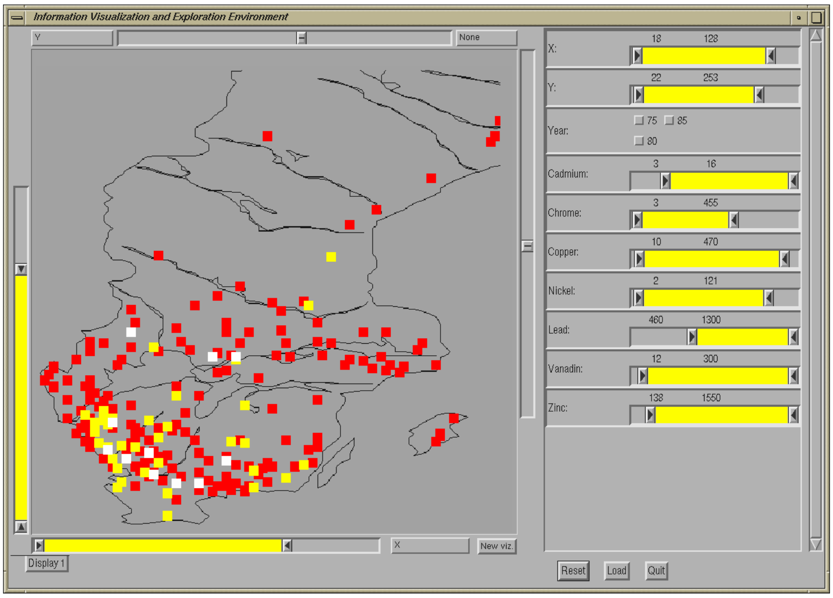


Figure 3: Example of a dynamic queries environment created with IVEE, in this case an environmental database browser. Users can browse and query a database of thousands of measurements of heavy metals in Sweden.

objects in two or three dimensions to create full-fledged interactive visualizations of objects such as buildings, airplanes, or abstract objects such as Cone Trees [18] and Tree-maps [14]. In the spirit of our goal to make IVEE easy to get started with, users can start by specifying nothing about polygons connected to database objects at all. This will associate a square with each object. When user defined polygons are specified this is done by providing them in a file along with the database. The file might contain one unique graphical object for each database object or a smaller number of polygons and a mapping from database objects to graphical objects. For a database of cars, motorcycles, and trucks three complex graphical objects might be specified, one for each type of database object.

To achieve the high performance graphics necessary for dynamic queries and interactive 2 and 3D graphics, IVEE utilizes an animation loop which adapts rendering strategies to the current performance of the computer and user behavior. Examples of rendering approximations are:

- Render objects at a lower resolution.
- Render shaded objects as wireframe models.
- Skip textual labels.
- Do not fully redraw the screen while performing expensive queries.
- Store multiple views of the same graphical object.

IVEE offers a number of visualizations where the basic one is the starfield [2]. A starfield is an interactive scatterplot with additional features for zooming, panning, details-on-demand, etc. Two ordinal variables from a database relation are chosen as the axes in the starfield (Figure 4).

The power of the starfield as a visualization of complex databases is that it allows for displaying a large number of database objects in one single screen. Each object is represented by a small graphical object which can be coded by color, brightness, shape, size, etc. The spatial location of an element can effectively encode two properties of a database object. The starfield provides important qualities of visualizations such as:

- The overview provides starting points for search.
- Query results may be presented in the context of the full database.
- Query result relevance can be judged quickly.
- Trends and anomalies can be explored effectively.
- Feedback can be provided continuously during queries.
- Works for many datasets.

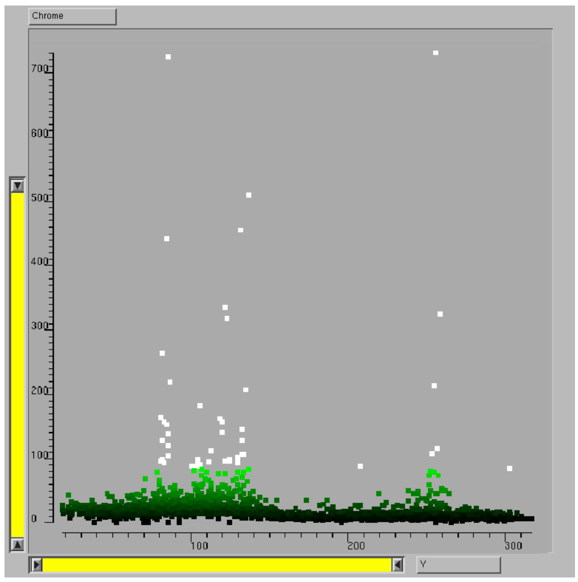


Figure 4: Starfield visualization of the same environmental database as in Figure 3. Users can zoom, pan, query, select-details-on-demand and interactively change the attributes that are visualized in the starfield.

In addition to allowing each database object be associated with a square in a starfield, IVEE supplies users with several other possibilities for coding properties of database objects into the starfield. Visualization elements can be colored using two different schemes – users might associate database values of an attribute with either:

- Colors of different hues (blue, yellow, red, etc.) which is useful for coding nominal attributes.
- Colors of the same hue, but shifting in brightness which is useful for coding continuous attributes.

When utilizing the latter scheme, users can specify a certain part of the range of an attribute to be colored differently – useful for indicating for example objects with values in the top 5% range. Users can also associate glyphs from a predefined library to nominal attribute values which can effectively complement the use of color.

3.1 Geographic visualizations

The immediate extension of the starfield is to provide a background map which can be used to create geographic visualizations. IVEE users can specify such a visualization context by providing background objects holding a number of polygons defining an appropriate background. Background objects are not attached to any database object.

Background objects might be geographic maps, but may also very well be a structural drawing of a house. An important property of geographic visualizations compared to for example starfields is that they provide important and often well-known context for the presentation of query results. As background objects as well as other graphic objects are provided on a vector format, arbitrary zooming can be performed smoothly.

3.2 Other visualizations

So far, the visualizations described have not utilized the possibility of associating arbitrary polygons with database objects. An interesting example where this can be used is node and link diagrams. Similarly to how a geographic map is specified, users can specify a layout of nodes and links, and each graphical object is associated with a database object. By manipulating query devices (elaborated below), users can select nodes of certain types, links for which an associated value is within a given range, etc.

This is not an ideal way of creating node and link diagrams. However, we do think it shows how the simple yet powerful way of associating database objects with complex polygons can be used to create interesting visualizations. Similar visualizations which can be created with IVEE are for example Tree-maps and Cone Trees. Hierarchy visualizations are widespread and we plan to incorporate functionality for handling those more smoothly.

Yet another class of visualizations that can be created are those depending on complex three dimensional objects. In Figure 5 a schematic visualization of a part of the computer science department at Chalmers University of Technology is presented – used in a dynamic queries interface for searching for persons in the department. Background objects as described above make up the outer structure of the building and employees (database objects) are represented by their offices. Manipulating query devices will grey out those rooms not fulfilling the search criteria and allows for detection of trends such as how professors with high salaries are more often located in the top of the building!

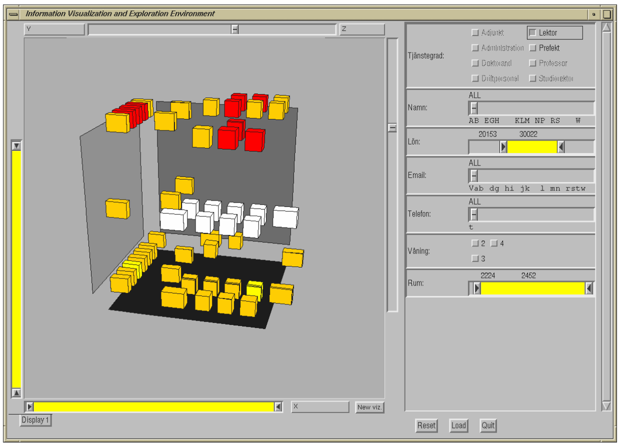


Figure 5: Dynamic queries interface created with IVEE, allowing users to browse some of the employees of the computer science department at Chalmers. Each box in the visualization corresponds to an office of a member of the staff in the department.

3.3 Multiple visualization areas

IVEE allows multiple visualizations to be created simultaneously, and when the query devices are manipulated all the visualizations are updated interactively, allowing for powerful exploration of data. Examples of how this might be used is described in [10]. Users might observe a cluster of points in a starfield. When zooming into the cluster a geographic map is updated next to the starfield, where the same points can be observed to all be situated in the same geographic region.

Further, users might activate more than one page of visualizations simultaneously. A notebook similar system with tab buttons (similar to the ones in Microsoft Windows and Motif) is used. By pushing the”New Visualization” button, a new visualization area is created and the old one stored in the background (Figure 6).

3.4 Visualization manipulation

Users can perform a number of operations on the starfield:

- Zooming
- 3D manipulation
- Panning
- Filtering
- Selection of details-on-demand

In this section zooming, 3D manipulation, and panning is discussed. Filtering and details-on-demand will be discussed below. Zooming is performed either with the mouse buttons (second mouse button to zoom in and third mouse button to zoom out) or the zoom bars (Figure 6)[13]. Manipulating the zoom bar on either the X or the Y axis of the visualization will either increase or decrease the size of the area in view, i.e. zooming is performed in either of the dimensions. Mouse button zooming allows for zooming in both dimensions simultaneously, while the zoom bars are appropriate for zooming each dimension separately.

Both zooming methods are useful, the former for example when there is an obvious spatial relation in the visualization which should be preserved, such as in a geographic map, and the latter when no such spatial relation exists, for example in a scatterplot. By dragging the area between the sliderthumbs in the zoom bars, users can pan the visualization without changing the zoom rate.

To rotate a 3D visualization users manipulate two sliders (Figure 6) to rotate the view around each of the X and Y spatial dimensions, similar to the slider approach described in [8]. Zooming in 3D is performed with the mouse buttons, as described above. However, the zoom bars are not enabled when manipulating a 3D visualization, as there is no direct mapping from a 2D range to a perspective view with the eye placed at an arbitrary viewpoint.

4 QUERY DEVICES IN IVEE

Based on the classification of the user specified relation IVEE selects query devices for each attribute. Query devices are selected from rangesliders, alphasliders [3], and toggles (all shown in context in Figure 6). Simple rules decide which query devices are assigned to which attributes. For example, for a string attribute with more than 10 distinct items, an alphaslider is selected, otherwise a group of toggles with a toggle for each distinct item is selected (of course, the threshold 10 can be changed). For integers and reals, rangesliders are selected, unless only 10 distinct items exist in the attribute – then a group of toggles is selected.

A different strategy would be to allow users to manually select which widget to use for each attribute. However, while this strategy might lead to better specifications of query environments, it would be to cumbersome for large databases with many attributes. Also, users can change the widget used for an attribute interactively while exploring a database. This is performed by activating a menu attached to the query device currently used for the attribute (Figure 6). From the same menu users can move the query device up, down, to the top, and to the bottom of the query area. Regrouping widgets in another order than the one provided implicitly by the order of the attributes in the database relation is quite useful, especially for relations with a large number of attributes.

4.1 Rangesliders

Rangesliders are useful for selecting range criteria for integer attributes and other attributes with an order relation (Figure 6). The rangeslider provides a natural representation of the query it is representing, i.e. a range query. Increasing or decreasing the range of an attribute allows for powerful exploration of trends and anomalies. Rangesliders are also very effective for relaxation of query parameters when the result set can be observed immediately in a visualization.

4.2 Alphasliders

The alphaslider is a widget for selecting items from long lists of for example strings [3]. The rationale behind the alphaslider is its small size - it allows for selection from lists of thousands of elements in a small screen area Figure 6. Screen area is precious in a visual query system where most screen space should be used for the visualization of data. The alphaslider can not only be used for selection of specific strings, it is also very useful for browsing categorical variables while observing the query result set in a visualization. For example in the FilmFinder [2] this could be used for browsing the directors attributes while observing result sets showing up in the upper corner of the display, indicating films to be new and popular.

4.3 Toggles

Toggles are the last of the widgets currently used for query formulation in IVEE (Figure 6). Toggles are useful when only a few alternatives exist for an attribute and these alternatives should be presented on the screen explicitly. Users may select multiple values for an attribute with a group of toggles.

4.4 Query evaluation and tight coupling

Queries are composed from the conjunction of all the query components defined by the query devices. Query devices in their initial state let through all database objects, i.e. they do not affect the result of the query. Manipulating a device immediately affects not only the visualization, but also the other query devices – the query mechanism is tightly coupled [4]. A query device manipulation restricts the query range of the other query devices to only include criteria existing in the remaining elements of the database. This useful for two purposes:

- To guide querying when searching for specific elements in the database – for example after having selected Ingemar Bergman with the director slider for a film database only actresses appearing in an Ingemar Bergman film will be selectable with the actresses slider.
- To facilitate browsing of categorical variables. A typical use of the alphaslider is to browse a categorical variable, such as sample number for a statistical data set. After having selected a subset of the samples with some other query device, users can browse the samples – without looking at the alphaslider – and focus on the visualization. When an interesting pattern is found in the visualization users can look at the alphaslider to find out the identification of the particular sample. This strategy is not nearly as useful if most of the alternatives on slider cause an empty visualization, i.e. an empty query result.

5 Details-on-demand

Equally important to users being provided with effective overviews of data is that they are able to retrieve full and rich descriptions of specific database elements. Only showing details when they are requested is instrumental for the concept of dynamic queries. To draw the objects in for example a starfield visualization only requires IVEE to access two or three attributes in each potentially very large database object. IVEE users select details-on-demand by clicking on a visualization object with the left mouse button

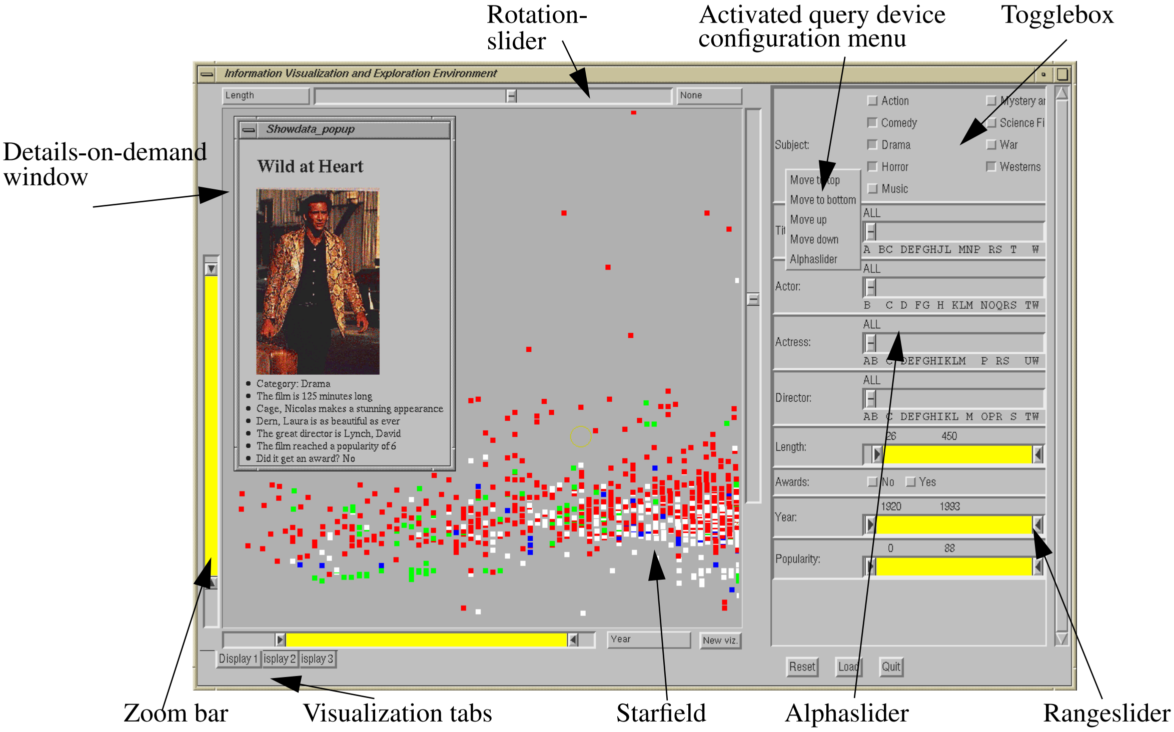


Figure 6: Reproduction of the FilmFinder in [2], created with IVEE.

(Figure 6). Upon moving the mouse cursor over a selectable object feedback is provided indicating its selectability. In addition to the "ordinary" data in a database object consisting of integers, reals, and strings, IVEE can hold file pointers to multimedia objects, such as images, speech, video, etc., on standard formats such as GIF and MPEG.

IVEE has a predefined approach to presenting data in database objects when they are selected. However, users can create much richer presentations by providing a HTML-based formatting document. IVEE utilizes the public domain HTML-widget in the NCSA Mosaic distribution. Along with a database, users may supply a file like the one in Figure 7. This is the standard HTML format, except for the placeholders indicating where database attribute values should be filled in. The placeholders consist of a ´#´ character followed by the attribute name. The HTML file may contain pointers to images, sound, video, and other local HTML files.

```
<head>
<title>Film description</title>
<body>
<h1>#Title</h1>
<img src="#Image">
<ul>
   <li> Category: #Subject
   <li> The film is #Length minutes long
   <li> #Actor makes a great appearance
   <li> #Actress is as beautiful as ever
   <li> The great director is #Director
   <li> Popularity: #Popularity
   <li> Did it get an award? #Awards
</ul>
```

Figure 7: HTML description of the details-on-demand window in Figure 6.

Being able to receive details-on-demand is not only important for examining individual database objects, it is also useful as a starting point for search. The overview provided by for example a starfield helps users overview potentially very large amounts of data. However, a complementary approach to starting a search from the overview is to start with a specific database object which might have been found in the visualization or by initially specifying a strict set of criteria leaving one or a few objects selected. This approach has the advantage of not overloading users with a large abstract set of complex data. Instead they can apply their knowledge of one specific element [26].

IVEE allows for this approach to searching through the details-on-demand popup-window. Users can click with the left mouse button on any of the boldface texts in the window, and thereby setting the value of the appropriate query device to the marked value.

6 Multiple databases

The initial assumption of IVEE is that the database consists of one single relation. For databases consisting of several relations this constraint can sometimes be overcome by creating a universal relation which allows a whole set of relations to be regarded as one single relation [15]. However, this approach is not always ideal, and accordingly users can load several database relations (sharing at least two attributes which can be visualized) simultaneously into IVEE. Each relation is given its own dedicated set of query devices, and each attribute of relation is given its own dedicated query device – i.e. IVEE does not attempt to join similar attributes which appear in several relations (although this might in some cases be useful).

Allowing several database relations to share the same manipulable visualization creates a powerful external representation for tasks such as the matching of objects in the respective relations. A good example is two relations, one holding job opportunities and one holding job applicants. Both might be visualized in for example a map or a starfield, and users (e.g. job agency employees) can manipulate query devices for both relations to find reasonable subsets and then visually scan the visualization to find matching opportunities/applicants.

7 Distributed exploring

In many interesting application areas for systems such as IVEE there are many people interested in the same data set. Above the task of matching of jobs and skills was mentioned, an other example might be analysis of nearly any statistical data set. Unless these people are at the same site, or even in the same corridor, they are most likely to run into the situation of having to discuss querying and analysis of their data set over the telephone. For the job/skill matching situation which we have examined extensively, a common situation is that employment agency employees have to discuss cases (persons, jobs) over the phone with their colleagues [24]. The traditional text based system currently used does not offer any help for this situation.

IVEE allows several persons to visualize and explore the same data set simultaneously. By starting several (at least two) clients of IVEE, loaded with the same database, and then requesting these to be connected through a UNIX socket (Figure 2), several users can manipulate the same visualization. Each client is responsible for the actual visualization and the corresponding local database, but operations such as querying, zooming, details-on-demand, etc., are distributed to all other connect clients. Only transmitting user actions and not actual query results allow this approach to be possible even without a high speed connection. The scheme allows several people to share a common view of a data set while communicating over a traditional communication system, e.g. the telephone. Currently we have only a rough implementation of socket communication between IVEE clients and many problems remain to be solved such as how several cursors can be managed, what to do when two users simultaneously perform contradicting actions, etc.

8 FUTURE WORK

We plan to continue to develop IVEE, and extend its functionality with for example:

- More query devices. The existing ones can be extended in functionality, but new ones need to be introduced to cover other kinds of queries than those described above.
- More types of visualizations. Hierarchies of various kinds are common and a large number of hierarchy visualizations exists. We want to extend the support in IVEE for those.
- Arbitrary boolean combinations of query components. Currently the query mechanism only supports AND:ing of all query components. For some situations it would be meaningful to be able to create more complex queries.
- Visualization wizard. Users of IVEE are faced with a large number of choices for visualization configuration parameters (today manipulated in a popup window). We would like to battle this problem by introducing a visualization wizard similar to the graph wizard in Microsoft Excel which can provide structure.
- Our current HTML document support does not extend to communication with world wide web (WWW) servers beyond our own department. If such functionality was introduced, IVEE could be an excellent base for visualization and interaction with the WWW.
- An obvious constraint on a tool such as IVEE is the size of the database it can handle. IVEE can currently handle a database consisting of 5000-6000 objects with some 10-15 attributes each running on a SGI Indy 100MHz machine. A necessary area of further research is to explore datastructures for pushing this to 50-100’000 objects.
- Another typical problem in a tool based on scatterplots for displaying data is overlapping points which hold exactly the same X and Y values in the display.

9 CONCLUSIONS

IVEE is an interactive visualization and query system based on the concept of dynamic queries. The IVEE architecture is based on the simple concept of attaching more or less complex graphical objects to database objects in visualizations. Graphical objects might be simple colored points of light, glyphs selected from a predefined library, or arbitrary sets of polygons in two and three dimensions. This allows IVEE users to fairly easily import database relations and create complex visualizations. The visualizations can be interactively queries, filtered, zoomed, and panned.

A major research area for our group right now is to look at how dynamic queries techniques can be utilized in the situation of matching jobs and skills. IVEE has allowed us rapidly prototype dynamic queries interfaces for evaluating visualizations for this job/skill matching application (starfields, geographic maps, and text output) and explore techniques for guiding query composition in direct manipulation systems [4].

Other applications of IVEE have been to create a visualization system for exploring data on spoken language, used by researchers in linguistics [6], and a system for browsing environmental data – as shown in Figure 3. The Human-Computer Interaction Laboratory at University of Maryland are currently using IVEE for their projects.

We believe that it is instrumental for work on visualization and information exploration to be close to end-users and real applications. New technology such as large high resolution screens, fast graphics, new input devices, and high capacity network connections makes large promises for exiting information visualization applications. Working with end-users and their problems helps us find meaningful use of these new technologies and also helps us push our own creativity further.

10 ACKNOWLEDGEMENTS

This work was in part supported by NUTEK, grant no: 5321-93-2760, and Arbetsmiljöfonden, grant no: 94-0525.

The authors want to thank Staffan Truvé, Jens Allwood, and Johan Hagman for fruitful discussions during the development of IVEE. Ann Rose and Catherine Plaisant at the HCIL, University of Maryland, has provided us with valuable comments and proposals sprung out of their use of IVEE. We also want to thank Ben Shneiderman for continued support and encouragement in the work with dynamic queries.

REFERENCES

[1] Ahlberg, C., Williamson, C., Shneiderman, B. (1992), Dynamic Queries for Information Exploration: An Implementation and Evaluation. *Proceedings ACM CHI'92: Human Factors in Comp. Systems, pages 619-626*. Also in Shneiderman, B. (Ed.), *Sparks of Innovation in Human-Computer Interaction*, Ablex, Norwood, N.J, 1993.

[2] Ahlberg, C., Shneiderman, B. (1994), Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. *Proceedings ACM CHI'94: Human Factors in Comp. Systems*, pages 313-317. Also in Baecker, R., Grudin J., Buxton, W., and Greenberg, S., *Readings in Human-Computer Interaction: Toward the Year 2000* (2nd Edition), Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[3] Ahlberg, C., Shneiderman, B. (1994), The Alphaslider: A Compact and Rapid Selector. *Proceedings ACM CHI'94: Human Factors in Comp. Systems*, pages 365-371.

[4] Ahlberg, C., Truvé, S. (1995), Tight Coupling: Guiding User Actions in a Direct Manipulation Based Information Retrieval System, Proceedings of EHCI'95: Engineering for Human-Computer Interaction, C. Unger, ed., IFIP Transactions series, Chapman & Hall.

[5] Ahlberg, C., Wistrand, E. (1995), IVEE: An Environment for Automatic Creation of Dynamic Queries Applications, *Proceedings of ACM CHI'95: Human Factors in Comp. Systems*.

[6] Allwood, J., Ahlberg, C. (1995), Visualizing Spoken Interaction, *Proceedings of the 15th Scandinavian Conference of Linguistics*, Oslo University.

[7] Card, S., Robertson, G., Mackinlay, J. (1991), The Information Visualizer, an Information Workspace, *Proceedings ACM CHI'91: Human Factors in Comp. Systems,* pages 181-188.

[8] Chen, M., Mountford, J., Sellen, A. (1988), A Study in Interactive 3-D Rotation Using 2-D Control Devices, *Proceedings ACM SIGGRAPH'88*, pages 121-129.

[9] Cleveland, W. (1994), *The Elements of Graphing Data*, Hobart Press, Summit N.J. 297 pages.

[10] Cleveland, W. (1993), *Visualizing Data*, Hobart Press, Summit N.J., 360 pages.

[11] Eick, S. (1994), Data Visualization Sliders, *Proceedings ACM SIGGRAPH Symposium on User Interface Software and Technology'94 proceedings*.

[12] Hemmje, M., Kunkel, C., Willet, A. (1993), LyberWorld – A Visualization User Interface Supporting Fulltext Retrieval, *Proceedings ACM SIGIR'93 Conference*, pages 249-257.

[13] Jog, N and Shneiderman, B. (1994), Interactive Smooth Zooming of an Information Visualization. *Technical report CAR-TR-714, CS-TR-3286, ISR-TR-94-94*, University of Maryland.

[14] Johnson, B., Shneiderman, B. (1991), Tree-maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures, *Proceedings IEEE Visualization'91*, pages 284-291.

[15] Kim, H., Korth, H., Silberschatz, A. (1988), PICASSO: A Graphical Query Language, *Software – Practice and Experience*, Vol 18(3), 169-203.

[16] Mackinlay, J., Robertson, G., Card, S. (1991), The Perspective Wall: Detail and Context Smoothly Integrated, *Proceedings of ACM CHI'91: Human Factors in Computing Systems,* pages 173-179.

[17] Robertson, G., Card, S., Mackinlay, J. (1989), The Cognitive Coprocessor Architecture for Interactive User Interfaces, *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology'89*, pages 10-18.

[18] Robertson, G., Mackinlay, J., Card, S. (1991) Cone Trees: Animated 3D Visualizations of Hierarchical Information, *Proceedings of ACM CHI'91: Human Factors in Computing Systems,* pages 189-194.

[19] Robertson, G., Card, S., and Mackinlay, J. (1993), Information Visualization Using 3-D Interactive Animation, *Communications of the ACM 36, 4*, pages 56-71.

[20] Shneiderman, B. (1994), Dynamic Queries for Visual Information Seeking, *IEEE Software* (November 1994).

[21] Smith, S., Bergeron, D., Grinstein, G. (1990), Stereophonic and Surface Sound Generation for Exploratory Data Analysis, *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, pages 125-132.

[22] Spoerri, A. (1993), InfoCrystal: A visual tool for information retrieval & management, Proceedings ACM Conference on Information & Knowledge Management'93, Washington D.C.

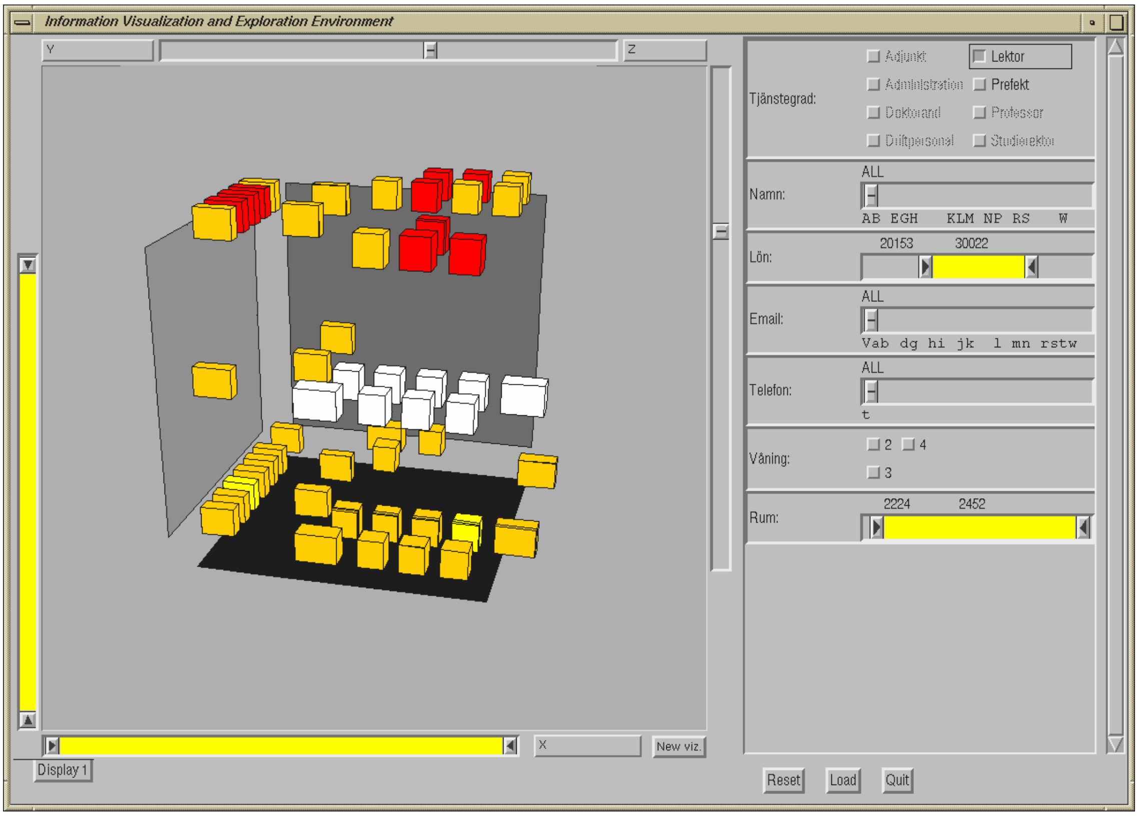
[23] Swayne, D. F., Cook, D., Buja, A. (1992), User's Manual for XGobi, a Dynamic Graphics Program for Data Analysis, Bellcore Technical Memorandum.

[24] Thomée, S., Allwood, C-M. (1995), Usability and database search at employment agencies, forth-coming SSKKII Technical Report, Göteborg University, Sweden.

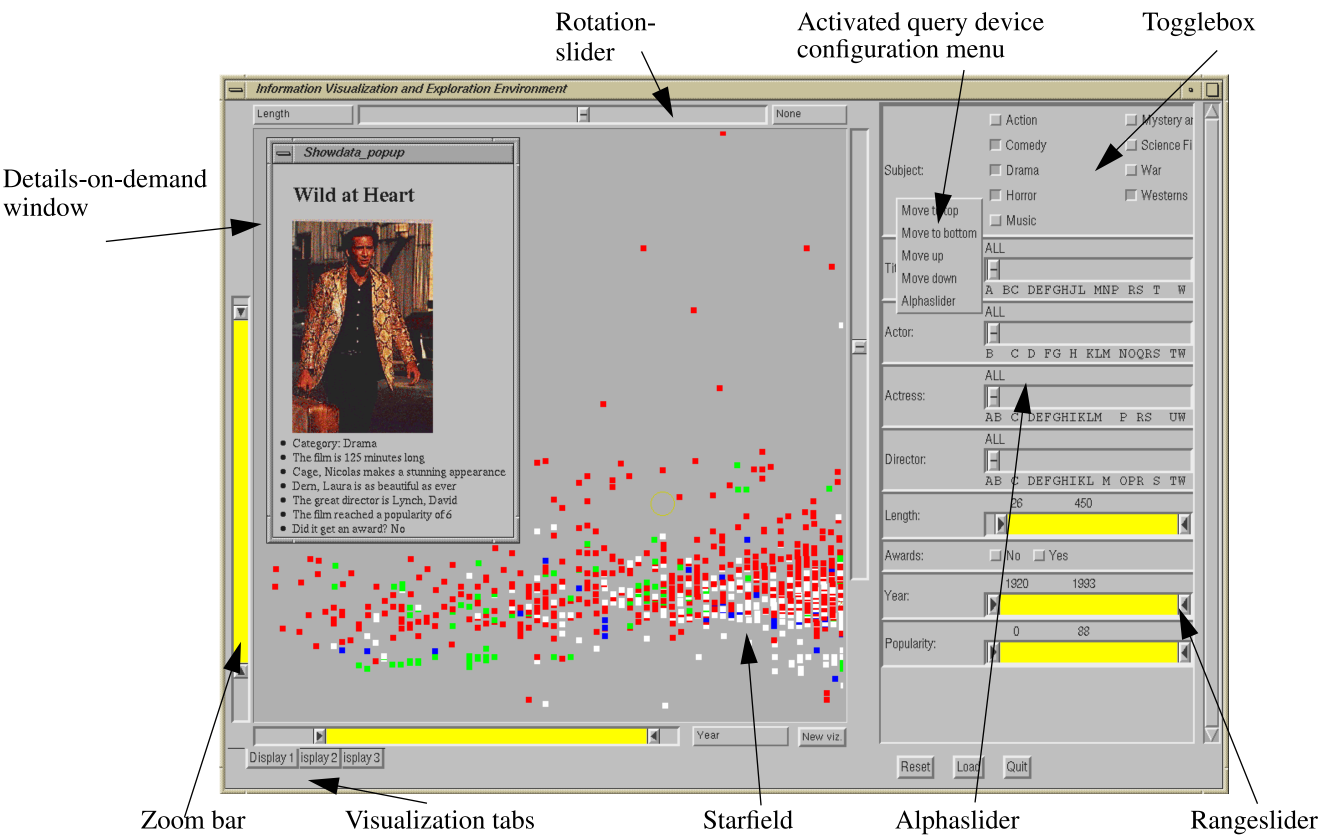
[25] Tufte, E. (1983), *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut, pages 197.

[26] Williams, M. (1984), What makes RABBIT run? *International Journal of Man-Machine Studies 21*, pages 333-352.

[27] Williamson, C., Shneiderman, B. (1992), The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system, *Proceedings ACM SIGIR'92 Conference*, pages 339-346, 1992.



Ahlberg & Wistrand, Figure 5: Dynamic queries interface created with IVEE, allowing users to browse some of the employees of the computer science department at Chalmers.



Ahlberg & Wistrand, Figure 6: Reproduction of the FilmFinder in [2], created with IVEE.

IVEE: An Information Visualization & Exploration Environment

Christopher Ahlberg & Erik Wistrand

Department of Computer Science and SSKKII
Chalmers University of Technology
S-412 96 Göteborg
Phone: +46 31 772 5410
Email: {ahlberg,wistrand}@cs.chalmers.se

To be published in Proceedings of IEEE Viz'95

SSKKII
c/o Department of Linguistics
Göteborg University
S-412 96 Göteborg
Sweden

IVEE: An Information Visualization & Exploration Environment

Christopher Ahlberg & Erik Wistrand

Department of Computer Science and SSKKII
Chalmers University of Technology
S-412 96 Göteborg
Phone: +46 31 772 5410
Email: {ahlberg,wistrand}@cs.chalmers.se

Abstract

The Information Visualization and Exploration Environment (IVEE) is a system for automatic creation of dynamic queries applications. IVEE imports database relations and automatically creates environments holding visualizations and query devices. IVEE offers multiple visualizations such as maps and starfields, and multiple query devices, such as sliders, alphasliders, and toggles. Arbitrary graphical objects can be attached to database objects in visualizations. Multiple visualizations may be active simultaneously. Users can interactively lay out and change between types of query devices. Users may retrieve details-on-demand by clicking on visualization objects. An HTML file may be provided along with the database, specifying how details-on-demand information should be presented, allowing for presentation of multimedia information in database objects. Finally, multiple IVEE clients running on separate workstations on a network can communicate by letting one user´s actions affect the visualization in an another IVEE client.

1 Introduction

The cognitive load while performing information retrieval tasks is usually high. Finding the right query formulation which will deliver a query result with high result precision is cognitively difficult. The benefit of static visualizations of data sets has long been known, visualizations of for example demographic, geographic, and economic data sets are commonplace today [25][9]. Allowing users to incrementally control animated visualizations of databases, queries, and query results can minimize the mental effort needed for:

- Finding the right query formulations
- Grasping relations between queries and query results
- Judging relations between individual query results.

Benefits of animated visualizations partially stem from the fact that we can perform many perceptual tasks such as detection of patterns and anomalies in pictures with little conscious mental effort. Not until recently has the use of these remarkable perceptual abilities found its way into information retrieval systems [7][1][12][19].

Visual representations of databases allows for presentation of many more database elements in single screens than traditional text based methods. This in turn allows for overviews in a way which is not possible with text based query systems. A visualization might present thousands of elements in one single screen, while a text based system is limited to the magnitude of tens of elements.

1.1 Dynamic queries

Dynamic queries is a concept for information exploration and database querying [1], defined as:

”a user controlled animated visualization of a query process, including databases, queries, and query results.”

Using dynamic queries, users manipulate query devices (e.g. rangesliders, toggles, and alphasliders) of various kinds to construct queries to incrementally update (with near immediate updates) a visualization of the current query result. Users might increase the value of a query parameter to see how the increase affects the size (recall) of the result set. The relevance (precision) of the result set can be judged quickly from the color of the visualization objects (i.e. the result set), or from the location of the result set in the visualization (indicating for example geographic proximity or some other high level semantic property). Users might also manipulate query devices to explore trends and patterns in data, or to detect anomalies. This is usually done by manipulating for example a rangeslider to observe if excluding a certain part of a query range correlates with visualization objects disappearing in some particular pattern. The power of dynamic queries compared to other query methods, such as form fill-in and natural language has been confirmed in controlled experiments [1][27].

1.2 Other visual query techniques

The Information Visualizer from Xerox Parc utilizes visualizations such as Cone Trees and the Perspective Wall [16][18]. 3D visualizations allow users to detect clusters in data sets and search for information in context of the whole database. Hjemme *et.al* [12] presents the LyberWorld system which holds a visualization controlling a full text retrieval process, partially based on Cone Trees. They emphasize the use of visualizations for perceiving hidden information carried by relations in complex data structures. Spoerri [22] introduced the InfoCrystal, with a visualization of the subresults from a complex query, and their relations to the query terms. By inspecting the visualization, users can quickly judge the relevance of the subqueries. Other interesting examples of visualization systems are XGobi [23] and the AT&T Data Visualization Sliders [11].

1.3 Dynamic queries examples

A number of interesting prototype dynamic queries applications has been built for experimental purposes, such as a dynamic periodic table [1] and a dynamic homefinder [27]. These prototypes depended on domain specific visualizations such as geographic maps and the periodic table of elements. The FilmFinder [2] extended the scope of dynamic queries by introducing the starfield display where two ordinal variables from a database relation are plotted against each other and used as an interactive visualization.

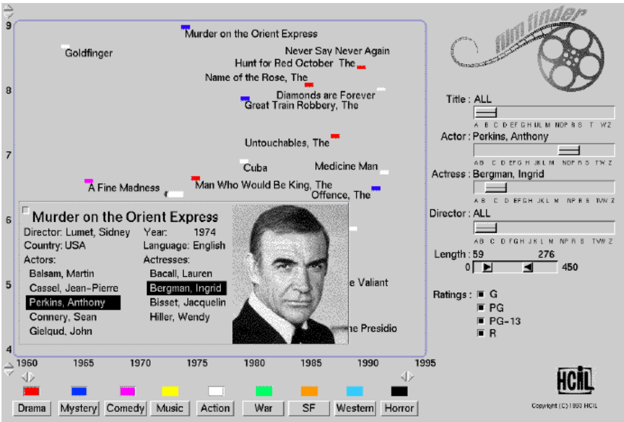


Figure 1: The FilmFinder dynamic queries prototype [2].

2 IVEE

Today, a form fill-in module is provided with most commercial database management systems (DBMS). The form fill-in module allows users to easily create forms for both entering data into and querying a database. Similar to a form fill-in module, a dynamic queries system could be provided with a DBMS. The dynamic queries system would allow for automatic creation of a visual query environment from a database relation.

The Information Visualization and Exploration Environment (IVEE) is an attempt at such a system [1]. IVEE can automatically create dynamic queries environments holding query devices and visualizations (Figure 3). IVEE imports relations with named attributes, given on a straightforward text format. The relations might originate from database systems such as Oracle or spreadsheet programs such as Microsoft Excel.

An important challenge for us was to design IVEE so that users could get started with their visualization and exploration tasks with as little effort as possible. Obviously this is a generic goal for most software developed today. However, visualization software is often quite complex and requires users to go through many steps before the actual exploration process can start. We want to demonstrate how a visualization system can be designed to automate the task of creating both visualizations and manipulation objects and let users start working directly with their high level exploration tasks.

Users of IVEE may use the tool at different levels:

- IVEE is used to explore a data set only using the standard visualizations (starfields) and the automatically created query devices.
- IVEE is used with modified visualizations and query devices, with the configuration provided along with the database (possibly created by another user).
- IVEE is used both to explore a data set and to interactively update visualizations (attaching graphical objects to database objects, varying color schemes) and query devices (changing query devices used for database querying, changing the layout).

IVEE examines the data in a user specified relation and classifies it into datatypes (integers and strings) and size (total number of values held in the attribute and number of distinct values held in the attribute). The data is stored in an internal IVEE database object (Figure 2). Attributes are stored as straightforward vectors, with multiple indexing to increase performance for various search tasks.

The user interface of IVEE holds two main components, a query area holding a number of query devices (right area in Figure 3) and a visualization area (left area in Figure 3), holding a number of visualization. These user interface objects are reflected in the architecture of IVEE in Figure 2. Notice that IVEE can hold several instantiations of visualizations, databases, and query areas.

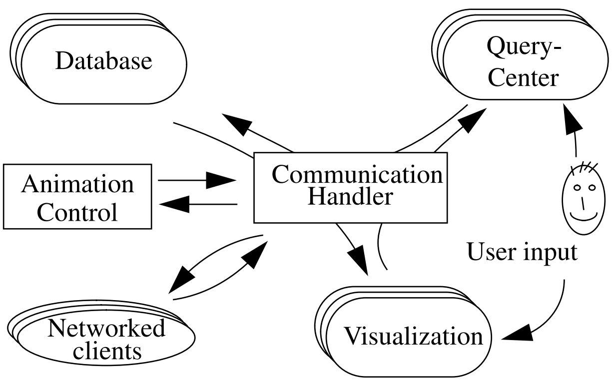


Figure 2: The IVEE architecture.

3 Visualizations in IVEE

Another design goal for IVEE has been to provide users with a rich collection of visualizations. Ours and many others' work on interactive visualization seem to point to that successful visualization environments does not depend on one single powerful visualization, quite contrary a whole smörgåsbord of visualizations appropriate for various tasks and datatypes is closer to a successful solution. Our approach depends on a simple, yet powerful architecture. To each object in a database relation a (possibly user defined) graphical object is attached. This graphical object may be a simple point of light or square as used in starfield visualizations [2], a glyph from a predefined library to create visualizations like those in [21], or arbitrary graphical

1. IVEE is available from ftp.cs.chalmers.se:/pub/IVEE. See also the WWW-page http://www.cs.chalmers.se/SSKKII/ivee.html

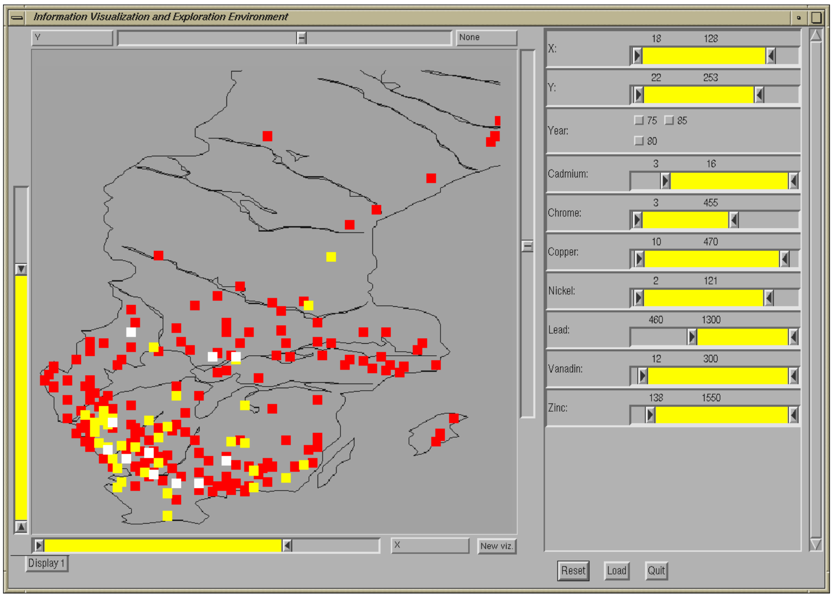


Figure 3: Example of a dynamic queries environment created with IVEE, in this case an environmental database browser. Users can browse and query a database of thousands of measurements of heavy metals in Sweden.

objects in two or three dimensions to create full-fledged interactive visualizations of objects such as buildings, airplanes, or abstract objects such as Cone Trees [18] and Tree-maps [14]. In the spirit of our goal to make IVEE easy to get started with, users can start by specifying nothing about polygons connected to database objects at all. This will associate a square with each object. When user defined polygons are specified this is done by providing them in a file along with the database. The file might contain one unique graphical object for each database object or a smaller number of polygons and a mapping from database objects to graphical objects. For a database of cars, motorcycles, and trucks three complex graphical objects might be specified, one for each type of database object.

To achieve the high performance graphics necessary for dynamic queries and interactive 2 and 3D graphics, IVEE utilizes an animation loop which adapts rendering strategies to the current performance of the computer and user behavior. Examples of rendering approximations are:

- Render objects at a lower resolution.
- Render shaded objects as wireframe models.
- Skip textual labels.
- Do not fully redraw the screen while performing expensive queries.
- Store multiple views of the same graphical object.

IVEE offers a number of visualizations where the basic one is the starfield [2]. A starfield is an interactive scatterplot with additional features for zooming, panning, details-on-demand, etc. Two ordinal variables from a database relation are chosen as the axes in the starfield (Figure 4).

The power of the starfield as a visualization of complex databases is that it allows for displaying a large number of database objects in one single screen. Each object is represented by a small graphical object which can be coded by color, brightness, shape, size, etc. The spatial location of an element can effectively encode two properties of a database object. The starfield provides important qualities of visualizations such as:

- The overview provides starting points for search.
- Query results may be presented in the context of the full database.
- Query result relevance can be judged quickly.
- Trends and anomalies can be explored effectively.
- Feedback can be provided continuously during queries.
- Works for many datasets.

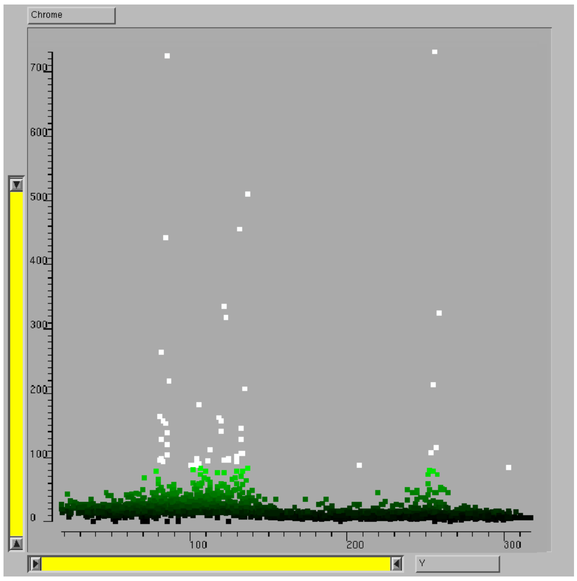


Figure 4: Starfield visualization of the same environmental database as in Figure 3. Users can zoom, pan, query, select-details-on-demand and interactively change the attributes that are visualized in the starfield.

In addition to allowing each database object be associated with a square in a starfield, IVEE supplies users with several other possibilities for coding properties of database objects into the starfield. Visualization elements can be colored using two different schemes – users might associate database values of an attribute with either:

- Colors of different hues (blue, yellow, red, etc.) which is useful for coding nominal attributes.
- Colors of the same hue, but shifting in brightness which is useful for coding continuous attributes.

When utilizing the latter scheme, users can specify a certain part of the range of an attribute to be colored differently – useful for indicating for example objects with values in the top 5% range. Users can also associate glyphs from a predefined library to nominal attribute values which can effectively complement the use of color.

3.1 Geographic visualizations

The immediate extension of the starfield is to provide a background map which can be used to create geographic visualizations. IVEE users can specify such a visualization context by providing background objects holding a number of polygons defining an appropriate background. Background objects are not attached to any database object.

Background objects might be geographic maps, but may also very well be a structural drawing of a house. An important property of geographic visualizations compared to for example starfields is that they provide important and often well-known context for the presentation of query results. As background objects as well as other graphic objects are provided on a vector format, arbitrary zooming can be performed smoothly.

3.2 Other visualizations

So far, the visualizations described have not utilized the possibility of associating arbitrary polygons with database objects. An interesting example where this can be used is node and link diagrams. Similarly to how a geographic map is specified, users can specify a layout of nodes and links, and each graphical object is associated with a database object. By manipulating query devices (elaborated below), users can select nodes of certain types, links for which an associated value is within a given range, etc.

This is not an ideal way of creating node and link diagrams. However, we do think it shows how the simple yet powerful way of associating database objects with complex polygons can be used to create interesting visualizations. Similar visualizations which can be created with IVEE are for example Tree-maps and Cone Trees. Hierarchy visualizations are widespread and we plan to incorporate functionality for handling those more smoothly.

Yet another class of visualizations that can be created are those depending on complex three dimensional objects. In Figure 5 a schematic visualization of a part of the computer science department at Chalmers University of Technology is presented – used in a dynamic queries interface for searching for persons in the department. Background objects as described above make up the outer structure of the building and employees (database objects) are represented by their offices. Manipulating query devices will grey out those rooms not fulfilling the search criteria and allows for detection of trends such as how professors with high salaries are more often located in the top of the building!

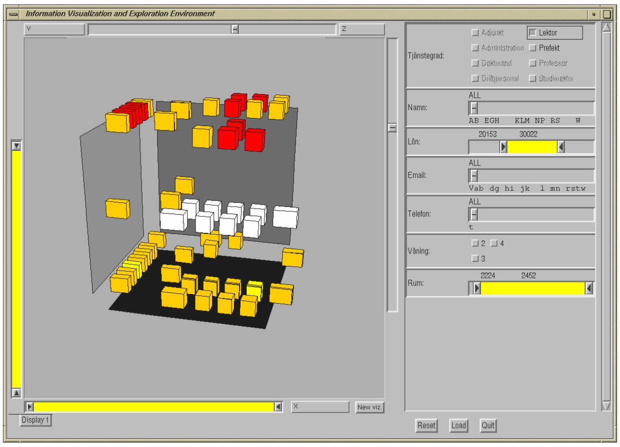


Figure 5: Dynamic queries interface created with IVEE, allowing users to browse some of the employees of the computer science department at Chalmers. Each box in the visualization corresponds to an office of a member of the staff in the department.

3.3 Multiple visualization areas

IVEE allows multiple visualizations to be created simultaneously, and when the query devices are manipulated all the visualizations are updated interactively, allowing for powerful exploration of data. Examples of how this might be used is described in [10]. Users might observe a cluster of points in a starfield. When zooming into the cluster a geographic map is updated next to the starfield, where the same points can be observed to all be situated in the same geographic region.

Further, users might activate more than one page of visualizations simultaneously. A notebook similar system with tab buttons (similar to the ones in Microsoft Windows and Motif) is used. By pushing the”New Visualization” button, a new visualization area is created and the old one stored in the background (Figure 6).

3.4 Visualization manipulation

Users can perform a number of operations on the starfield:

- Zooming
- 3D manipulation
- Panning
- Filtering
- Selection of details-on-demand

In this section zooming, 3D manipulation, and panning is discussed. Filtering and details-on-demand will be discussed below. Zooming is performed either with the mouse buttons (second mouse button to zoom in and third mouse button to zoom out) or the zoom bars (Figure 6)[13]. Manipulating the zoom bar on either the X or the Y axis of the visualization will either increase or decrease the size of the area in view, i.e. zooming is performed in either of the dimensions. Mouse button zooming allows for zooming in both dimensions simultaneously, while the zoom bars are appropriate for zooming each dimension separately.

Both zooming methods are useful, the former for example when there is an obvious spatial relation in the visualization which should be preserved, such as in a geographic map, and the latter when no such spatial relation exists, for example in a scatterplot. By dragging the area between the sliderthumbs in the zoom bars, users can pan the visualization without changing the zoom rate.

To rotate a 3D visualization users manipulate two sliders (Figure 6) to rotate the view around each of the X and Y spatial dimensions, similar to the slider approach described in [8]. Zooming in 3D is performed with the mouse buttons, as described above. However, the zoom bars are not enabled when manipulating a 3D visualization, as there is no direct mapping from a 2D range to a perspective view with the eye placed at an arbitrary viewpoint.

4 QUERY DEVICES IN IVEE

Based on the classification of the user specified relation IVEE selects query devices for each attribute. Query devices are selected from rangesliders, alphasliders [3], and toggles (all shown in context in Figure 6). Simple rules decide which query devices are assigned to which attributes. For example, for a string attribute with more than 10 distinct items, an alphaslider is selected, otherwise a group of toggles with a toggle for each distinct item is selected (of course, the threshold 10 can be changed). For integers and reals, rangesliders are selected, unless only 10 distinct items exist in the attribute – then a group of toggles is selected.

A different strategy would be to allow users to manually select which widget to use for each attribute. However, while this strategy might lead to better specifications of query environments, it would be to cumbersome for large databases with many attributes. Also, users can change the widget used for an attribute interactively while exploring a database. This is performed by activating a menu attached to the query device currently used for the attribute (Figure 6). From the same menu users can move the query device up, down, to the top, and to the bottom of the query area. Regrouping widgets in another order than the one provided implicitly by the order of the attributes in the database relation is quite useful, especially for relations with a large number of attributes.

4.1 Rangesliders

Rangesliders are useful for selecting range criteria for integer attributes and other attributes with an order relation (Figure 6). The rangeslider provides a natural representation of the query it is representing, i.e. a range query. Increasing or decreasing the range of an attribute allows for powerful exploration of trends and anomalies. Rangesliders are also very effective for relaxation of query parameters when the result set can be observed immediately in a visualization.

4.2 Alphasliders

The alphaslider is a widget for selecting items from long lists of for example strings [3]. The rationale behind the alphaslider is its small size - it allows for selection from lists of thousands of elements in a small screen area Figure 6. Screen area is precious in a visual query system where most screen space should be used for the visualization of data. The alphaslider can not only be used for selection of specific strings, it is also very useful for browsing categorical variables while observing the query result set in a visualization. For example in the FilmFinder [2] this could be used for browsing the directors attributes while observing result sets showing up in the upper corner of the display, indicating films to be new and popular.

4.3 Toggles

Toggles are the last of the widgets currently used for query formulation in IVEE (Figure 6). Toggles are useful when only a few alternatives exist for an attribute and these alternatives should be presented on the screen explicitly. Users may select multiple values for an attribute with a group of toggles.

4.4 Query evaluation and tight coupling

Queries are composed from the conjunction of all the query components defined by the query devices. Query devices in their initial state let through all database objects, i.e. they do not affect the result of the query. Manipulating a device immediately affects not only the visualization, but also the other query devices – the query mechanism is tightly coupled [4]. A query device manipulation restricts the query range of the other query devices to only include criteria existing in the remaining elements of the database. This useful for two purposes:

- To guide querying when searching for specific elements in the database – for example after having selected Ingemar Bergman with the director slider for a film database only actresses appearing in an Ingemar Bergman film will be selectable with the actresses slider.
- To facilitate browsing of categorical variables. A typical use of the alphaslider is to browse a categorical variable, such as sample number for a statistical data set. After having selected a subset of the samples with some other query device, users can browse the samples – without looking at the alphaslider – and focus on the visualization. When an interesting pattern is found in the visualization users can look at the alphaslider to find out the identification of the particular sample. This strategy is not nearly as useful if most of the alternatives on slider cause an empty visualization, i.e. an empty query result.

5 Details-on-demand

Equally important to users being provided with effective overviews of data is that they are able to retrieve full and rich descriptions of specific database elements. Only showing details when they are requested is instrumental for the concept of dynamic queries. To draw the objects in for example a starfield visualization only requires IVEE to access two or three attributes in each potentially very large database object. IVEE users select details-on-demand by clicking on a visualization object with the left mouse button

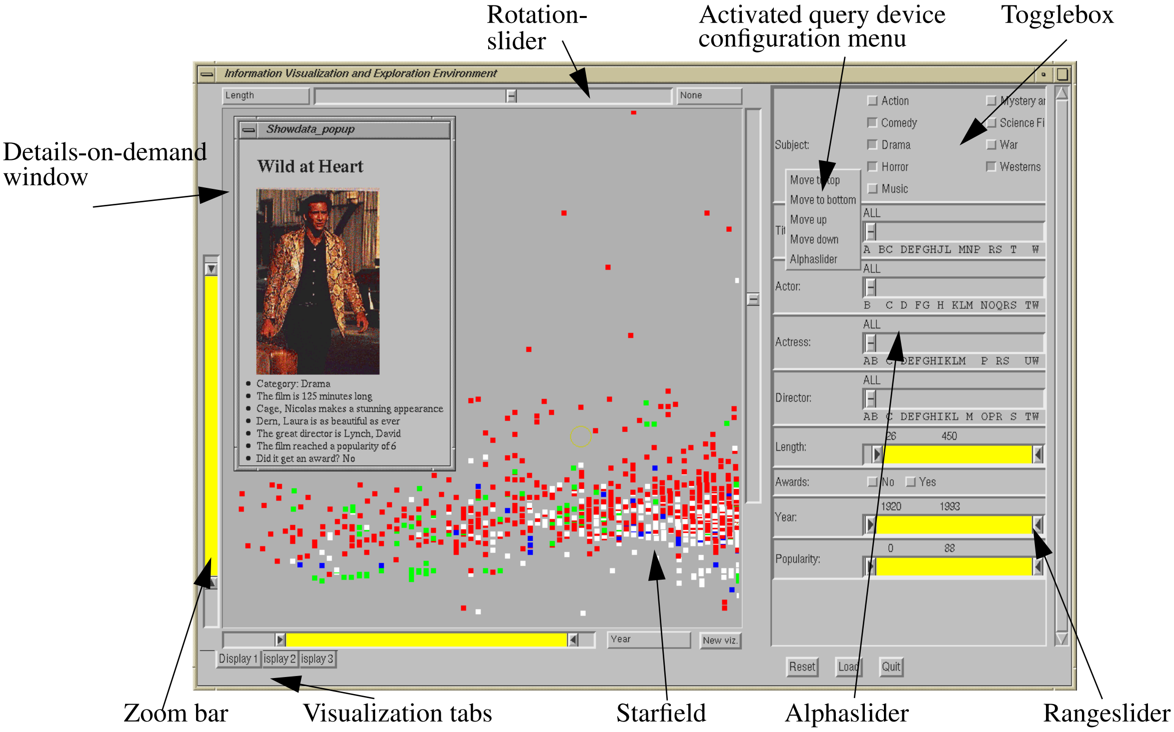


Figure 6: Reproduction of the FilmFinder in [2], created with IVEE.

(Figure 6). Upon moving the mouse cursor over a selectable object feedback is provided indicating its selectability. In addition to the "ordinary" data in a database object consisting of integers, reals, and strings, IVEE can hold file pointers to multimedia objects, such as images, speech, video, etc., on standard formats such as GIF and MPEG.

IVEE has a predefined approach to presenting data in database objects when they are selected. However, users can create much richer presentations by providing a HTML-based formatting document. IVEE utilizes the public domain HTML-widget in the NCSA Mosaic distribution. Along with a database, users may supply a file like the one in Figure 7. This is the standard HTML format, except for the placeholders indicating where database attribute values should be filled in. The placeholders consist of a ´#´ character followed by the attribute name. The HTML file may contain pointers to images, sound, video, and other local HTML files.

```
<head>
<title>Film description</title>
<body>
<h1>#Title</h1>
<img src="#Image">
<ul>
   <li> Category: #Subject
   <li> The film is #Length minutes long
   <li> #Actor makes a great appearance
   <li> #Actress is as beautiful as ever
   <li> The great director is #Director
   <li> Popularity: #Popularity
   <li> Did it get an award? #Awards
</ul>
```

Figure 7: HTML description of the details-on-demand window in Figure 6.

Being able to receive details-on-demand is not only important for examining individual database objects, it is also useful as a starting point for search. The overview provided by for example a starfield helps users overview potentially very large amounts of data. However, a complementary approach to starting a search from the overview is to start with a specific database object which might have been found in the visualization or by initially specifying a strict set of criteria leaving one or a few objects selected. This approach has the advantage of not overloading users with a large abstract set of complex data. Instead they can apply their knowledge of one specific element [26].

IVEE allows for this approach to searching through the details-on-demand popup-window. Users can click with the left mouse button on any of the boldface texts in the window, and thereby setting the value of the appropriate query device to the marked value.

6 Multiple databases

The initial assumption of IVEE is that the database consists of one single relation. For databases consisting of several relations this constraint can sometimes be overcome by creating a universal relation which allows a whole set of relations to be regarded as one single relation [15]. However, this approach is not always ideal, and accordingly users can load several database relations (sharing at least two attributes which can be visualized) simultaneously into IVEE. Each relation is given its own dedicated set of query devices, and each attribute of relation is given its own dedicated query device – i.e. IVEE does not attempt to join similar attributes which appear in several relations (although this might in some cases be useful).

Allowing several database relations to share the same manipulable visualization creates a powerful external representation for tasks such as the matching of objects in the respective relations. A good example is two relations, one holding job opportunities and one holding job applicants. Both might be visualized in for example a map or a starfield, and users (e.g. job agency employees) can manipulate query devices for both relations to find reasonable subsets and then visually scan the visualization to find matching opportunities/applicants.

7 Distributed exploring

In many interesting application areas for systems such as IVEE there are many people interested in the same data set. Above the task of matching of jobs and skills was mentioned, an other example might be analysis of nearly any statistical data set. Unless these people are at the same site, or even in the same corridor, they are most likely to run into the situation of having to discuss querying and analysis of their data set over the telephone. For the job/skill matching situation which we have examined extensively, a common situation is that employment agency employees have to discuss cases (persons, jobs) over the phone with their colleagues [24]. The traditional text based system currently used does not offer any help for this situation.

IVEE allows several persons to visualize and explore the same data set simultaneously. By starting several (at least two) clients of IVEE, loaded with the same database, and then requesting these to be connected through a UNIX socket (Figure 2), several users can manipulate the same visualization. Each client is responsible for the actual visualization and the corresponding local database, but operations such as querying, zooming, details-on-demand, etc., are distributed to all other connect clients. Only transmitting user actions and not actual query results allow this approach to be possible even without a high speed connection. The scheme allows several people to share a common view of a data set while communicating over a traditional communication system, e.g. the telephone. Currently we have only a rough implementation of socket communication between IVEE clients and many problems remain to be solved such as how several cursors can be managed, what to do when two users simultaneously perform contradicting actions, etc.

8 FUTURE WORK

We plan to continue to develop IVEE, and extend its functionality with for example:

- More query devices. The existing ones can be extended in functionality, but new ones need to be introduced to cover other kinds of queries than those described above.
- More types of visualizations. Hierarchies of various kinds are common and a large number of hierarchy visualizations exists. We want to extend the support in IVEE for those.
- Arbitrary boolean combinations of query components. Currently the query mechanism only supports AND:ing of all query components. For some situations it would be meaningful to be able to create more complex queries.
- Visualization wizard. Users of IVEE are faced with a large number of choices for visualization configuration parameters (today manipulated in a popup window). We would like to battle this problem by introducing a visualization wizard similar to the graph wizard in Microsoft Excel which can provide structure.
- Our current HTML document support does not extend to communication with world wide web (WWW) servers beyond our own department. If such functionality was introduced, IVEE could be an excellent base for visualization and interaction with the WWW.
- An obvious constraint on a tool such as IVEE is the size of the database it can handle. IVEE can currently handle a database consisting of 5000-6000 objects with some 10-15 attributes each running on a SGI Indy 100MHz machine. A necessary area of further research is to explore datastructures for pushing this to 50-100’000 objects.
- Another typical problem in a tool based on scatterplots for displaying data is overlapping points which hold exactly the same X and Y values in the display.

9 CONCLUSIONS

IVEE is an interactive visualization and query system based on the concept of dynamic queries. The IVEE architecture is based on the simple concept of attaching more or less complex graphical objects to database objects in visualizations. Graphical objects might be simple colored points of light, glyphs selected from a predefined library, or arbitrary sets of polygons in two and three dimensions. This allows IVEE users to fairly easily import database relations and create complex visualizations. The visualizations can be interactively queries, filtered, zoomed, and panned.

A major research area for our group right now is to look at how dynamic queries techniques can be utilized in the situation of matching jobs and skills. IVEE has allowed us rapidly prototype dynamic queries interfaces for evaluating visualizations for this job/skill matching application (starfields, geographic maps, and text output) and explore techniques for guiding query composition in direct manipulation systems [4].

Other applications of IVEE have been to create a visualization system for exploring data on spoken language, used by researchers in linguistics [6], and a system for browsing environmental data – as shown in Figure 3. The Human-Computer Interaction Laboratory at University of Maryland are currently using IVEE for their projects.

We believe that it is instrumental for work on visualization and information exploration to be close to end-users and real applications. New technology such as large high resolution screens, fast graphics, new input devices, and high capacity network connections makes large promises for exiting information visualization applications. Working with end-users and their problems helps us find meaningful use of these new technologies and also helps us push our own creativity further.

10 ACKNOWLEDGEMENTS

This work was in part supported by NUTEK, grant no: 5321-93-2760, and Arbetsmiljöfonden, grant no: 94-0525.

The authors want to thank Staffan Truvé, Jens Allwood, and Johan Hagman for fruitful discussions during the development of IVEE. Ann Rose and Catherine Plaisant at the HCIL, University of Maryland, has provided us with valuable comments and proposals sprung out of their use of IVEE. We also want to thank Ben Shneiderman for continued support and encouragement in the work with dynamic queries.

REFERENCES

[1] Ahlberg, C., Williamson, C., Shneiderman, B. (1992), Dynamic Queries for Information Exploration: An Implementation and Evaluation. *Proceedings ACM CHI'92: Human Factors in Comp. Systems, pages 619-626*. Also in Shneiderman, B. (Ed.), *Sparks of Innovation in Human-Computer Interaction*, Ablex, Norwood, N.J, 1993.

[2] Ahlberg, C., Shneiderman, B. (1994), Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. *Proceedings ACM CHI'94: Human Factors in Comp. Systems*, pages 313-317. Also in Baecker, R., Grudin J., Buxton, W., and Greenberg, S., *Readings in Human-Computer Interaction: Toward the Year 2000* (2nd Edition), Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[3] Ahlberg, C., Shneiderman, B. (1994), The Alphaslider: A Compact and Rapid Selector. *Proceedings ACM CHI'94: Human Factors in Comp. Systems*, pages 365-371.

[4] Ahlberg, C., Truvé, S. (1995), Tight Coupling: Guiding User Actions in a Direct Manipulation Based Information Retrieval System, Proceedings of EHCI'95: Engineering for Human-Computer Interaction, C. Unger, ed., IFIP Transactions series, Chapman & Hall.

[5] Ahlberg, C., Wistrand, E. (1995), IVEE: An Environment for Automatic Creation of Dynamic Queries Applications, *Proceedings of ACM CHI'95: Human Factors in Comp. Systems*.

[6] Allwood, J., Ahlberg, C. (1995), Visualizing Spoken Interaction, *Proceedings of the 15th Scandinavian Conference of Linguistics*, Oslo University.

[7] Card, S., Robertson, G., Mackinlay, J. (1991), The Information Visualizer, an Information Workspace, *Proceedings ACM CHI'91: Human Factors in Comp. Systems,* pages 181-188.

[8] Chen, M., Mountford, J., Sellen, A. (1988), A Study in Interactive 3-D Rotation Using 2-D Control Devices, *Proceedings ACM SIGGRAPH'88*, pages 121-129.

[9] Cleveland, W. (1994), *The Elements of Graphing Data*, Hobart Press, Summit N.J. 297 pages.

[10] Cleveland, W. (1993), *Visualizing Data*, Hobart Press, Summit N.J., 360 pages.

[11] Eick, S. (1994), Data Visualization Sliders, *Proceedings ACM SIGGRAPH Symposium on User Interface Software and Technology'94 proceedings*.

[12] Hemmje, M., Kunkel, C., Willet, A. (1993), LyberWorld – A Visualization User Interface Supporting Fulltext Retrieval, *Proceedings ACM SIGIR'93 Conference*, pages 249-257.

[13] Jog, N and Shneiderman, B. (1994), Interactive Smooth Zooming of an Information Visualization. *Technical report CAR-TR-714, CS-TR-3286, ISR-TR-94-94*, University of Maryland.

[14] Johnson, B., Shneiderman, B. (1991), Tree-maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures, *Proceedings IEEE Visualization'91*, pages 284-291.

[15] Kim, H., Korth, H., Silberschatz, A. (1988), PICASSO: A Graphical Query Language, *Software – Practice and Experience*, Vol 18(3), 169-203.

[16] Mackinlay, J., Robertson, G., Card, S. (1991), The Perspective Wall: Detail and Context Smoothly Integrated, *Proceedings of ACM CHI'91: Human Factors in Computing Systems,* pages 173-179.

[17] Robertson, G., Card, S., Mackinlay, J. (1989), The Cognitive Coprocessor Architecture for Interactive User Interfaces, *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology'89*, pages 10-18.

[18] Robertson, G., Mackinlay, J., Card, S. (1991) Cone Trees: Animated 3D Visualizations of Hierarchical Information, *Proceedings of ACM CHI'91: Human Factors in Computing Systems,* pages 189-194.

[19] Robertson, G., Card, S., and Mackinlay, J. (1993), Information Visualization Using 3-D Interactive Animation, *Communications of the ACM 36, 4*, pages 56-71.

[20] Shneiderman, B. (1994), Dynamic Queries for Visual Information Seeking, *IEEE Software* (November 1994).

[21] Smith, S., Bergeron, D., Grinstein, G. (1990), Stereophonic and Surface Sound Generation for Exploratory Data Analysis, *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, pages 125-132.

[22] Spoerri, A. (1993), InfoCrystal: A visual tool for information retrieval & management, Proceedings ACM Conference on Information & Knowledge Management'93, Washington D.C.

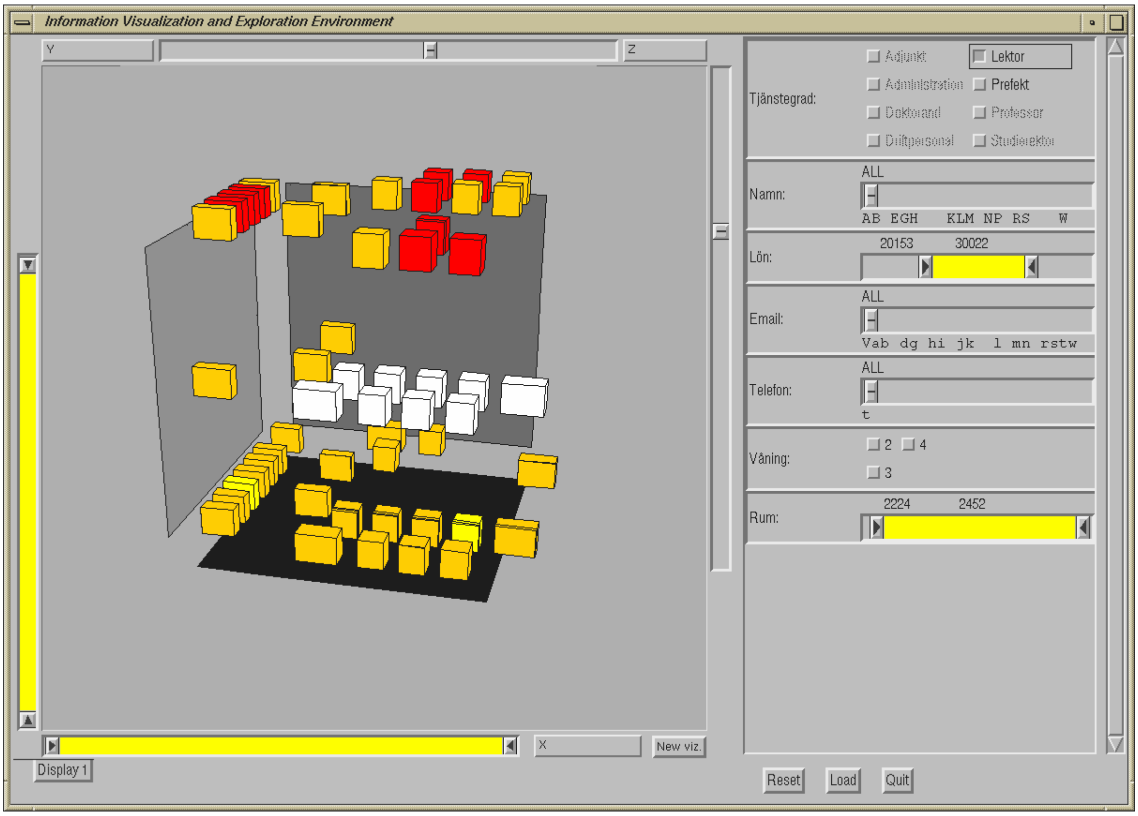
[23] Swayne, D. F., Cook, D., Buja, A. (1992), User's Manual for XGobi, a Dynamic Graphics Program for Data Analysis, Bellcore Technical Memorandum.

[24] Thomée, S., Allwood, C-M. (1995), Usability and database search at employment agencies, forth-coming SSKKII Technical Report, Göteborg University, Sweden.

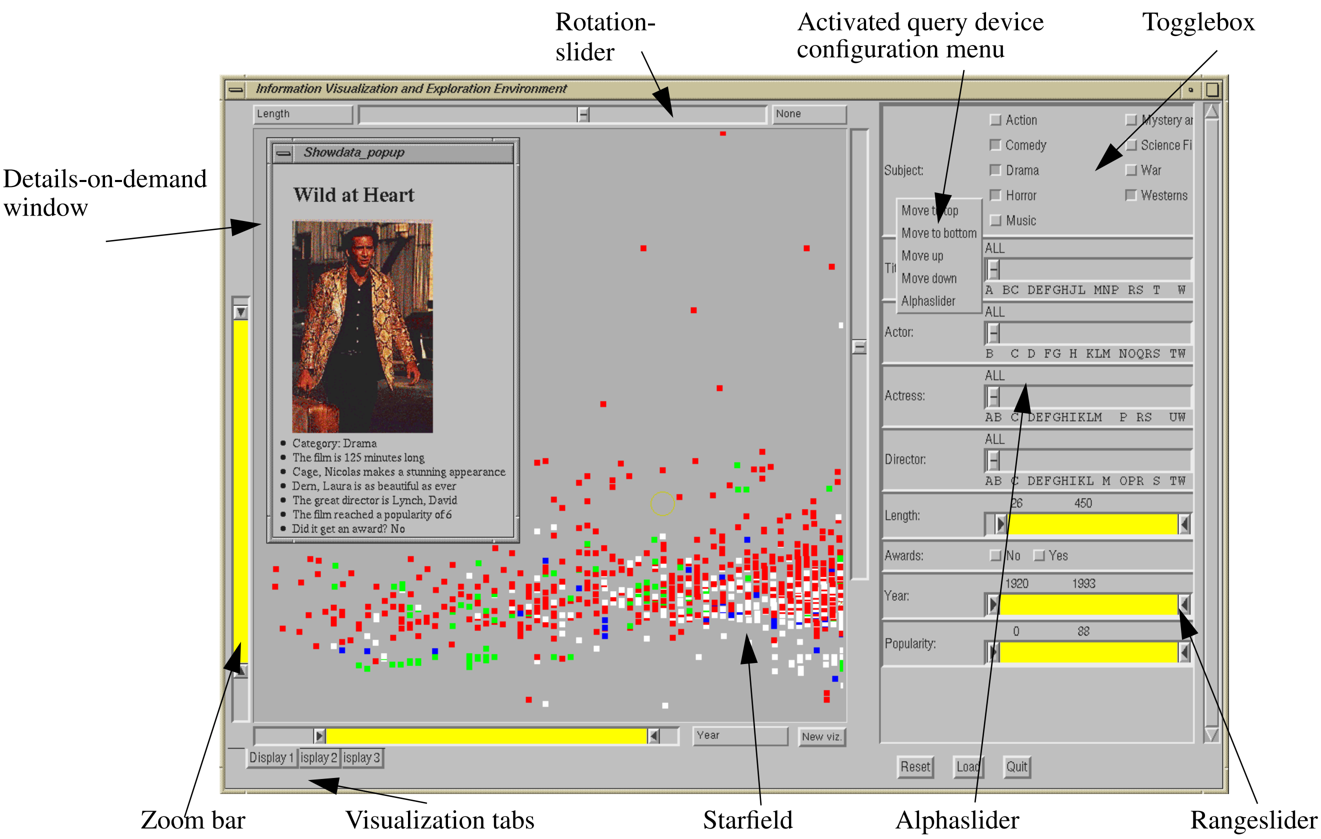
[25] Tufte, E. (1983), *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut, pages 197.

[26] Williams, M. (1984), What makes RABBIT run? *International Journal of Man-Machine Studies 21*, pages 333-352.

[27] Williamson, C., Shneiderman, B. (1992), The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system, *Proceedings ACM SIGIR'92 Conference*, pages 339-346, 1992.



Ahlberg & Wistrand, Figure 5: Dynamic queries interface created with IVEE, allowing users to browse some of the employees of the computer science department at Chalmers.



Ahlberg & Wistrand, Figure 6: Reproduction of the FilmFinder in [2], created with IVEE.