Cigales*: A Visual Query Language for Geographical Information System: The User Interface

D. CALCINELLI, M. MAINGUENAUD

FRANCE TELECOM - Institut National des Télécommunications
9 Rue Charles FOURIER
F91011 EVRY - FRANCE
+ 33 1 60 76 47 82
+ 33 1 60 76 47 80 (fax)
Email: calci@int-evry.fr maing@int-evry.fr

Abstract:

In this paper, we focus on the design and the implementation of a query language, Cigales, for Geographical Information Systems. This language is based on a visual and declarative Query-By-Example philosophy. The semantics of an end-user query is expressed by a drawing.
We present the querying philosophy of the Cigales language. To take into account the particularities of the visual query language and the geographical data, we introduce several extensions to the classical database models (i.e., the icons, the variables, the overlap of the spatial representation).
To illustrate the concepts of the Cigales language, we provide the representation of several queries (involving a unique "object," two "objects" without operators, two "objects" linked with a spatial operator, several "objects" linked with several spatial operators).
Since the design of a user interface is an iterative process, the implementation is still going on and we present the current state of the CIGALES prototype.

Keywords:

Geographical Information System, Visual Query Language, User Interface

* Cigales stands for: Cartographical Interface Generating an Adapted Language for Extensible Systems

I. INTRODUCTION

In the current research toward the design of more powerful tools for urban planning, remote sensing, ... different research groups are simultaneously concentrating their work on Geographical Information Systems (GIS). GIS needs are very well known [19]. Nevertheless, several problems are still open. In this paper, we focus on the design of a user-friendly query language. Designing the user interface is one of the main issues to deal with when building spatial information systems. It will be increasingly true since the geographic information is now affecting such general applications as tourism and non-specialist users. A user should have a query language with a high level of abstraction. The visual (or graphical) query language concept appeared with the development of "cheap" graphical devices. Propositions of such languages are detailed for example in [5,10,11,14,18]. A query language is said to be visual whenever the semantics of the query is expressed by a drawing. It is said to be declarative whenever the query specifies the properties to be verified by the result, but not the way of obtaining them (i.e., imperative).

This paper is based on the CIGALES experiment. Cigales [4,13] is a visual and declarative query language for GIS. A query is defined with a graphical Query-By-Example philosophy. Since a query is represented by a drawing, the management of the query definition process is more complex. We present the querying philosophy of the Cigales language. To take into account the particularities of the visual query language and the geographical data, we introduce several extensions to the classical database models (i.e., the icons, the variables, the overlap of the spatial representation). To illustrate the concepts of Cigales, we provide the representation of several queries (involving a unique "object," two "objects" without operators, two "objects" linked with a spatial operator, several "objects" linked with several spatial operators). Since the design of a user interface is an iterative process, the implementation is still going on and we present the current state of the CIGALES prototype.

Part II presents the querying philosophy of Cigales, Part III presents a set of examples, Part IV presents the implementation and Part V presents the conclusion and future work.

II. THE QUERYING PHILOSOPHY OF THE CIGALES LANGUAGE

A visual Query-By-Example philosophy implies a graphical representation of the query. Since the semantics is expressed by a drawing, several ambiguities may appear [4]. In this part, we recall the functions of the graphical query editor, we present the data model handling and we present the basic operators.

II.1. The graphical query editor

The figure 1 presents the graphical query editor of Cigales. This editor supplies two kinds of structures, called the drawing area and the button. A drawing area is used to visualize the spatial relationships involving the basic objects of a graphical query. A button is a rectangular box linked to three concepts: (1) the semantics of the button, represented by either a text label or a graphic label inside the box, (2) the management of the button, which is either a single mode (i.e., a click in the box does not change the semantics of the button) or a switch mode (i.e., a click in the box changes the label and the semantics of the button), (3) the actions induced by the activation of the button. In this part, we describe these structures, and we explain how to build a graphical query.

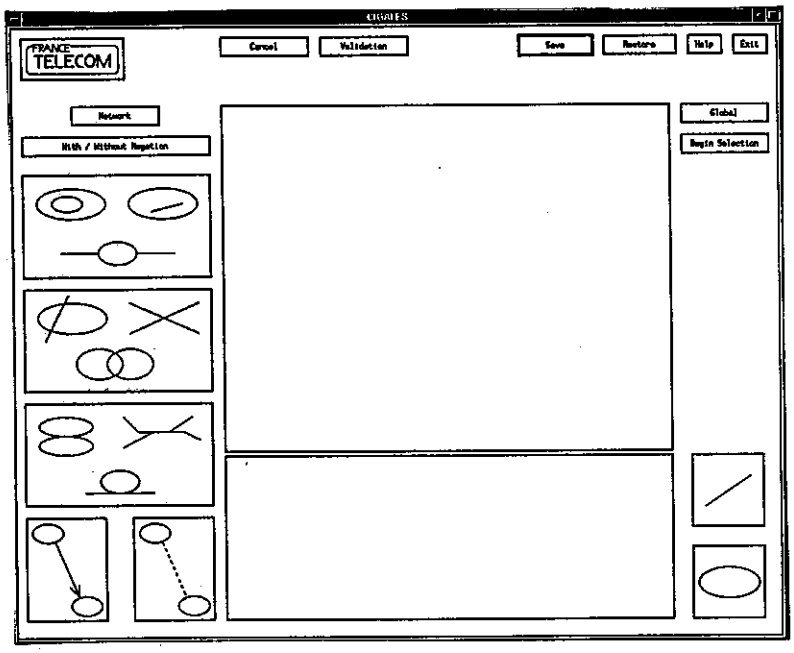

Figure 1

II.1.1. The elements of the editor

The editor contains two drawing areas named the query-area and the working-area. The query-area is the large empty rectangle at the middle of the editor. The working-area is the smallest empty rectangle below the query-area.

The query-area is used to display step by step the construction of the final query. The working-area is used to display step by step the construction of a sub-query. At each validation of a sub-query, the graphical expression inside the working-area is added into the query-area.

The Zone button and the Line button are located at the right-side bottom of the editor. The semantics of these buttons is represented by a graphic label. This graphic label corresponds to a metaphor completely independent of the geometrical storage database model (i.e., the spatial representation with points, lines and polygons). The database model offers different semantics for a zone (i.e., towns, forests, lakes) and for a line (i.e., roads, railways, rivers) depending on the application. The management of the buttons is based on the single mode. The associated actions are (1) to provide an automatic access to the database model, to define the semantics associated to the selected zone or the selected line, (2) to display this graphic label, and the associated semantics, into the working-area.
In the following a zone or a line is called an object.

The five buttons located at the left-side of the editor represent the five spatial operators. The semantics of these five operators is: the inclusion, the intersection, the bordering, the path and the Euclidian distance. The graphic label illustrates the several possible configurations. They are defined with the metaphors of the objects depending on the semantics of the operator. The management of each button is based on the single mode. The associated action is to display the graphic representation of the spatial relationship between the two objects currently in the working-area.

The With/Without Negation button is located over the five buttons of the operators. The semantics of this button is to express or to cancel the concept of negation upon the spatial operators. This semantics is given through the text label "With/Without Negation." The management of the button is based on the single mode. The associated action is to switch the semantics of the spatial operators. If the current semantics does not express the negation (the default configuration option), the activation of the button will be materialized by a cross in the graphic labels of the operators depending on their semantics. If the current semantics expresses the negation, the activation of the button will be materialized by the suppression of the cross in the graphic labels of the operators. The Euclidian distance is the unique spatial operator that is not involved in these modifications, since its negation is equivalent to the intersection operator.

The Network button is located over the With/Without Negation button. The semantics of this button is to express either a geometric-oriented aspect or a network-oriented aspect of the spatial operators, when an ambiguity may arise [4]. The management of the button is based on the switch mode. The "Geometric" (resp. "Network") text label expresses the geometric-oriented aspect (resp. network-oriented aspect) is on. The action of this button is to switch the semantics of the spatial operators, with no consequence on the visualization of the editor, since the user knows the current semantics by the current text label of the button. This button appears on the screen editor only in case of ambiguity [4].

The Begin Selection button is located over the Line button. The semantics of this button is to allow the selection of sub-objects resulting on the application of an operator into the query-area. Since several sub-objects of the same object may be considered as a single component of a sub-query [4], a mechanism must provide the beginning and the end of the selection in the query-area. The management of the button is based on the switch mode. The associated action is to mark either the beginning (with the "Begin Selection" text label) or the end (with the "End Selection" text label) of the selection in the query-area.

The Global button is located over the Begin Selection button. The semantics of this button is to allow the selection of either a full object (Global) or a sub-part (Partial) of an object (i.e., the result of a spatial operator) into the query-area. The selection of a full object is the usual way, but it is inadequate to express some queries. As an example, let (a) be a query:

(a) "Which roads cross a town in its non-forest part?"

Let us suppose the graphical expression of an intersection between a town and a forest is currently available into the query-area. The user needs to select (1) the line button to represent the road, (2) the part of the town that is not a forest, and (3) the intersection operator.

Since the user may select either a full object or a sub-part of an object in the query-area, the management of the Global button is made by the switch mode. The associated action is to provide the selection of either a full object (with the "Global" text label) or a part of an object (with the "Partial" text label).

The "France Telecom" Logo button is located at the left-side top of the editor. This button provides some internal system facilities on the configuration of the editor (i.e., the visualizations of the debug structures). It is only available in the debug configuration. The management of the button is based on the single mode. The associated actions provide the accesses to internal structures.

The Cancel button is located at the right-side of the Logo button. The semantics of this button is to cancel the current work. The management of the button is based on the single mode. The actions depend on the current state of the query definition process. Whenever the working-area is not empty, the associated action is to clear this working-area. Whenever the working-area is empty, the associated action is to clear the query-area.

The Validation button is located at the right-side of the Cancel button. The semantics of this button is to validate the current work. The management of the button is based on the single mode. The actions depend on the current state of the query definition process. Whenever the working-area is not empty, the associated actions are (1) to add the graphic representation of the spatial relationship currently in the working-area into the query-area, and (2) to clear the working-area. Whenever the

working-area is empty, and the query-area is not empty, the action of this button is to translate the graphical query into a formal internal query language.

The Save button is located at the right-side of the Validation button. The semantics of this button is to save an intermediate query. The management of the button is based on the single mode. The associated action is to save the graphical representation of the query available into the query-area with its corresponding formal internal expression.

The Restore button is located at the right-side of the Save button. The semantics of this button is to restore an intermediate query. The management of the button is based on the single mode. The associated action is to restore a graphical representation of a query into the query-area with its corresponding formal internal expression.

The Help button is located at the right-side of the Save button. The semantics of this button is to provide on-line help facilities. The management of the button is based on the single mode. The associated action is to display a help message on the screen, depending on the current state of the query definition process.

The Exit button is located at the right-side of the Help button. The semantics of this button is to leave the editor. The management of the button is based on the single mode. The associated action is to close a working session.

II.1.2. The construction of a graphical query

To explain the management of the Cigales editor, we present in this part the step by step construction of a complex query, involving the main buttons previously presented. Let (b) be the query:

(b) "Which routes go from Paris to Nice with the qualification that they border a lake for a while and then cross a town of more than 100,000 inhabitants in its non-forest part?"

To build the first step of the query (i.e., the sub-query representing the "routes from Paris to Nice"), the user first clicks on the Zone button. This choice induces (1) the access to the database model with which the user qualifies the zone as a town named "Paris," and (2) the drawing of the zone and the displaying of its qualification into the working-area. As the second step of the construction of this sub-query, the user clicks again on the Zone button, qualifies this zone as a town named "Nice," and the qualified zone appears into the working-area. The user now clicks on the path operator button. This choice induces (1) the access to the database model with which the user may define some constraints (as for example the price is less than 1000 units), and (2) the drawing of the

relationship representing the routes from Paris to Nice into the working-area. The user clicks on the Validation button, inducing the drawing of the representation of the sub-query into the query-area, while the working-area is cleared.

The user begins the second step of the construction of the query (i.e., the sub-query representing "the intersection between a town of more than 100,000 inhabitants and a forest") by a click on the Zone button inducing the qualification of this zone as a town of more than 100,000 inhabitants through the database model. The representation of the town appears into the working-area. He repeats a similar sequence, qualifying the second zone as a forest. The representation of the forest appears into the working-area. The Network button is automatically set to "Geometric" since no ambiguity may arise for an intersection between two zones [4] (the semantics of the intersection operator is therefore a geometric intersection). The user now clicks on the intersection operator button, inducing the drawing of the relationship representing the intersection of a town and a forest into the working-area. The user clicks on the Validation button, inducing the drawing of the representation of the sub-query into the query-area, while the working-area is cleared.

The third step will express the fact that "the routes cross the town in its non-forest part." Since the line representing the routes is available in the query-area, the user clicks successively (1) on the Begin Selection button, (2) on the line representing the routes in the query-area, (3) on the End Selection button. The line representing the routes appears the into the working-area. The Global button is actually on the Global position (the default one). Since the intersection between town and forest is available in the query-area, the user clicks successively (1) on the Begin Selection button, (2) on the Global button, to switch it to the Partial position and allow the selection of the sub-part of an object, (3) on the non-forest part of the town in the query-area, (4) on the End Selection button. The zone representing the non-forest part of the town appears the into the working-area. The Network button is automatically set to "Geometric" since no ambiguity may arise for an intersection between a line and a zone [4] (the semantics of the intersection operator is therefore a geometric intersection). The user now clicks on the intersection operator button, inducing the drawing of the relationship representing the intersection of the route and the non-forest part of the town into the working-area. The user clicks on the Validation button, inducing the drawing of the representation of the sub-query into the query-area, while the working-area is cleared. In the query-area, the user now sees the drawing of routes between Paris and Nice, crossing a town in its non-forest part .

To build the last step of the query (i.e., the sub-query representing the "routes bordered by a lake before crossing a town"), the user first clicks on the Zone button. This choice induces (1) the access to the database model with which the user qualifies the zone as a lake, and (2) the drawing of the zone and the displaying of its qualification into the working-area. The Global button is actually on the Partial position. Since the line representing the routes between Paris and Nice is available in the

query-area, the user clicks successively (1) on the Begin Selection button, (2) on the first part (i.e., before the intersection with the town) of this line in the query-area, (3) on the End Selection button. The line representing the first part of the routes appears into the working-area. The user now clicks on the bordering operator button, inducing the drawing of the relationship representing the bordering of the lake and the routes into the working-area. The user clicks on the Validation button, inducing the drawing of the representation of the final query into the query-area, while the working-area is cleared.

The user can now validate the query. A click on the Validation button provokes the translation of the graphical query into the formal internal language [12]. The graphical editor is set up in the default configuration. The query resolution engine is in charge of managing the expression.

II.2. The data model handling

Several database models have been proposed to handle geographic information. They are based on (1) an extended relational approach with Abstract Data Types or a N1NF philosophy [2,6,7,15,16,17,24], (2) the object-oriented paradigm [25], (3) rules and facts [9] or (4) an algebraic approach [8,21]. Our goal is not to define a new data model. To simplify the presentation, we use an extended relational formalism since the relational model is widely accepted [20,22]. The toy database is a sub-set of the one defined in [3].

Town (name, population, economical_activity, cost_hotel, spatial_representation)
Forest (name, spatial_representation)
Polluted_area (#area, pollution_type, spatial_representation)
Transport (#transport, name, transport_type, price, spatial_representation)
Network (#line, origin, destination, departure_h, arrival_h, #transport)

The Network relation corresponds to the first level of abstraction of the relation Transport. It allows seeing the Transport relation as a logical graph. In the following a relation is called a theme. We present the extensions (the icons, the variables and the overlap of the spatial representation) defined on the data model to take full advantage of the graphical representation of a query.

II.2.1. The icons

The icons are introduced to get the screen to look attractive. An icon is defined for each theme and for each attribute. A basic object involved in a query belongs to a theme and may have some selection criteria. Let (c) be a query:

(c) Which are the towns with more than 100,000 inhabitants?

To materialize the labelling function the following rules are applied:

- The icon of the theme is displayed
  (unless the theme is defined with a disjunction i.e., theme = Town *or* theme = Forest);
- The icons of the non key attributes involved in the selection criteria are displayed i.e., the icon of the attribute population in the following selection criteria:
  theme = Town and population > 100,000;

- The key attributes involved in the selection criteria are displayed in the alphanumeric form (unless the key is defined with a disjunction, i.e.,
  theme = Town and (name = Nice *or* name = Paris);

The figure 2 represents the icons of the theme Town and the attribute population for the query (c).

Figure 2

The couple (icon of the theme, alphanumeric labelling of a key) allows distinguishing between two objects with the same name, i.e., a town named Vincennes and a forest named Vincennes (since the name is only defined as a key for a theme and not for the global database). Two objects in the query area with the same icon of theme are two independent objects (i.e., two towns). Otherwise only one object would appear since the selection button allows using an already defined object.

II.2.2. The variables

The imperative programming languages (i.e., C or Pascal) offer the concept of variable to store a partial result. The join operator of the relational model is defined with a theta operator (i.e., >, <, ≥, =, ≤) between two attributes. Let (d) be a query:

(d) Which towns have the same economical activity?

The associated SQL statement is presented figure 3:

```
Select   *
From     Town T1 T2
Where    T1.economical_activity = T2.economical_activity and
         T1.name < T2.name[1]
```

Figure 3

While defining the labelling function associated to an object, we introduce the concept of variable. The semantics is a mixing between the variables of the imperative programming language and the join operator of the relational model. A variable is defined to materialize a theta operator between two attributes. It represents a set of values (as the relational model does) and it is sequentially defined, labelling function of an object after labelling function of an object (as an imperative programming language does). A variable is graphically defined by a number in the icon of the involved attribute and is formally noted $Xi (i=1, ..., n). Figure 4 represents the graphical representation of a variable for the query (d).

Figure 4

The network-oriented operators involve a notion of order. Let (e) and (f) be two queries:

(e) Which routes go from Paris to Nice with the qualification that visited towns include Grenoble and Valence?

(f) Which routes go from Paris to Nice with the qualification that visited towns include Grenoble and then Valence?

These queries involve a qualification on the nodes of the graph modelling the transportation network. This notion of order may also appear while managing the edges. Let (g) be such a query:

(g) Which routes go from Paris to Nice with the qualification that in case of connection the departure time is greater than the arrival time?

To handle this notion of order, we define the concept of local variable. It follows the same philosophy as the variable of control while defining a loop in the imperative programming language.

---

[1] We introduce this clause to avoid the symmetry (a, b) and (b, a) in the result.

It is defined by an initial value, an end value and an expression involving the attributes. The constraint of the query (g) is formally represented figure 5:

```
Arrival_time [$X1 - 1] < Departure_time [$X1]
$X1 = 2 .. length[2]
```

Figure 5

II.2.3. The overlap of the spatial representation

Depending on the semantics of the theme, the spatial representation may represent a partition of the plane (i.e., the theme Town). Nevertheless, a theme may have no reason to define a partition of a plane (i.e., the theme Polluted_area). Let (h) and (i) be two queries:

(h) Which roads have an intersection with an urban part such as the length is less than 1 km?
(i) Which roads have an intersection with a polluted area such as the length is less than 1 km?

The semantics of the query is exactly the same: an intersection between two objects with an aggregate constraint. The user may require to formulate such queries using the same constructor. The query (h) can be expressed with the Extended SQL statement presented figure 6:

```
Select   *[3]
From     Transport, Town
Where    intersect (Transport.spatial_representation, Town.spatial_representation) and
         Transport.transport_type = 'Road'
Group by      Transport.#transport
Having   Sum (length[4]
              (intersection (Transport.spatial_representation, Town.spatial_representation) ) ) < 1
```

Figure 6

Such a structure for the query (i) would provide a wrong answer as soon as a road crosses two polluted areas having an intersection (the intersection is computed twice in this structure). The graphical representation allows defining the queries (h) and (i) with the same formalism.

---

[2] Length is a pre-defined function on the path operator providing the length (i.e., the number of edges) of a path.
[3] To simplify the presentation, we do not take into account here the constraint due to the Group by clause.
[4] Length is a pre-defined function on the geometric operator providing the geometric length.

Nevertheless, to guarantee a correct translation into the Extended SQL statements, the possible overlap of two instances of a theme must be defined in the data model.

II.3 The basic operators

To analyze the query and to translate it into DBMS understandable orders, a visual query must be represented with an internal formalism. A query is defined by a combination of several operators (the results of an operator may be used by any other operators). A functional-like language is well adapted to model such a combination. This part presents the operators retained to model a query.

The operators are defined as Abstract Data Type operators. We consider, here, the logical relationships between two objects. These operators are independent from the specifications of an operator following the spatial representation (i.e., is there an intersection between two spatial representations when they have only one point in common?). The operators involved in the Cigales language are represented in figure 7. These operators are decomposed into two parts: the user level and the system level. The user level operators are manipulated by the end-users. The system level operators are used to fill the gap between the semantic level of a visual query language and a DBMS query language. The semantics of the operators are defined in [1]. Each operator has two representations: the operator and the negation of the operator (except the euclidian distance and the before operator).

The user level operators: the geometrical intersection (∩), the euclidian distance (√), the adjacency (¤), the path (->), the inclusion (C) The system level operators : the difference (Δ), the union (∪), before (<<), the join of paths (-><-), the direct link (Π_D),the intersection of paths (Π_{is_e}), the inclusion of paths (Π_{ic_e}), the inclusion of stop place(s) in a path (Π_{ic_ne}), the intersection of stop places (Π_{is_n}), the inclusion of stop places (Π_{ic_n}), the beginning of a path (>->), the end of a path (-><)

Figure 7

The user level operators have various translations in the query modelling depending on the operands. The use of (1) the Begin selection, (2) the Global button and (3) the selected part of an object determine the correct system level operator. The grammar modelling a user defined query is presented in [12].

II.4. Conclusion

A visual query language is more natural to the end-users than alphanumeric query language [3]. By end-users, we mean people who may be experts in some domain other than programming language and who want to use a GIS to meet their particular needs. Our focus is not so much on the spatial analysis itself, but on the computational environment in which it is embedded (i.e., the query expressive power, the management of the graphical ambiguities and the semantics of the visual query language).

The spatial relationships (i.e., left_of) are not graphically handled for the time. They appear with the straight line operator (i.e., the angle definition [12]) but the relative orientation in the query area is still not provided because the semantics of such queries is very difficult to handle correctly [23].

III. EXAMPLE OF QUERIES

In this part, we present some examples of queries. These basic queries are part of the query language expressive power benchmark for GIS defined in [3][5]. The query Q_1 ("Which are the towns with more than 100,000 inhabitants?") involves a unique object and presents the notion of icon (i.e., the query c). The query Q_2 ("Which towns have the same economical activity?") involves two objects. These objects are not linked with a spatial operator (i.e., a graphical link between these two objects) but are linked with a variable (i.e., the query d). The query Q_3 ("Which towns are bordered by any forest?") presents two objects linked with a thematic-oriented operator (i.e., the borderingoperator). The query Q_4 ("Which routes go from Paris to Nice via the TGV train?") involves two objects linked with a network-oriented operator (i.e., the path operator). The query Q_5 ("Which roads have an intersection with an urban part such as the length is less than 1 km?") and Q_6 ("Which roads have an intersection with a polluted area such as the length is less than 1 km?") present two objects linked with a spatial operator (i.e., the intersection operator) defined with an aggregate constraint; these queries illustrate the notion of overlap (query h and query i). The query Q_7 ("Which roads do not cross towns with more than 100,000 inhabitants?") presents the notion of negation. The query Q_8 ("Which routes go from Paris to Nice with the qualification that they border a lake for a while and then cross a forest?") presents a complex query involving several objects and several operators.

---

[5] Due to space limitation, the step by step presentation of each query is not possible, therefore we present the final step of the query definition process (just after the query-area validation).

Figure 8 presents the query Q_1: "Which are the towns with more than 100,000 inhabitants?"

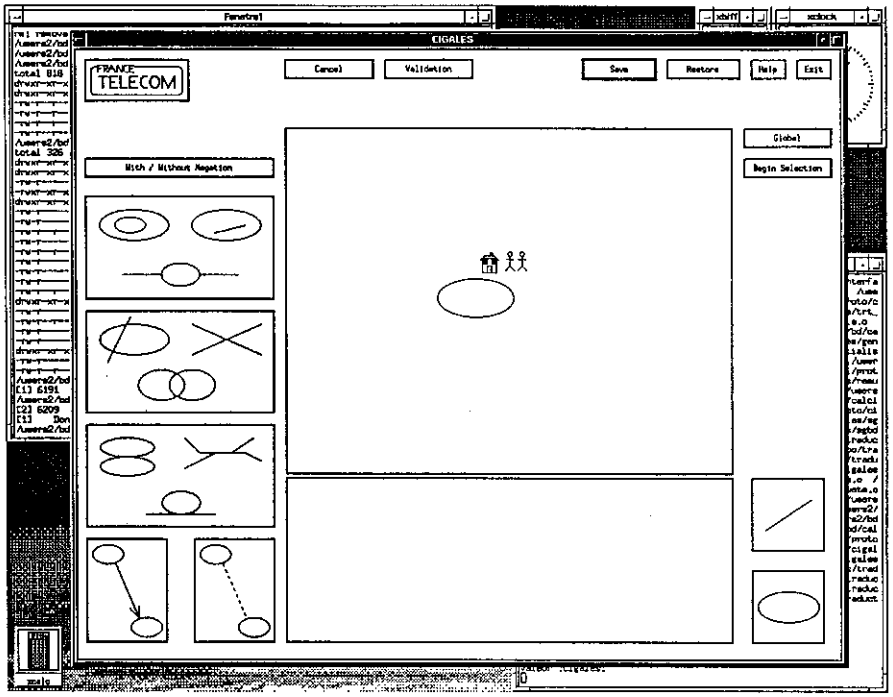

Figure 8

Figure 9 presents the query Q_2: "Which towns have the same economical activity?"

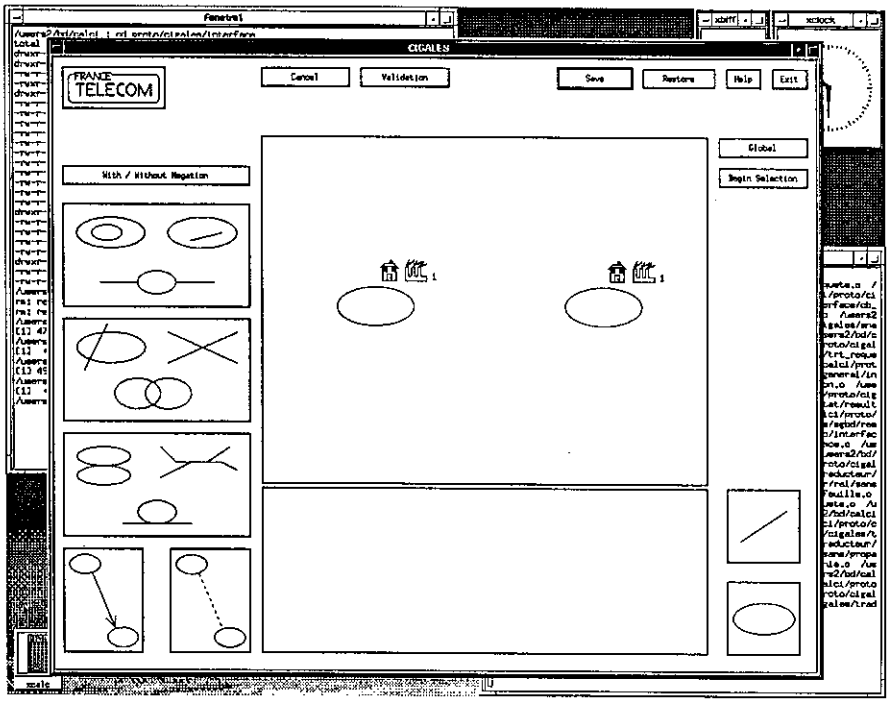

Figure 9

Figure 10 presents the query Q3: "Which towns are bordered by any forest?"

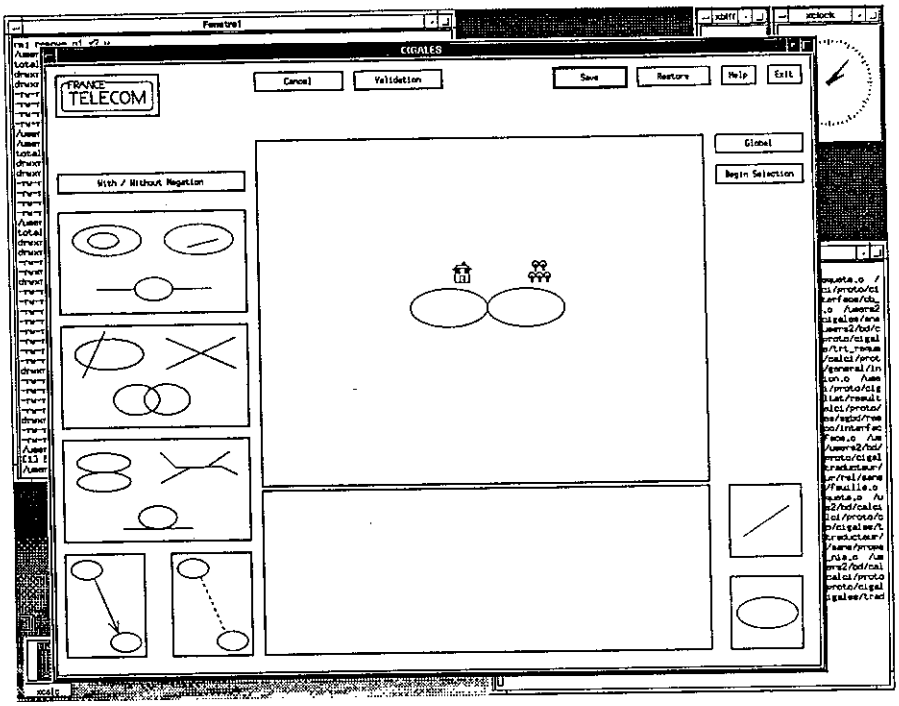

Figure 10

Figure 11 presents the query Q4: "Which routes go from Paris to Nice via the TGV train?"

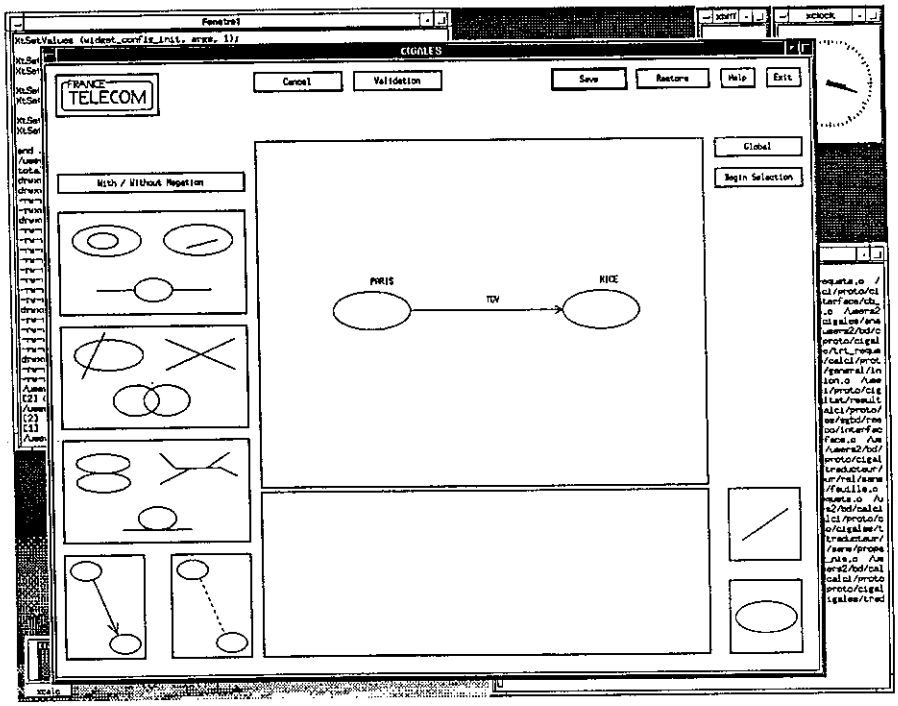

Figure 11

Figure 12 presents the query Q5: "Which roads have an intersection with a urban part such as the length is less than 1 km?"

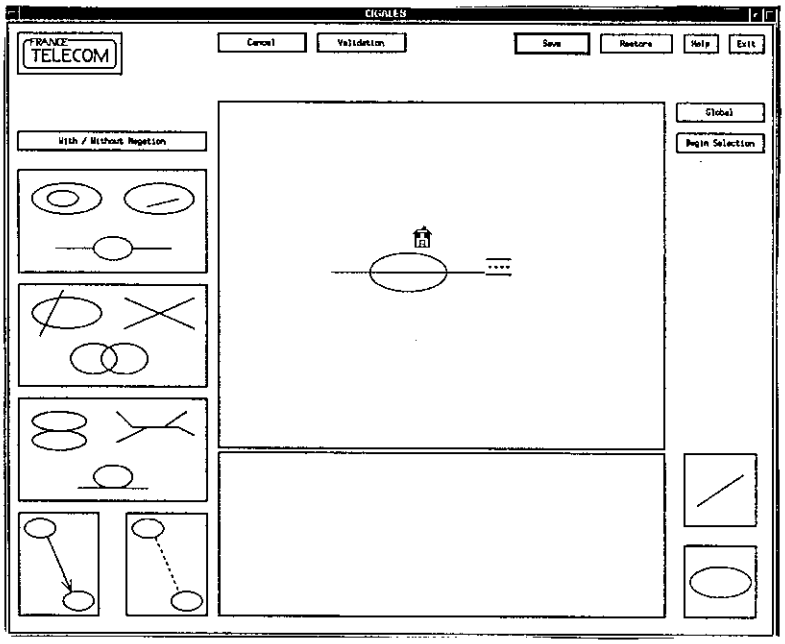

Figure 12

Figure 13 presents the query Q6: "Which roads have an intersection with a polluted area such as the length is less than 1 km?"

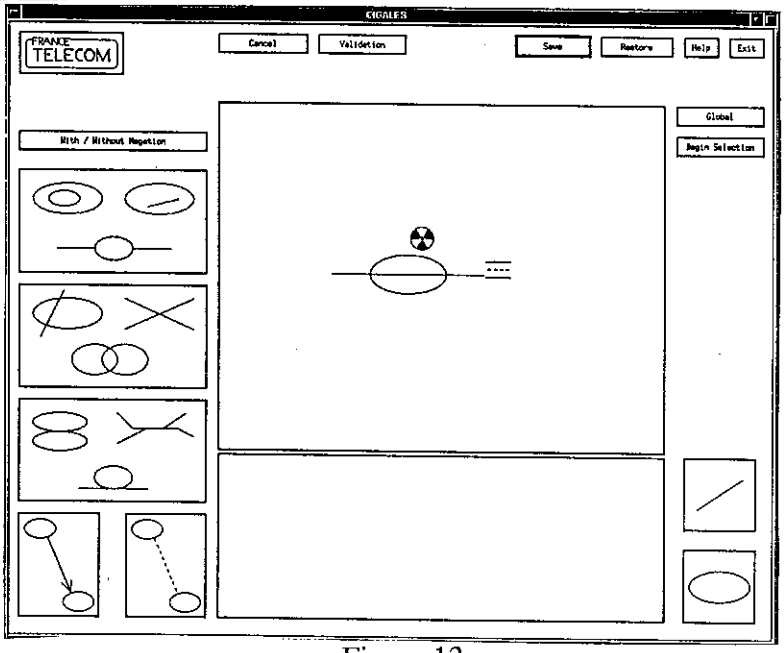

Figure 13

One can remark that the query Q5 and Q6 have the same semantics (i.e., an intersection between objects with an aggregate constraint). The ways of expressing such queries follow exactly the same philosophy. The difference is introduced while defining the label: Town versus Polluted_area (see the section II.2.3).

Figure 14 presents the query Q7: "Which roads do not cross towns with more than 100,000 inhabitants?"

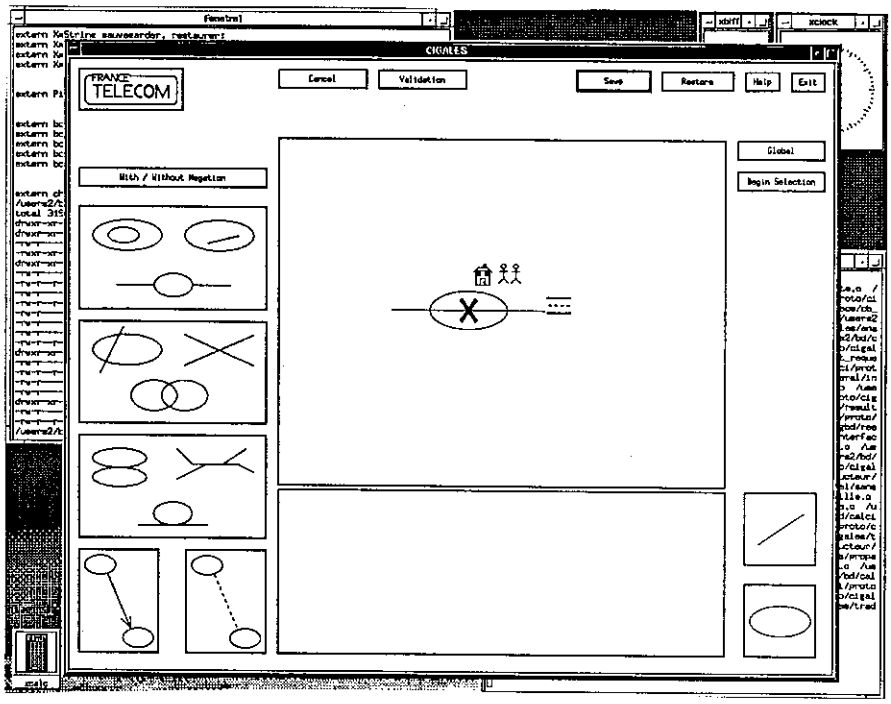

Figure 14

Figure 15 presents the query Q8: "Which routes go from Paris to Nice with the qualification that they border a lake for a while and then cross a forest?"

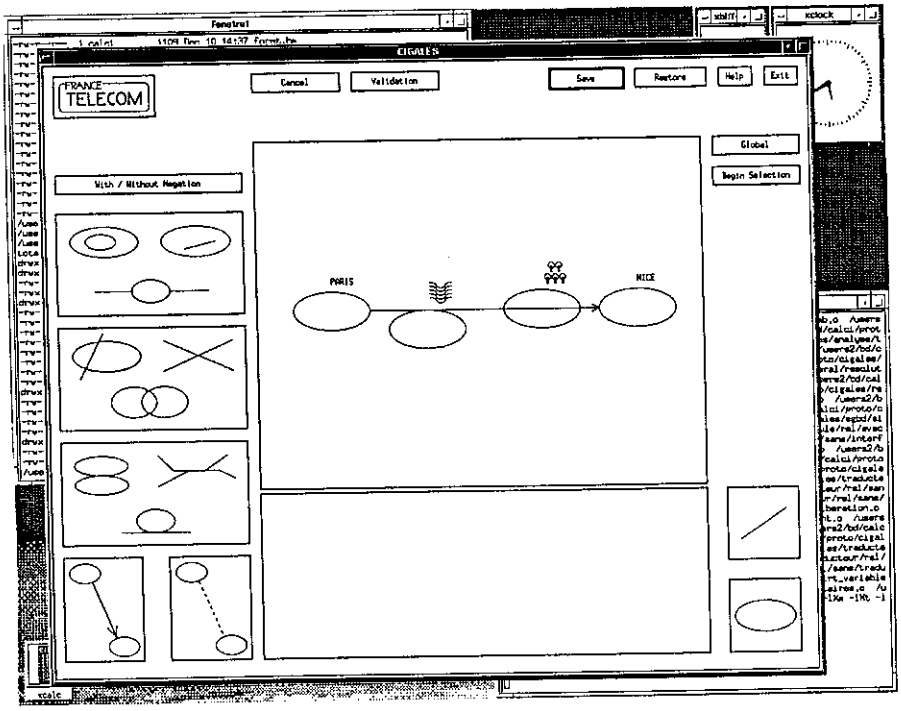

Figure 15

IV. IMPLEMENTATION

It is well known that programming user interfaces is difficult. There are a number of reasons that software for modern user interfaces is inherently more difficult to write than other kinds of software, i.e., iterative design, difficult to get the screen to look attractive, asynchronous inputs, error handling, aborts and undo commands. Three parts can be defined in the design of a GIS user-interface: the definition of a query, the management of the results and the update mechanism. Our current work is only oriented towards the query definition process. Several reasons explain our choice: GIS do not have a complete query language; the semantics of the Extended SQL statements is not very clear and needs to be formalized (i.e., the management of the results is not obvious).
This part presents the solution we adopted to design the structure of the CIGALES prototype and the user-interface.

IV.1. The structure of the CIGALES prototype

Since the design of the user interface is an iterative process, it is of prime importance to separate the user interface from the query resolution engine. Figure 16 represents the three major components of the prototype. The user interface is in charge of managing the definition of the query and provides a functional expression to the query resolution engine. The query resolution engine is in charge of analyzing the functional expression and translating it into Data Base Management System (DBMS) understandable orders (i.e., a set of Extended SQL statements). The DBMS is in charge of providing the alphanumeric and geometric results.

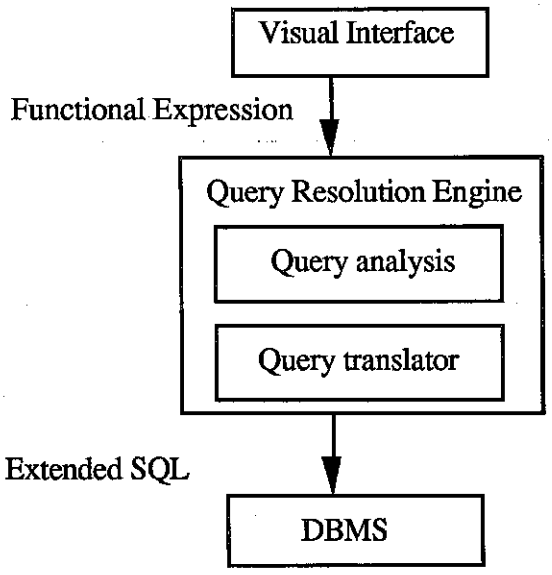

Figure 16

The use of a functional expression allows modifying the user interface without interfering with the query resolution engine and the translator allows modifying the DBMS.

IV.2. The user interface

IV.2.1. The modelling

Several formal tools can be used to design a user interface such as state transition diagrams, BNF grammar, petri-nets, event languages and production systems, declarative languages, constraint languages, high level specification languages or object-oriented languages.
We retain the state transition diagrams [1] to model the dynamic of the user interface and a BNF grammar to model the expressed queries.

We do not present the connexion or deconnexion processes, but we just focus on the states that are used to build a graphical query. The figure 17 is a synthetic representation of the finite states automata of the user interface.

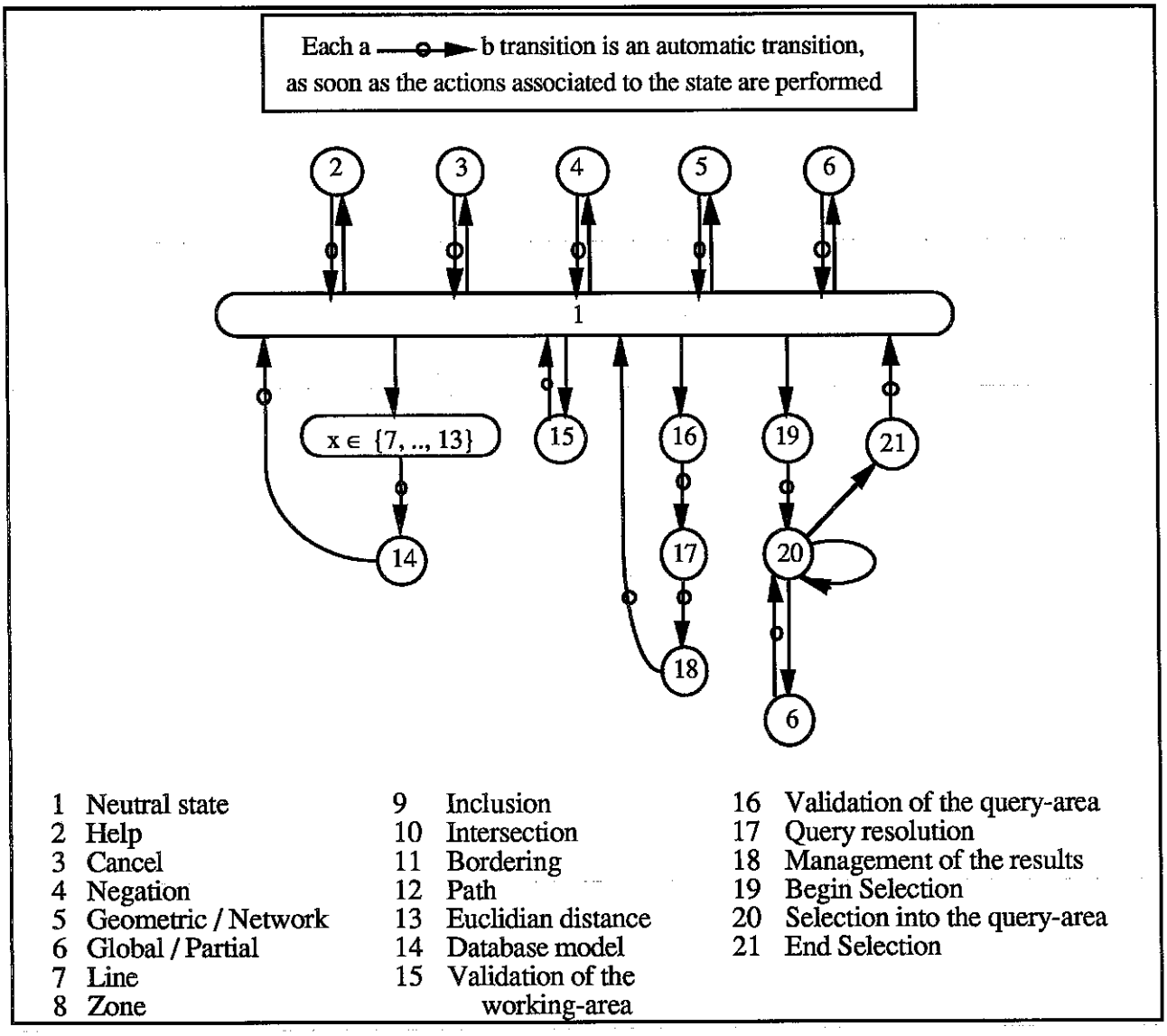

Figure 17

The rules presented figure 18 insure the coherency of the finite states automata. Let |w| be the number of objects in the working-area.

Rule 1 :	1 -> x (x ∈ {7, 8})	=> |w| < 2
Rule 2 :	1 -> x (x ∈ {9, ..., 13})	=> |w| = 2
Rule 3 :	1 -> 19	=> |w| < 2

Figure 18

IV.2.2. The management

The interface generators can help the implementation of a user-interface. They provide tools to form the general design and to build various components as menus or dialog boxes. Their main drawback is their inability to manage the specific objects of the application (i.e., the graphical objects linked to the semantics of the interface) because those objects change dynamically during a working session, contrarily to the static components of the interface.
In the Cigales interface, most of the components are easy to implement with or without an interface generator. The different buttons require the management of either a single mode or a switch mode, and the realization of the straightforward actions according to their semantics. The working-area requires drawing and clearing features. These requirements are easy to satisfy.
On the other hand, the management of the query-area is much more difficult. This area requires drawing and clearing features as the working-area does. Nevertheless, it also requires to handle the spatial relationships between the objects involved in the query. Two classes of problem appear: the management of a sub-query and the interactions between the sub-queries.

The first class of problems corresponds to the number of sub-queries involved at each step of the query definition process. A user may define different independent sub-queries before defining an interaction between one (or several) object(s) of a sub-query and one (or several) object(s) of another sub-query. If a final query has a high level of complexity, the query-area needs to provide enough sub-areas, one for each independent sub-query into the query-area. Since the number of sub-queries is dynamic, it may increase and/or decrease during the query definition process. Several solutions are possible within which: the re-drawing of the query-area at each modification of the number of sub-queries, the management of "sub-query-pages" with one page for each sub-query with a scrollbar mechanism, or the limitation to a fixed number of sub-areas to allow most of the end-user queries. The very last solution is the one we retained.

The second class of problems corresponds to the number of operators applied to an object. When an object is involved in two relationships, it is necessary to merge the two graphical representations corresponding to the two sub-queries. When this object is involved in several relationships, the

realization of the graphical merging becomes complex. To avoid this problem, we chose to limit the number of operators applied to an object by determining a fixed number of "interaction points" to allow most of the end-user queries. This solution is particularly available for the zone objects, since a line may be extended as much as necessary. For a zone, we have to provide several interaction points on the sides (for the intersection, bordering, path and Euclidian distance operators), as well as interaction points inside the zone (for the inclusion operator). In spite of the user-friendliness of the visual approach, the representation of a query may be confusing. We do not provide any difference in the visual representation for the Geometric/Network mode and for the Global/Partial mode (we rely on the user's memory).

IV.3. Conclusion

The static components (i.e., the buttons) and the working-area of the Cigales interface are easy to manage. On the other hand, the query-area presents various difficulties due to the dynamics of the drawing in this area. The problems result either from the management of several sub-queries or from the interactions between the sub-queries. The construction of a query cannot induce any ambiguity [12] at the formal internal representation level. Nevertheless, the graphical representation of a query may look ambiguous for the user. Providing an explanation facility might be a possible solution to avoid this problem.

V. CONCLUSION

Many applications need to manage multi-media data. Since the number of applications is considerable, it is of prime importance to offer an application-independent and DBMS-independent query language. Visual programming languages are very promising since they are more natural than textual query languages (i.e., extended SQL) and they offer a higher level of abstraction. Nevertheless, since the level of abstraction is higher, the interaction with the DBMS is more complex. This paper is based on the CIGALES experiment. Cigales is a GIS query language and follows a visual and declarative Query-By-Example philosophy. The end-user "draws" an example of the wished results to define a query. We present the higher level of the CIGALES prototype, the user interface. To do so, we recall the philosophy of the Cigales query language detailing some semantic problems such as the data model handling, we present some examples of queries and we present the current state of the implementation.

The Cigales prototype is developed using OSF/Motif (for the user-interface), Yacc (for the analysis of the internal functional expression modelling the query) and C programming language (for the query management) on Sun workstations. The future work is to study the management of the results (i.e., the ambiguities due to the aggregate functions, the independence between several components of the answer).

References

[1] Aho AV, Sethi R., Ullman JD: Compilers: Principles, Techniques and Tools, Addison-Wesley, 1986

[2] Bennis K., David B., Quilio I., Viemont Y.: Géotropics: Database Support Alternatives for Cartographic Applications, 4th Int. Symp. on Spatial Data Handling, Zurich, Switzerland, July 1990

[3] Boursier P., Mainguenaud M.: Spatial Query Languages: Extended SQL vs. Visual Languages vs. Hypermaps, 5th Int. Symp. on Spatial Data Handling, Charleston USA, Aug. 1992

[4] Calcinelli D., Mainguenaud M.: The Management of the Ambiguities in a Graphical Query Language for GIS, 2nd Symp. on Large Spatial Databases, Zurich, Switzerland, Aug. 1991, Lecture Notes in Computer Science (LNCS) n° 525

[5] Cruz IF, Mendelzon AO, Wood PT: A Graphical Query Language Supporting Recursion, Proc. SIGMOD Conf., San-Fransisco, USA, May 1987

[6] Egenhofer M.: Why not SQL !, Int. Jou. Geographical Information Syst., vol 6 n°2, 1992

[7] Frank A., MAPQUERY: Database Query Language for Retrieval of Geometric Data and their Graphical Representation, Computer Graphics, Vol 16, n°3, July 1982

[8] Güting RH, Geo-Relational Algebra: A Model and Query Language for Geometric Database System, Proc. of the Int. Conf. on Extending Data Base Technology, Venice, Italy, March 1988

[9] Jungert E.: Inference rule in a Geographical Information System, IEEE Workshop on Language for Automation, New-Orleans, USA, November 1984

[10] Kim HY and al: PICASSO: A Graphical Query Language, Software-Practice and Experience, Vol 18(3), 169-203, March 1988, Ed. J. Wiley and Sons Ltd

[11] Kirby K.C., Pazner M.: Graphic Map Algebra, 4th Int. Symp. on Spatial Data Handling, Zürich, Switzerland, July 1990

[12] Mainguenaud M.: From the User Interface to the Database Management System : Application to a Geographical Information System, 5th Int. Conf. on Human Computer Interaction, Oralndo, USA, 8-13 Aug. 1993

[13] Mainguenaud M., Portier MA: CIGALES: A Graphical Query Language for Geographical Information Systems, 4th Int. Symp. on Spatial Data Handling, Zurich, Switzerland, July 1990

[14] Myers BA (Ed.): Languages for Developing User Interfaces, Jones and Bartlett Publishers, Boston, 1992

[15] Sacks-Davis R., Mc Donnell KJ, Ooi BC: GEOQL: A Query Language for GIS, Australian and New Zealand Ass. for the Advancement of Science Congress, Townsville, Australia, Aug. 1987

[16] Schek HJ, Waterfeld W: A Database Kernel System for Geoscientific Applications, 2nd Spatial Data Handling, Seatlle, USA, July 1986

[17] Scholl M., Voisard A.: Thematic Map Modelling, 1st Int. Symp. on Large Spatial Databases, Santa-Barbara, USA, July, 1989, LNCS n° 274

[18] Shu N.C.: Visual Programming, Van Nostrand Reinhold Cie, New York, 1988

[19] Smith TR, Menon S, Star JL, Ester JE: Requirements and Principles for the Implementation and Construction of Large Scale GIS, Int. Jou. Geographical Information Syst., vol 1 n°1, 1987

[20] Stemple D, Sheard T, Bunker R: Abstract Data Types in Databases: Specification, Manipulation and Access, Proc of Int. Conf. on Data Engeneering, Los Angeles, USA, Feb. 1986

[21] Svensson P., Huang Z.: Geo-Sal: A Query Language for Spatial Data Analysis, 2nd Symp. on Large Spatial Databases, Zurich, Switzerland, Aug. 1991, LNCS n° 525

[22] Ullman J.D. : Principles of Databases Systems, Computer Science Press, Maryland 1982

[23] Vandeloise C.: L'espace en français, Seuil Ed., Linguistique (in french)

[24] Vijlbrief T., Van Oosterom P.: The GEO++ System: an Extensible GIS, 5th Int. Symp. on Spatial Data Handling, Charleston USA, Aug. 1992

[25] Zdonic S., Maier D.: Readings in Object Oriented Database System, Morgan-Kaufman, 1989