

A Card-based Visual Query System for Geographical Information Systems

Shiguang Ju¹, Weigang Guo¹ and Héctor J. Hernández²

¹Computer Science Department, Jiangsu University, P.R. China (Jushig@ujs.edu.cn)

²Computer Science Department, Texas Tech University, Lubbock, TX, USA 79409
(hector@cs.ttu.edu)

Abstract. This paper introduces a visual query system for geographical information systems, which is easy to use for non-professional persons. The system creates a new iconic metaphor — a card, which has a very strong expressive power and can be used to represent complex spatial objects and spatial relationships between objects. The end users can describe their query requirements in a visual programming environment by selecting appropriate cards and putting them into the proper query boxes. The result of the query is also displayed in graphical form, and this allows the user to understand the information easily.

1 Introduction

Following the progress of the society and the development of technology, there is much more demand for geographical information. But, most of the geographical information systems (GISs) available now are geared toward professionals. They do not have a friendly query interface, and only well-trained persons can get information from them. Ordinary people cannot benefit from GISs directly. So, it is valuable to construct an easy-to-use visual query system to retrieve geographical information.

The research on visual querying of spatial databases mainly focuses on two aspects. One is how to visually express the objects to be retrieved; the other one is how to express the spatial relationships between these objects. There are three main methods. The first method uses predefined icons which represent objects and spatial operations to provide a visual query interface [1,6]. The shortcoming of this method is that the predefined icons do not have strong expressive power, and the consequent query capability is limited. The second method is “Query-by-Sketch” [2, 3], in which the method of specifying spatial relationships is by drawing. The third is a hybrid method [4] that uses text and sketching. Using this method, users can draw spatial configurations of the objects they would like to retrieve from the GIS, while the textual part permits the specification of the geographical semantics. The shortcoming of the second and third methods is that they need the user to repeatedly interact with the system until the system understands what the user means. So, it is also not convenient for the non-professional users.

This paper introduces a card-based visual query system for GISs. The system is based on the CQL language [5] which was designed and developed at Jiangsu

University. A new kind of visual element – a card, which has stronger expressive power than other ordinary icons, is used to express spatial objects and spatial relationships. A visual query sentence editor also has been developed, which let end users express their query requirements easily; the query result is also displayed visually. With this query system the end users can get their information directly.

2 Visual elements: Cards

2.1 Definition of Card

In CQL, a card is defined by a four-tuple $[X_m, X_i, X_t, X_p]$, where X_m is the name of the card, X_i is the image of the card, X_t is the type of the card, and X_p is the parameter of the card. Here, X_p enhances the expressive power of the card. It can accurately describe the spatial objects and spatial operations. A card with different X_p will have different meaning and represent a different object or attribute.

In our system, there are three types of cards: *Real object* cards: these cards exactly represent the spatial data of the database, i.e., real objects of the earth's surface; for example, highways, rivers, roads, oil wells, houses, etc. *Conceptual object* cards: these cards represent the database files which store the real objects and the relationships between the objects; for example, a country, a region, a district, a project's area, etc. *Operation* cards: these cards represent the actions or computational processes of the system.

2.2 Designing the cards' appearance

The three types of cards have the same structural appearance, as shown in figure 1. Visually, on the face of a primitive card, slot X_m is to show the name of the card, and slot X_i is to place the image, which visually represents one class of entities. Another slot, for the parameter X_p , is used to accept user's input. X_t , as the internal type of the card, is stored in the system's card library and is not displayed on the card.

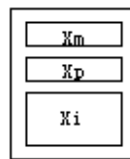



Fig. 1. A primitive card

2.2.1 Design of the real object cards

Every country has its own survey standard that defines the shape of the surface objects. Thus, the appearance of the real object cards is classified and designed according to the survey standard. This will make the ordinary civil engineers understand the meaning of a card at a glance.

2.2.2 Design of the conceptual object cards

Conceptual objects mostly do not have an inherent visual appearance. For example, the object “an opened file” does not have an exact shape. Hence, its logical meaning is abstracted and its visual representation is designed in a form that most people can understand. Here, the image “” is used to represent an opened file. Generally, in the system, the pictures are designed according to the contents of the files. For example, a national map of China is used to represent the conceptual object “region,” and some engineering maps to represent the object “project.” Among the conceptual objects, there is a special object: the temporary object. It is used to express a sub-query result of a complex query, and a subsequent query can retrieve further information from the temporary object.

2.2.3 Design of the operation cards

The operation cards represent the query actions of the system. There are four types of operation cards; in turn each type consists of several operation cards. *System* operations: OPEN, CLOSE, BROWSE, UNION, INTERSECT, MINUS; *Orientation* operations: DIRECT, AZIMUTH, ABOVE_OF, BELOW_OF; *Metric* operations: NEAR_OF, ALONG_OF, DISTANCE_OF, DISTANCE, LENGTH, AREA; *Topology* operations: INSIDE_OF, OUT_OF, CONNECT_OF.

In order to describe the spatial relationships more clearly, there are five operation cards (UNION, AZIMUTH, NEAR_OF, ALONG_OF, DISTANCE_OF), which can be the value of the parameter X_p , while others do not have any parameter related to them.

The picture for the slot X_m is designed according to the inherent meaning of the operation. For example, for the OPEN operation, a picture of “a person opening a book” is adopted. The operation INSIDE_OF represents an object inside another one; then the picture of “a person entering an area” is used. This enables the end user to understand the cards’ meaning rapidly.

Figure 2 demonstrates the appearance of some cards of the system.

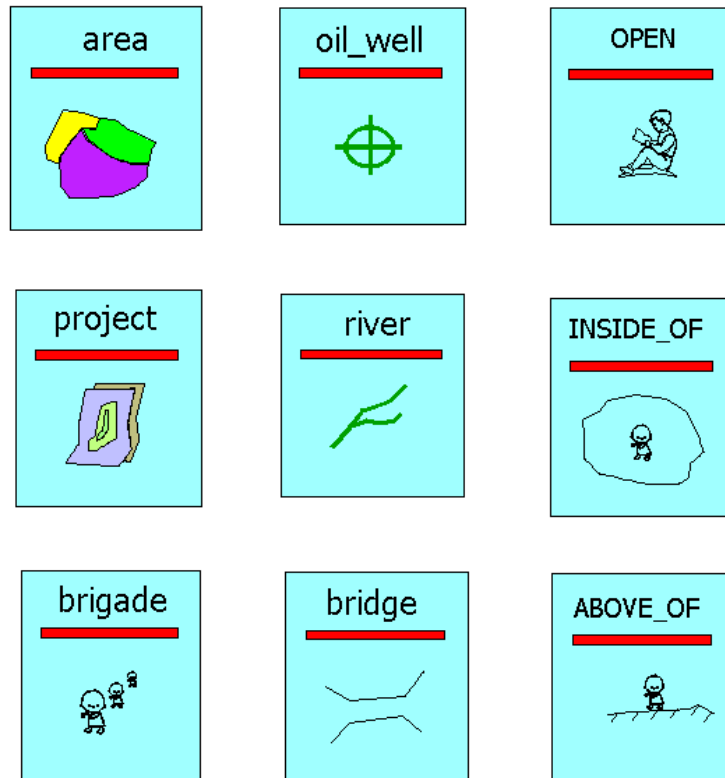


Fig. 2. Examples of conceptual object cards (left column), real object cards, and operation cards (right column).

3 Supporting environment of the visual query system

3.1 Architecture of the system

Figure 3 shows the architecture of the system. The visual query sentence (VQS) editor allows users to construct their VQSs, and the VQS interpreter translates the VQSs to the system's internal code. That is, under the light of the system's storage model, the series of images representing query sentences are converted to the system's internal codes.

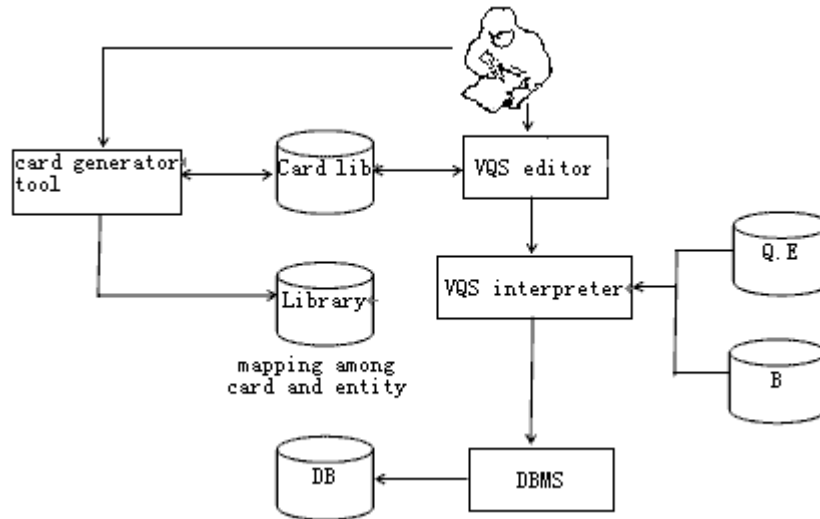


Fig. 3. Architecture of the system

3.2 The relationship of the spatial database

The system is based on a spatial database management system (DBMS). The original spatial data of the database is based on vector data format. Its data structure is object-relational, and the DBMS adopts a two-level storage mechanism to store data.

Here, the data of a Mexican company's oil wells management are given as an example, and four relations are used to describe the relational data. The details are shown in tables 1-4.

Table 1. Relation PROJECT

Proj_name	Area_name	Brigade_key	Description
DETALLE_DR	Camargo	nes_9	reality

Table 2. Relation OIL_WELL

W_name	X	Y	Proj_name	Brigade_key	Date	Profundity
Cerro	13000	21000	DETALLE_DR	Nes_9	19850301	4567
Dr_coss	83700	65250	DETALLE_DR	Nes_9	19851101	2611

Table 3. Relation BRIGADE

Brigade_key	Company	leader	Location	phone	Energy
nes_9	PEMEX	J. Pérez	villahermosa	65342	minas

Table 4. Relation MAPS

Map_name	Proj_name	Description
----------	-----------	-------------

H55_28	DETALLE_DR	take out company DETENAL
H55_29	DETALLE_DR	take out company DETENAL

In table 4, each Map_name corresponds to a real map, and the map is stored through a two-level storage mechanism. Each map is composed of two files whose name extensions are DEF and GRA.

3.3 The visual programming environment

The user interface of the system, as shown in figure 4, consists of four main parts: VQS editor, card menus, graphical output area of the retrieval result, and the textual output area of the retrieval result. The LOCAL BUTTON is effective only within the local area while the GLOBAL BUTTON is effective in the whole interface.

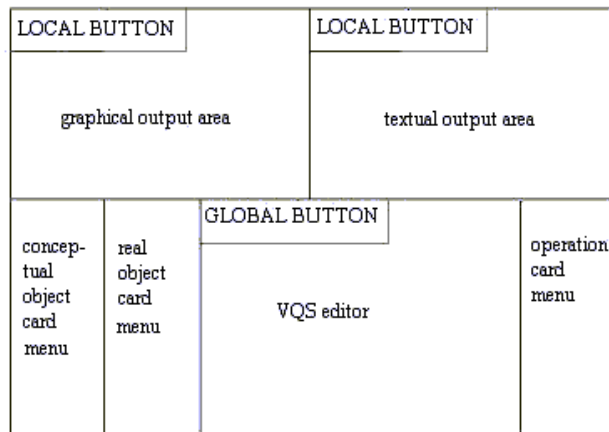


Fig. 4. The outline of the system's user interface

The VQS editor consists of three programming boxes (see figure 5). BOX1 is used to place the object cards, which represent the entities the user wants to retrieve. BOX2 is used to place the object cards from which the user retrieves, and BOX3 is used to place operation cards, which represents "how to query" or the relationships between BOX1 and BOX2.

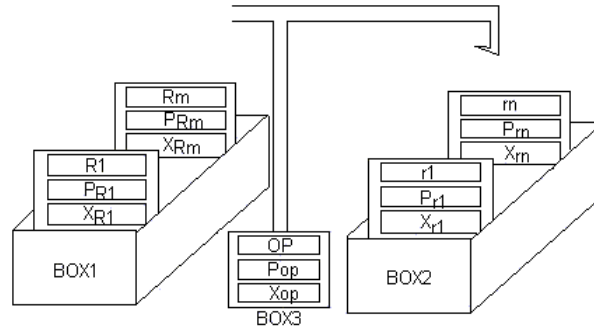


Fig. 5. The structure of the VQS editor

In figure 5, the cards R_1, \dots, R_m represent m entities to be retrieved from the database; r_1, \dots, r_n represent n entities from which the query will be executed; OP represents one operation; $Xr_1, \dots, Xr_n, Xop \square XR_1, \dots, XR_m$ are the images of the cards; $Pr_1, \dots, Pr_n, PR_1, \dots, PR_m, Pop$ represent the parameters of the cards. The meaning of the query in figure 5 is to retrieve the entities R_1, \dots, R_m from the entities r_1, \dots, r_n by using the operation defined by the card OP .

When an end-user wants to make a query, there are two main steps. The first is to divide the query into three parts: what to query, where to query, and how to query. The second is to program with the VQS editor. The method or process of programming is easy to understand. The user just selects the proper cards and puts them into the proper BOXes; the user can also define some parameters for the cards where necessary. After the user finishes programming, the system will understand what to do.

Three-dimensional shadow technology is adopted to design and implement a VQS editor, as shown in figure 7. On the front of BOX1, there is a picture that expresses the meaning “what to select” in the hieroglyphics used by the Aztecs. On the face of BOX2, there is another picture that expresses the meaning “from where to select” in the same symbolic language. The third part simulates an elevator at work to select cards from BOX2. Its picture means “how to select.” On the face of the elevator one or more active process cards can be attached. They come from the library of process cards.

4 Visual process of the query

Queries begin by clicking the button “Clear.” The end users only use the mouse to select the cards that represent the entities that will be processed. In the following, we give some examples to illustrate the visual query process.

Example 1: One end user wants to know which oil wells are inside the “DETALLE_DR” project. From the beginning, he/she can look for the card “OIL_WELL” in the real objects column in the card library. After having found it, he/she then clicks on the card and a dialogue box will be shown to ask for parameters

of this card, as shown in figure 6. If he/she wants to query all information about the oil wells, he/she does not need to enter any parameters in the slot. After he/she has selected parameters and indicated into which box the card will be inserted (here, click the check box of BOX1), the restricted card of oil well will be transformed into an active card. As he/she clicks the “ok” button, the active card “OIL_WELL”

Parameters

Which BOX insert in?

Box1

Box2

OK

Fig. 6. A dialogue box for parameters input

will be put in BOX1. The next step is to define from where to select the oil wells. The user looks for the object card “PROJECT” in the card library, and clicks it, then inputs “W_name= DETALLE_DR”, and clicks check box “BOX2” in the dialogue box; the active card will be put into BOX2. The third step is to define “how to select.” The user needs to find the appropriate operation card in the right column of the card library. Here he/she must invoke the appropriate “INSIDE_OF” card. If the card needs some parameters for the operation, the system will pop up a dialogue box in the same way. After clicking the button “ok” in the dialogue box, an active card “INSIDE_OF” will be attached to the elevator (BOX3).

Now, the user has constructed a visual query, shown in figure 7, in the screen with the VQS editor. This visual query is a simple program, and it is representing the query: Retrieve all the oil wells that are located in the project “DETALLE_DR.”

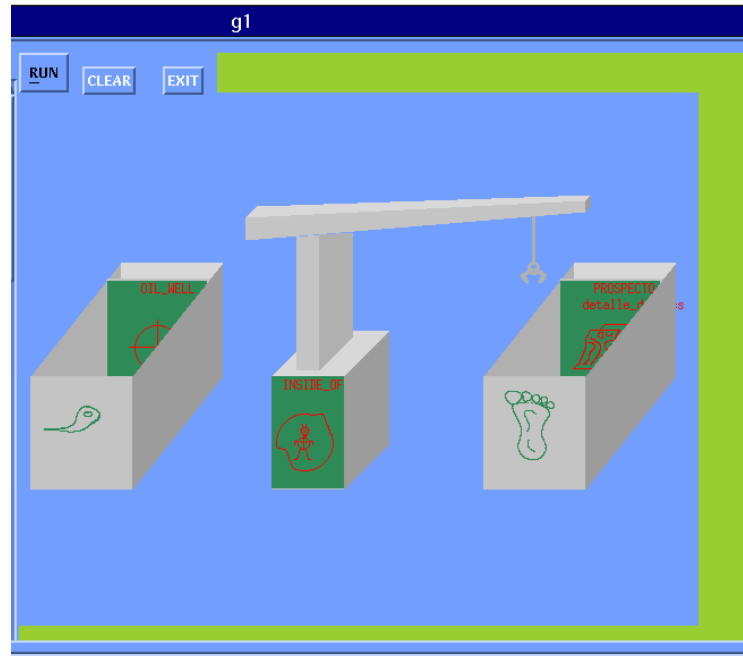


Fig. 7. Retrieve the oil wells that are located in the project “DETALLE_DR”.

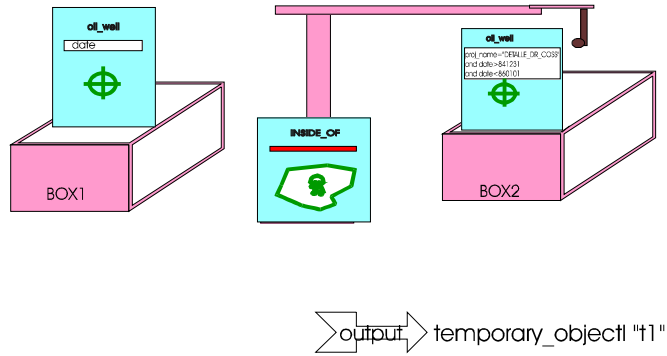
Example 2: Retrieve the names of all oil wells that were built after the built-date of the oil wells of the project “DETALLE_DR,” which were built in 1985. Its visual representation is shown in figure 8.

5 Visualization of the query result

Visualization of the query result is also very important to the end user. There are four output forms for the user to select: 1) **GRAPHICS**: the result of the query will be graphically displayed in the graphics output area; 2) **TEXT**: the result will be presented textually; 3) **TEMPORARY OBJECT**: the result of the query will be saved as a temporary file, and it will be used to further query; and 4) **ALL**: the result of the query will be output in the three ways described above.

In the graphical output mode, the system should convert the textual result to graphics. Thence, the system has a graphics generator, which can generate topographic maps in real time from the query result of the spatial database.

VQS1:



VQS2:

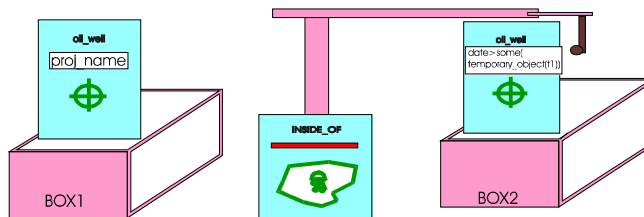


Fig.8. Visual query of example 2.

Surface objects are the basic elements of a topographic map. Usually, every country has its survey standard that defines the graphical symbols of the surface objects. The graphical output of the system is consistent with this standard. As shown in figure 9, if the user retrieves one (or some) certain type(s) of surface objects, then she/he can get the thematic map of these surface objects.

The general functions of the graphics generator are as follows: (1) It has built a library of the surface objects. Given the necessary data, it can draw the surface objects automatically; (2) it processes the transformation of real world coordinates to the screen coordinates -- different surface objects have their own transformation scale; (3) it solves the intersection problem between the surface objects while they are displayed on the screen. The principle of hidden line removal is the priority of the surface objects.

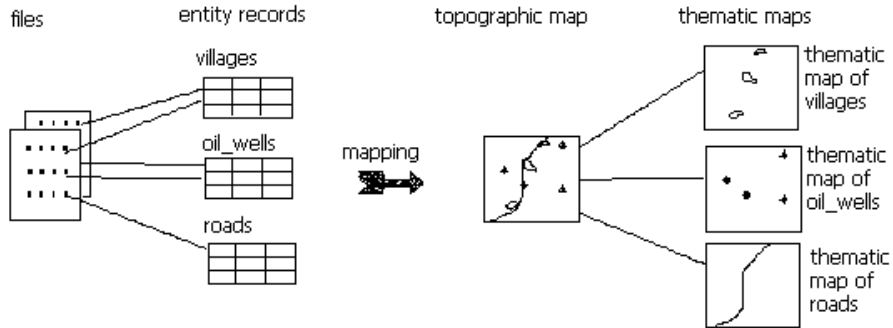


Fig. 9. Production of thematic map from the same type of entities in the database

Figure 10 shows an example of the visual output of the query result. It has two parts: graphical output and textual output.

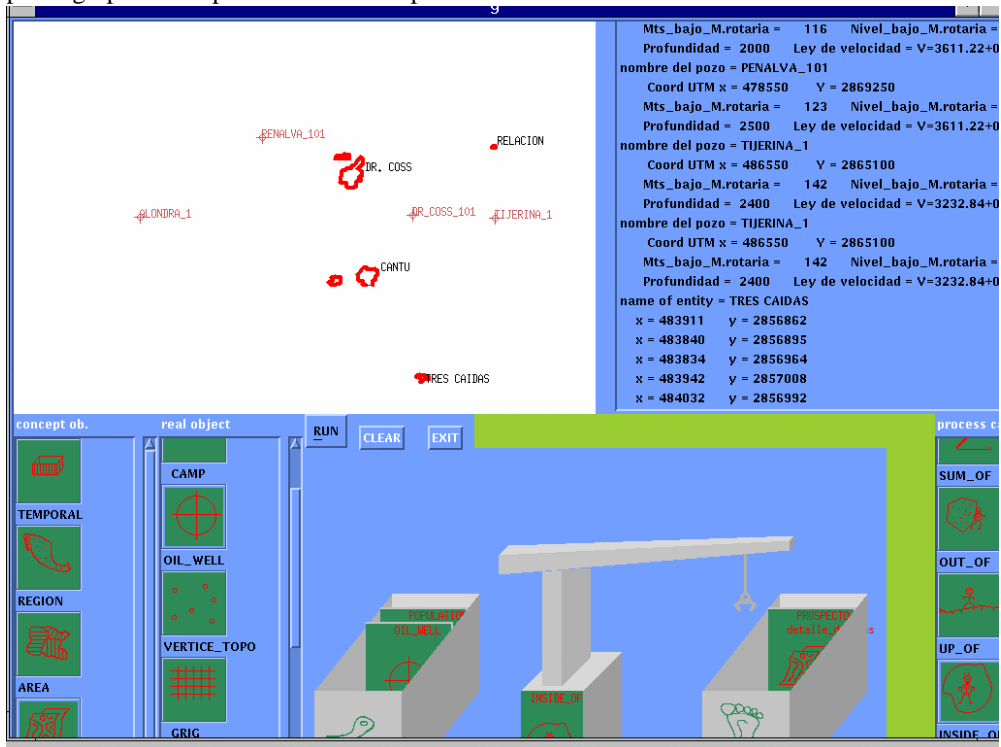


Fig. 10. Graphical and textual output of the retrieval result

6 Conclusions

The system is based on a spatial database management system developed by ourselves, and it is developed on a UNIX platform with X-Windows and Motif. The

programming language is C. The system makes it possible for the civil engineers to visually retrieve spatial information in the whole query process. Further research includes the adoption of the Query-by-Example philosophy to improve the interactive ability while setting the cards' parameters and perceptual research is also a topic for further work. Hopefully, the system can be applied in several fields.

References

- [1] Jungert, Erland, "Towards a visual query language for an object oriented geographical information system," *Proceedings of the 1990 IEEE Workshop on Visual Languages*, Oct. 4-6, 1990, pp 132-137.
- [2] Egenhofer, Max J, "Spatial-Query-by-Sketch," *Proceedings of IEEE Symposium on Visual Languages*, Sept. 3-6, 1996, pp 60-67.
- [3] Haarslev, Volker and Wessel Michael, "Querying GIS with animated spatial sketches," *Proceedings of IEEE Symposium on Visual Languages*, Sept. 23-26, 1997, pp 197-204.
- [4] <http://cosanostra.msh.unicaen.fr/~maurice/VQL>.
- [5] Shi-guang Ju, "Visual query language CQL for spatial database," *Chinese Journal of Computers*, Vol. 22, No.2, Feb.1999, pp.205-211.
- [6] Havard Tveite., "Structured modelling for geographical information systems using icons," *Proceedings of ScanGIS`2001*, pp. 185-201