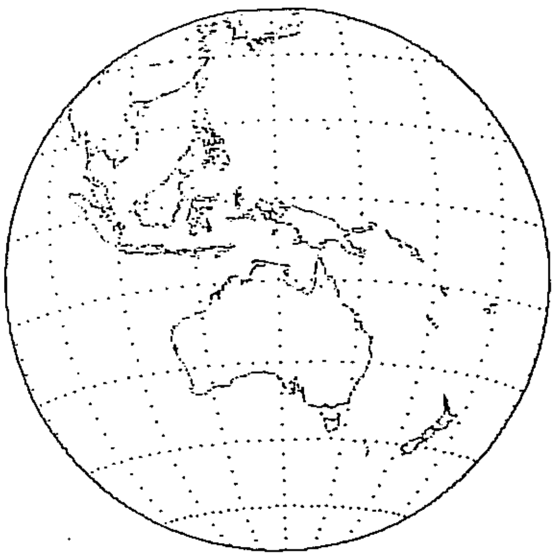
# PROCEEDINGS

THIRD
INTERNATIONAL SYMPOSIUM ON
SPATIAL DATA HANDLING



August 17 - 19, 1988

Sydney, Australia

# Designing Object-Oriented Query Languages for GIS: Human Interface Aspects*

Max J. Egenhofer
Andrew U. Frank
Surveying Engineering Program
University of Maine
Orono, ME 04469, USA
MAX@MECAN1.bitnet
FRANK@MECAN1.bitnet

## Abstract

Spatial information systems deal with large, heterogeneous collections of spatial and non-spatial data which require flexible methods for interactive inquiries and representation. Current database query languages, which are well-suited to treat alphanumeric data, do not reflect the properties of spatial data. Due to these properties, query languages and interfaces for spatial information systems are more complex than purely lexical systems.

In order to visualize the power needed for a spatial query language, a user interface is designed which gives specific considerations to the coexistence of representations, such as graphical renderings and lexical tables. The different properties of spatial and non-spatial data give rise to lexical formulations of queries in combination with references to graphical 'objects' or areas on maps. In particular, immediate reference to objects on drawings, with direct manipulation devices, as a crucial technique must be embedded in the interaction. Variation of graphical presentations by colors, patterns, etc. demands for appropriate tools in the interface to manipulate the presentation of spatial objects. Interface snapshots are used to simulate the interaction, giving prospective users a close impression of the actual interaction during the design phase.

## 1 Introduction

Within the last decade, the design of Geographic Information Systems (GIS) has been explored in a bottom-up manner, focussing on the architecture [Frank 1984a], [Smith 1984], [Aronson 1983], storage and access management [Frank 1983] [Guttman 1984] [Nievergelt 1984] [Tamminen 1982], and data structures. At the same time, only little attention has been paid

*This work was partially funded by grants from Digital Equipment Corporation and NSF under No IST-8609123

to the interface through which a user views such a system, and it is still a common opinion that the interface is "something to be done after the design and the implementation has been completed". From the viewpoint of a product, the interface is one of the most important parts because the success and effectiveness of a product rely on its ease of use. Future Geographic Information Systems must have user interfaces that are easy to learn, appear natural to the user, and are independent of any internal data structures.

The interface of a spatial information system, as the integrating part of all particular applications, hides internal details—how data are stored, composed, or decomposed, etc.—from users and directs the interaction with them. A query language is an essential part of an interface. It enables the users to communicate with the system by articulating their instructions in a form that can be 'understood' by a system. The demands of a query language for a GIS differ significantly from those for query languages of traditional database applications with exclusively alphanumeric data. For instance, a GIS query language must include tools for all the essential operations to inquire about spatial <u>and</u> non-spatial data. The design of query languages for spatial data will benefit from the considerations about interaction between user and system, and the representation of spatial data.

This paper investigates the influence of considerations about the user interface on a spatial query language. It starts with a discussion of human factors in interfaces and their impact on the design of query languages. In chapter 3, dialogue involving spatial data and the representation of spatial data are investigated. Selection by pointing is identified as an essential feature for the interaction with spatial data on maps. Chapter 4 discusses five challenges for a spatial query language. Graphical representation modes give rise to the dynamic composition of maps; content evaluation allows the user to inquire about the composition of a map; colors, patterns, intensity, and symbols are tools to vary the graphical representation; the legend permits the user to interpret the representation; finally, appropriate graphical context is necessary for the understanding of spatial objects in their environment. The paper concludes that interaction and representation must be considered in the design of object-oriented query languages for spatial information systems in order to provide user-friendly communication.

## 2 User Interface Design

The design of user/computer interfaces has been stimulated by the fast innovation of computer hardware within the last decade. Inexpensive raster-graphics and pointing devices have enabled new styles of interaction. With the advent of workstation screens and mice, light-pens, etc., the purpose of renderings on a screen has conceptually changed. Previously, input and output were strictly separated, as in the case of punch cards and hardcopy consoles. Results were presented as static renderings which users could view, and modifications were only possible if the entire rendering was redone. More elaborate techniques promote interactive communication enabling users to manipulate text or drawings directly. Today's interface design is driven by the attempt to make software systems which are faster to learn and easier to use, responding to the needs of the humans who use a computer system.

### 2.1 Human Factors

'Human interface' is the keyword for new techniques and methods which have significantly improved the design of interfaces. Studies under interdisciplinary efforts in Computer Science and Psychology have been concentrating on improving the design of interfaces with computers. The major goal of this work is to accomodate the interaction between users and system as closely as possible to human thinking, rather than enforcing the users to organize their thoughts as dictated by a system. The design of the interface is directed by 'human factors'. A famous design principle resulting from these joint efforts is WYSIWYG. The 'what-you-see-is-what-you-get' concept states that the interface reflects at all times exactly the state of the system, such that the user's view of the system and its actual state coincide. Popular WYSIWYG systems are text editors and spreadsheets which update the interface after each user command.

Another factor that promotes more efficient interaction is direct-manipulation, an advanced way of refering to representations on a screen [Shneiderman 1983]. With the help of direct manipulation devices objects on screen drawings can be selected by pointing at them, or dragged across the screen by moving a mouse with an object tied to it. This technique encourages new interaction concepts which are geared toward 'seeing and pointing' in lieu of 'remembering and typing' [Smith 1983]. Pull-down menus from which users select commands became popular interaction tools liberating users from considerable typing.

### 2.2 A Model of Interfaces

The implementation of user interfaces is influenced by fundamental software engineering aspects. Considerations about independent modification of the application, dialogue, interface layout, or representation are crucial for an adjustable interface. Investigations showed that in today's interactive systems, on average 50-60% of the lines of code of an application are dedicated to dialogue management [Sutton 1977].

The Seeheim model of interfaces [Green 1985] differentiates between three parts of a User Interface Management System: (1) the presentation, (2) the dialogue, and (3) the application interface. If dialogue management is intermixed with other purposes, independent changes of either the application or the dialogue are difficult to achieve, prone to error, and often even impossible.

### 2.3 Query Languages

Structured query languages for question/answer-dialogue dominate the interaction with conventional, purely alphanumeric information systems. Well-known examples are Query-By-Example [Zloof 1977], SQL [Chamberlin 1976], and QUEL [Stonebraker 1976]. The query language is the user's tool to select the data of interest, and most conventional query languages emphasize this issue by providing often complex methods for retrieval of data and formulation of logical constraints upon data. Nevertheless, they lack powerful features like recursion.

Two major deficiencies of the traditional query languages are observed. First, current query languages were designed as interfaces for a specific data model and cannot be easily accomodated to other models: SQL, for example, is the immediate interface for the relational model [Codd 1970] and is dictated by tabular representation of relations, making the intrinsic form of the relational model visible to the user. While this approach is appropriate for applications in which users deal with tabular representation, it is unfit for applications in which users do not have the mental model of tables. Spatial data is normally associated with geometry, map representation, etc. The object-oriented approach has been promoted to be suitable to model complex situations [Dittrich 1986], such as CAD/CAM [Buchmann 1985] [Sidle 1980], office automation [Härder 1985], molecules in chemistry [Batory 1984], and spatial data in GIS/LIS [Frank 1984a]. Object-oriented query languages are expected to enable users to perceive complex objects in a form which is closer to humans' imagination and thinking than tables are.

The second deficit is that traditional languages are geared toward serving both as interactive query language and as query language embedded in an application program. These two contradicting demands—interactive vs. batch—are an impediment when designing user-friendly languages, because most human factors for interaction are conceptually not compatible with batch processing; therefore, in conventional query languages only little consideration is paid to the interaction between the user and the system, and dialogue and representation play a minor role in the language design.

## 3 GIS Interaction

Graphical renderings, such as maps, animate to interpret points, lines, and areas as objects, and use direct-manipulation devices to refer to them as 'objects'. The goal of the interaction with a spatial information system is to use the representation available to the user as reference in upcoming queries.

Subsequently, the requirements for the interaction with and representation of spatial data are investigated. A user interface, tailored to the needs of interaction with spatial objects, is designed. The coexistence of graphical and lexical data leads to a hybrid interaction in which lexically formulated queries can be enhanced by pointing to objects on a drawing. The functionality of an interactive GIS query language must include selection of objects based upon their spatial location, graphical output, and reference to objects on a drawing.

### 3.1 Interface Layout

The layout of the interface must support the user's analysis. Fundamental paradigms are that (1) a user should be able to see the query formulation, and (2) the answer should be presented so that it is visible to the user. Qualitative analysis, which is often based upon visually comparing several situations, is difficult to achieve unless the situations to be compared are visible at a time; therefore, GIS interfaces must be arranged such that one or several maps, as well as a lexical result, are simultaneously visible. Such a layout gives rise to comparing the differences of two representations and inquiring lexical information about a situation on a map. The general desire, to arrange all results so that they are always comparable, is drastically restricted by the limited screen size. It was shown that overlapping windows complicate analyses significantly [Bly 1986],

and rather a tiled interface with a limited number of concurrent windows discourages users from spending their time 'playing' with the window arrangement [Newman 1983].

The interface layout in Figure 1 provides the user with concurrent representation of several results, and control over frequently used language features. Non-overlapping areas for lexical output (1), and graphical output (2), are in the focus of the user. Control mechanisms are arranged around the representations: an area for lexical input (3), control panels for different types of representations (4), and menus to control map-specific characteristics (5) and to manipulate the graphical representation (6).

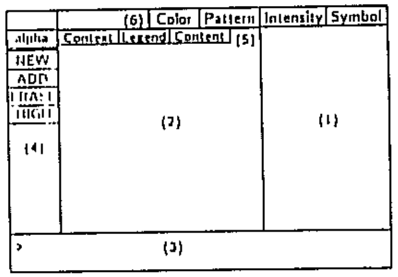


Figure 1: Layout of a GIS interface.

This interface design will be used to visualize the specific GIS components of a query language.

### 3.2 Dialogue

The first phase of a query is the dialogue between user and system during which the query is specified sufficiently to be processed by the system, without any further assistance from the user. The formulation of the query is based on either the user's knowledge, or information which was provided from the system. A map on a graphical display, for example, provides a large variety of information which the user can exploit for upcoming queries. Dialogue with a GIS must incorporate means to reference objects or subareas on maps.

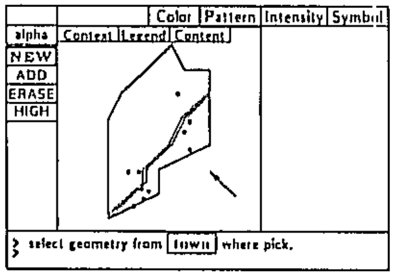
#### 3.2.1 Selection By Pointing

Users prefer to reference objects by pointing to them, rather than typing some form of identification. A town on a map, for example, is more conveniently referenced by pointing to it instead of using its name as identification. A typical GIS query with direct graphic input is "How many people live in this town?", and the user refers to the representation of a building on a map. The same query could have been asked with purely lexical means if the name of the town had been known to the user. Selection by pointing is a major object-oriented feature of graphical

The interaction must lead the user through the possibly complex process of selecting objects by pointing to them. The dialogue is initiated by some lexical input containing keywords for selection by pointing. The following interaction processes must be supported in the interface:

- The user is prompted to select the object(s). The prompting must provide the user with feedback about which object types to select, and activate the pointing device. The feedback is especially important for queries with mutiple *picks* on different object classes. Ideally, the query as typed is used in this communication, highlighting the object types of the current selection.
- The selected objects are immediately verified so that the users have control over their selections and what the system actually identified. The particular identification of the target is a separate process, outside of the interaction, and will not be discussed here.
- Users can alter their choice at any time before confirming the selection and cancel the previously selected objects. Such corrections must include the corresponding verification which resets the presentation of the cancelled objects to their original state.
- Confirmation is the final user action. Various techniques, such as 'double clicking', pressing a dedicated button on a multi-button mouse, or using the return key on the keyboard are commonly used.

The snapshot in figure 2 shows how a query references objects which are displayed on the current map. The lexically formulated query must be completed by pointing with the mouse to the corresponding town. The highlighted word *town* in the query reminds the user of the action to be taken.



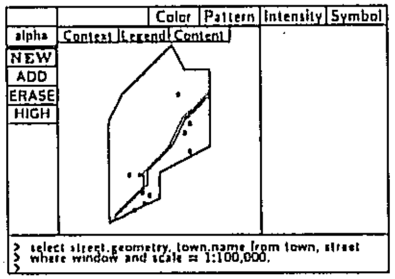
> select geometry from town where pick.
>

Figure 2: Selection by pointing: The user is prompted to select a town with the cursor to it.

During geographic analysis, a situation is often first viewed from a greater distance before the region or object of interest is investigated more closely. For such iterative approaches, a graphical scene serves as reference to the selection of the following rendering. This process must be supported from the dialogue using direct-manipulation methods to select the area of concern. A typical query with reference to a subarea is, "Show all streets and towns with this region" and the user selects the region from a rendering on the screen.

The selection of subareas has four interaction phases: (1) the prompt for the user's selection, (2) verification of the user's input, (3) possible corrections, and (4) confirmation of the selection by the user. An efficient method for selecting a part on a map is to choose a rectangle with its edges parallel to the screen axes. With a direct-manipulation device, the rectangle can be selected by pointing at two of its opposite corners.

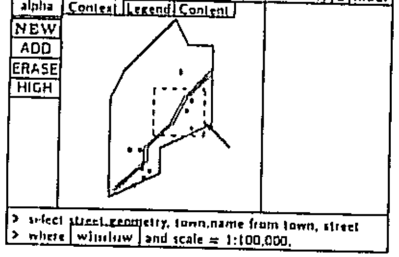
Zooming into a drawing is often linked to the users' desire of having maps in specific even scales, for instance, 1:10,000 is preferred over 1:9,972.32. Selection of scale and subarea can be linked together in the interface. Because the range of the viewport is determined by the scale, a subarea can be easily selected by dragging a scaled window over the desired location. This technique gives the users an accurate impression of the range they are going to see. The following three interface snapshots visualize the interaction. First, the user types in a query from the keyboard. The keyword *window* states that the subarea will be selected with a direct-manipulation device.



> select street.geometry, town.name from town, street
> where window and scale = 1:100,000.
>

Figure 3: Typing a query with the reference to select the area on the current drawing.

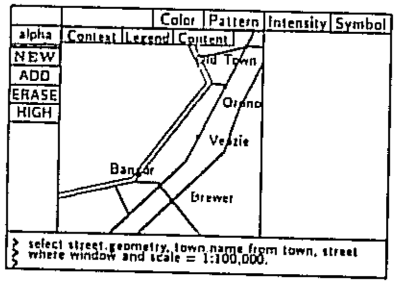
As feedback from the system, a window in the desired scale pops up on the screen. The highlighted keyword *window* in the query guides the user through the selection, indicating the task to be completed. The user moves the window over the desired area and confirms the selection by clicking on a mouse button.



```
> select street.geometry, town.name from town, street
> where window and scale = 1:100,000.
```

Figure 4: Dragging the scaled window over the area of interest.

The result is a more detailed map of the subarea chosen with the window.



```
> select street.geometry, town.name from town, street
> where window and scale = 1:100,000.
```

Figure 5: The larger-scaled map.

## 3.3 Representation

GIS technology must explore methods to create information systems with capabilities that go beyond the production of static maps. The knowledge collected in a GIS can be presented both lexically and graphically.

### 3.3.1 Lexical Representation

The interaction with a GIS can be partially covered by a language in a traditional style when users formulate a query which they type in from a keyboard, and expect the result to be presented as text. A typical example for a purely lexical query in a GIS environment is, "How far is it from

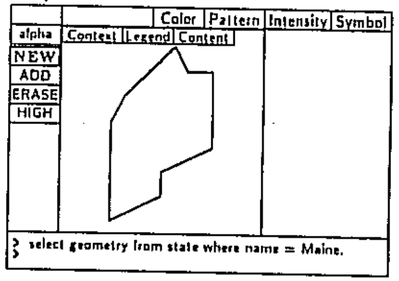
capabilities for treating spatial objects and properties [Egenhofer 1987], such as MAPQUERY [Frank 1982], enhanced with powerful 'expert system' tools, such as LOBSTER [Frank 1984b] or KBGIS [Smith 984], are expected to partially satisfy the interaction with future spatial information systems

### 3.3.2 Graphical Representation

Analysis of spatial data requires graphical representation, either in the form of quick sketches, sophisticated maps, or other non-conventional representations, such as 3-D terrain models.

Graphical representation adds constraints to the interface which are not covered by traditional techniques for lexical representation. The limitation of space on a screen, for example, made designers investigate how large amounts of data can be presented in an understandable way. Estimates derived from counting objects displayed on a screen show that useful screen drawings contain about 2000 to 5000 objects. Screens with substantially less data appear as empty and do not convey enough information about an area, whereas screens with more data are too crowded to be easily read.

The snapshot in figure 6 shows a query and its graphical result.



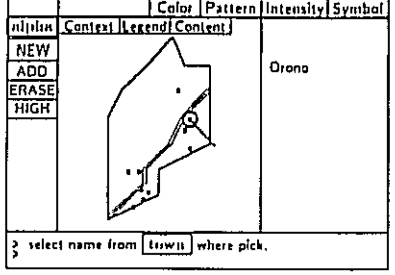
```
> select geometry from state where name = Maine.
>
```

Figure 6: A query with a graphical result.

### 3.3.3 Concurrent Graphical/Lexical Representation

Geographic analysis frequently requires the extraction of non-spatial information from a spatially rendered scene, or spatial information in lexical form. Such answers can be sufficiently analyzed only if both the reference and the answer are simultaneously visible and the user can compare them. The user interface must support such queries.

The interface in figure 7 shows a scene where a user asked for information about a certain town which was represented on a map. The highlighted town marks the selected object on the map, and the lexical information is presented in the lexical output window.



> select name from town where pick.

Figure 7: Inquiring lexical information about objects on a drawing.

The layout of the interface supports the user's analysis because graphical and lexical representations do not overlap. As a result, both representations are simultaneously visible.

## 4 Object-Oriented Spatial Query Languages

The ease of use of these methods will be boosted by embedding them into a sophisticated interface.

### 4.1 Directing the Graphical Representation

The ability to present spatial data graphically is the most obvious difference between spatial and conventional information systems. Interactive-graphic presentation is powerful for mapping systems because the content of maps can be quickly modified. While traditional mapping is a static product—once the map is printed it cannot be updated unless overwriting it with some pencil—mapping with spatial information systems promotes immediate manipulations on drawings. Objects can be added to, removed from, or modified on an existing map without the need to redo the drawing from scratch. This implies that the user must have tools to manipulate the composition of a map.

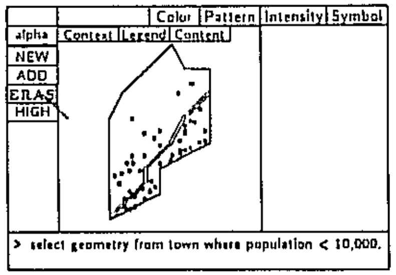
Unlike conventional systems which treat each result as an entirely new representation and do not refer to earlier results, several interactive-graphic drawings are often overlayed with each other, or one 'thematic layer' is removed from another. These are powerful processes in spatial information systems, combining spatial data of arbitrary sources to a composition that appears optimal for a user's application.

Another important interactive feature is that individual objects on a map can be highlighted to facilitate their identification in complex situations. Imagine a town map with streets and buildings on it, and the user wants to find the City Hall. Instead of checking each building, it is faster and more convenient to choose a presentation for the City Hall that makes it easy

to distinguish it from other buildings. The four graphical presentation types in an interactive-graphic information system are the following:

- *New* refreshes the viewport before drawing the next picture.
- *Overlay* adds the result of the current query to the existing picture.
- *Remove* erases the result of the current query from the existing picture.
- *Highlight* presents the result in a fashion that draws the users' attention to it.

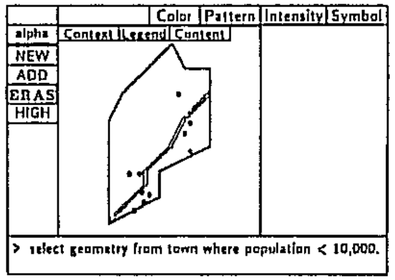
The following two snapshots show how a scene is dynamically modified. Prior to the first snapshot, the user had selected all towns in Maine. In order to correct the crowded scene, all 'small' towns are removed.



> select geometry from town where population < 10,000.

Figure 8: Removing information from a drawing.

As result, only those cities with 10,000 inhabitants or more remain on the map.

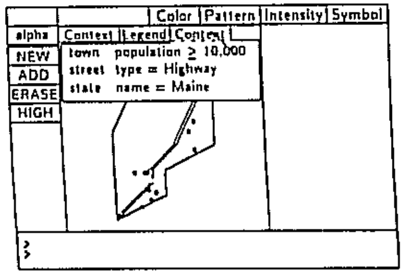


> select geometry from town where population < 10,000.

Figure 9: The result of a dynamic change of a map.

Note that the database query is exactly the same as one for making a map with the small towns, and only the selection of the representation mode (ERASE) made the result be removed.

The possibility of dynamic changes of drawings gives rise to the combination of several query results to a single rendering. This concept is different from the traditional production of a completely new result for each query. Instead, the representation the user sees may be used as the reference for the upcoming query, promoting a more dynamic working style, and control over the representation and its content may be lost. With traditional one-query one-representation techniques, the result is always associated with a single query. Composing the results of several queries makes it harder to keep track of what is actually presented. An essential feature in a query environment with multiple representation modes is a control mechanism which allows the user to check, after a sequence of queries, what a single query for the current drawing would have been. This knowledge can be of vital interest to the user, preventing incorrect or misleading assumptions about the presentation with possibly disastrous consequences. The interface snapshot in figure 10 shows the user checking the content of the map from figure 9. Since the towns with less than 10,000 population were erased, the map shows only towns with a population of 10,000 or more.



```
town    population ≥ 10,000
street  type = Highway
state   name = Maine
>
>
```

Figure 10: Verifying the content of a map.

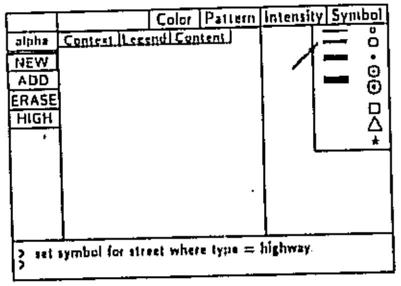
Graphical representations are only useful if the compilation of the corresponding queries is accessible from the interface. In analogy, a hard copy of a map should never be produced without the corresponding query set which produced it.

## 4.3 Varying Graphical Presentations

In GIS applications, often the same data is presented in various fashions according to different views. The view of a road, for example, is different for a surveyor, a transportation company, or a cartographer. One sees it as a parcel, the other as a classified transportation line, another as symbolic line, and for each purpose the road must be presented differently. Such varying presentations must be discriminated from generalization. While generalization establishes different abstraction levels for an object class [Brodie 1984], varying presentation relies upon the same

The choice of the presentation must be controlable from the interface to allow the users to adopt the most appropriate presentation type for their application. This flexibility is an important contribution to the multi-purpose applications of a single data collection, and the user interface must provide tools to accomodate the presentation to the individual, often subjective, needs.

Four object-oriented methods, based on the purely graphical properties of Bertin [Bertin 1983], are distinguished to vary the graphical representation of an object: (1) colors, (2) patterns, (3) intensity, and (4) symbols. With a command language, for example, the user can define the representation specifications. By supporting these definitions from pull-down menus, the user's choice is efficiently aided. The interface sketch in figure 11 shows the selection of a symbol for a highway with support from a pull-down menu.

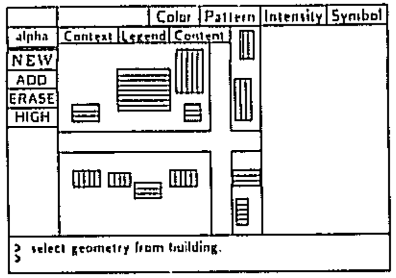


```
> set symbol for street where type = highway
>
```

Figure 11: Selecting a symbol from the menu.

## 4.4 Legend

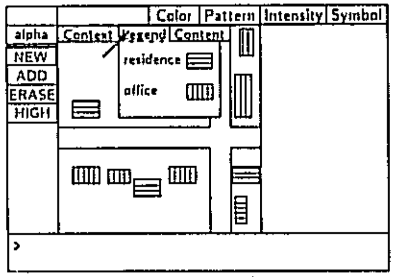
The interpretation of graphical representations becomes harder with a greater variety of symbols for different classes. Without a suitable explanation, symbols are sometimes difficult to interpret. For example, a map with two different symbols for buildings. one representing residences, another one offices, requires an interpretation.



```
> select geometry from building.
>
```

Figure 12: The result of a query without context.

Traditional cartography uses the legend as explanation of rendering. The legend is specific for a map; therefore, each graphical representation must have its own legend. The interaction with the legend asks for a flexible design. It must be accessible immediately from the user interface whenever information is required. Due to the limited screen area, it is not possible to have the whole legend permanently visible. The snapshot in figure 13 shows the user examining the legend to discriminate residences from office buldings.

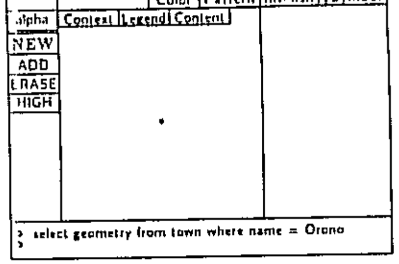


```
>
```

Figure 13: Controlling the legend to interpret a drawing.

## 4.5 Appropriate Graphical Context

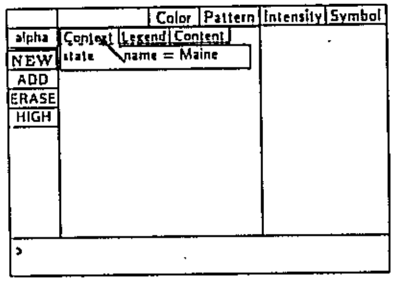
The interpretation of graphical representation is extremely sensitive to the context and environment in which it is shown. Unlike lexical representation, it is often not sufficient to draw only those objects that were asked for. Imagine a query "Show the town of Orono", where the result is only a point in the middle of the screen [Frank 1982] (figure 14).



```
> select geometry from town where name = Orono
>
```

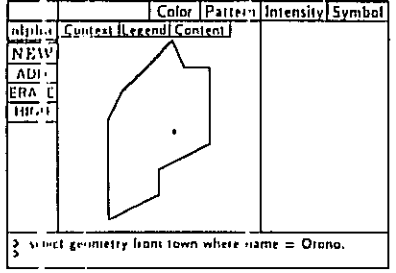
Figure 14: The result of a query without context.

A reasonable answer would have required a context in which the position of the point could have been spatially interpreted. For example, by showing the borders of the state of Maine, sufficient information is given to locate Orono. Sophisticated graphical representation must consider the selection of appropriate context which depends upon the purpose of the drawing, the scale, and the data density. The users must be provided with tools to select the most efficient context for their maps from the user interface. The following snapshots show how the graphical representation would have been understandable if a context was previously defined.



```
>
```

Figure 15: Setting a gaphical context.



? select geometry from town where name = Orono.

Figure 16. The result of a query without context.

## 5 Conclusion

The design of user interfaces plays a vital role for spatial information systems and the interaction differs significantly from conventional, purely lexical systems. The query languages for such systems must incorporate techniques to treat the specific properties of spatial objects. This paper identified the graphical, object-oriented representation of spatial data as a crucial issue demanding various display types, such as overlay, erase, and highlight. Interaction with graphically represented spatial objects requires methods, such as selection by pointing to objects and selection of subareas. The environment specific for representation of spatial data needs selection of context and tools to manipulate the graphical representation. These concepts were visualized in a series of interface snapshots.

## 6 Acknowledgement

Thanks to Cathy Page, Mark Palmer, and Jeff Jackson for their comments on the layout of the interface.

## References

[Aronson 1983] P. Aronson and S. Morehouse. The ARC/INFO Map Library: A Design for a Digital Geographic Database. In: Auto-Carto VI, 1983.

[Batory 1984] D.S. Batory and A.P. Buchmann. Molecular Objects, Abstract Data Types, and Data Models: A Framework. In: 10th VLDB conference, Singapore, 1984.

[Bertin 1983] J. Bertin. Semiology of Graphics. The University of Wisconsin Press, Madison (WI), 1983.

[Bly 1986] S.A. Bly and J.K. Rosenberg. A Comparison of Tiled and Overlapping Windows. In: Human Factors in Computing Systems, CHI '86, April 1986.

[Brodie 1984] M.L. Brodie. On the Development of Data Models. In: M.L. Brodie et al., editors, On Conceptual Modelling, Springer Verlag, New York (NY), 1984.

[Buchmann 1985] A.P. Buchmann and C.P. de Celis. An Architecture and Data Model for CAD Databases. In: 11th VLDB conference, Stockholm, 1985.

[Chamberlin 1976] D.D. Chamberlin et al. SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control. IBM Journal of Research and Development 20(6), 1976.

[Codd 1970] E.F. Codd. A Relational Model for Large Shared Data Banks. Communications of the ACM, 13(6), June 1970.

[Dittrich 1986] K. Dittrich. Object-Oriented Systems: The Notation and The Issues. In: International Workshop in Object-Oriented Database Systems, Pacific Grove (CA), 1986.

[Egenhofer 1987] M. Egenhofer. An Extended SQL Syntax To Treat Spatial Objects. In: Y.C. Lee, editor, Proceedings of the Second International Seminar on Trends and Concerns of Spatial Sciences, Fredericton (New Brunswick), 1987.

[Frank 1982] A. Frank. MAPQUERY—Database Query Language for Retrieval of Geometric Data and its Graphical Representation. In: D. Bergeron, editor, SIGGRAPH '82, ACM Computer Graphics, Boston (MA), July 1982.

[Frank 1983] A. Frank. Problems of Realizing LIS: Storage Methods for Space Related Data: The Field Tree. Technical Report 71, Swiss Federal Institute of Technology, Zurich (Switzerland), 1983.

[Frank 1984a] A. Frank. Requirements for Database Systems Suitable to Manage Large Spatial Databases. In: International Symposium on Spatial Data Handling, Zurich (Switzerland), August 1984.

[Frank 1984b] A. Frank. Extending a Database with Prolog. In: L. Kerschberg, editor, Proceedings of the First International Workshop on Expert Database Systems, Kiawah Island (SC), October 1984.

[Green 1985] M. Green. Report on Dialogue Specification Tools. In: G.E. Pfaff, editor, Computer Graphics Forum, Springer Verlag, 1985.

[Guttman 1984] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proceedings of the Annual Meeting ACM SIGMOD, Boston (MA), 1984.

[Härder 1985] T. Härder and A. Reuter. Architecture of Database Systems for Non-Standard Applications (in German),. In: A. Blaser and P. Pistor, editors, Database Systems in Office, Engineering, and Scientific Environment, Springer Verlag, New York (NJ), 1985.

[Newman 1983] W.M. Newman and T. Mott. Officetalk-Zero: An Experimental Integrated Office System. In: P. Degano and E. Sandewall, editors, Integrated Interactive Computing Systems, North-Holland Publishing Company, Stresa (Italy), 1982.

[Nievergelt 1984] J Nievergelt et al. The GRID FILE: An Adaptable, Symmetric Multi-Key File Structure. ACM Transactions on Database Systems, 9(1), 1984.

[Shneiderman 1983] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. Computer, 16(8), 1983.

[Sidle 1980] T.W. Sidle. Weakness of Commercial Database Management Systems in Engineering Applications. In: 17th Design Automation Conference, 1980.

[Smith 1983] D.C. Smith et al. Designing the Star User Interface. In: P. Degano and E. Sandewall, editors, Integrated Interactive Computing Systems, North-Holland Publishing Company, 1983.

[Smith 1984] T. Smith and M. Pazner. Knowledge-Based Control of Search and Learning in a Large-Scale GIS. In: International Symposium on Spatial Data Handling, Zurich (Switzerland), August 1984.

[Stonebraker 1976] M. Stonebraker et al. The Design and Implementation of INGRES. ACM Transactions on Database Systems, 1(3), September 1976.

[Sutton 1977] J.A. Sutton and R.H. Sprague. A Study of Display Generation and Management in Interactive Business Applications. Technical Report RJ2392, IBM Research Division, San Jose (CA), November 1977.

[Tamminen 1982] M. Tamminen. Efficient Spatial Access to a Data Base. In: ACM-SIGMOD, Orlando (FL), 1982.

[Zloof 1977] M.M. Zloof. Query-by-Example: A Database Language. IBM Systems Journal 16(4), 1977.

# MODELING ERROR IN RASTER-BASED SPATIAL DATA

Dr. Michael F. Goodchild
Visiting Professor
Department of Geography
University of California
Santa Barbara, California 93106
U. S. A.

Wang Min-hua
University of Western Ontario

## Introduction

The problem of error in GIS products has attracted considerable attention in recent years (see for example Chrisman, 1987; Goodchild and Dubuc, 1987; Burrough, 1986; Walsh, Lightfoot and Butler, 1987). Spatial data handling systems process data with high precision, so it is perhaps surprising to find that the results of query or analysis frequently conflict with ground truth. Unfortunately the input to such systems is almost always an abstraction of reality, and processing frequently increases the degree of abstraction. For example the data input from a forest inventory often consist of classified stands of timber, defined by polygonal boundaries and homogeneous attributes. Since the forest stand itself is rarely if ever homogeneous, even on the simplest attributes, such input constitutes an abstraction of reality; furthermore the degree of abstraction is usually unknown. To compute an estimate of marketable timber, the attributes and area of the stand will be input to a set of yield tables, but the results will be subject to additional uncertainties because yields are only crudely predictable.

Research on error in GIS has two objectives, both of substantial practical significance: first, to minimize error in products, and second, to develop models of error which can be used to compute measures of uncertainty. At present we know of no system which routinely estimates and reports measures of the error inherent in any of its products.

Some GIS operations contribute directly to error, and such cases are relatively easy to characterize and model. Errors introduced during digitizing and processing are termed processing errors. Consider a homogeneous patch of land, represented as a polygon in a vector database. Conversion to raster representation introduces a distortion, since the patch must now be represented by a set of pixels of fixed size. If the area of the patch is estimated from the raster representation, it is relatively straightforward to model the effects of this distortion on the estimate of area, in the form of a measure of standard error (Goodchild, 1980). But while such measures can be used as useful guides in selecting pixel sizes, they are based on the assumption that the homogeneous patch with precise boundary is a correct