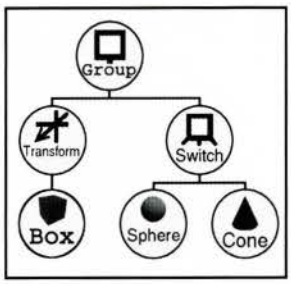
Fig. 1. An example of a scene-graph.



Hassoun Karam and Pierre Côté

Knowledge-Based Representation in the Design Studio: A Case Study Investigation

Reasoning on architectural knowledge representation, especially when studying architectural transformations, relies upon the definition of the modeling operations, their structured implementation, and the input of the user. Manipulating a given transformation is a cognitive process based on the possibilities of interaction defined by the system's designer. The user needs to know and understand the meaning of the available operations in order to use them; that is where semantics come in. Semantics are a fundamental element in artificial intelligence[1] and its study leads to a variety of theoretical approaches, representation formalisms, and practical applications. The emergence and expansion of the World Wide Web, made it necessary to add to these another interactive component: the 3D data exchange standards across the net.

The scene-graph is a widely used data structure; it is the underlying core of the four main 3D data exchange standards on the Web: Open Inventor, VRML, Java3D, and X3D. VRML was chosen for the design studio applications because it is widely supported in 3D graphic modeling software, it implies a relatively low learning curve for the students, and, as we will see later, it supports knowledge representation.

Our general objective was to enhance the representation threshold of the VRML, by mapping some of its functional properties to the interactive operations applied by a user on a 3D representation. For that purpose, and in the first part of this article, we introduce the scene-graph and a set of semantic relations according to the technical specifications of the VRML 2.0. We then define a hierarchy for those relations within the context of the VRML syntax and its elementary semantics. The construction of the hierarchy and the modeling of our VRML system both stem from a systemic approach to knowledge representation. In the second part, we consider a practical application of that hierarchy and represent an example of transformation taken from the site of our design studio project. A set of architectural transformations with a common pattern were encountered in the historical neighbourhood, transformations such as adding an upper floor, extending the main elevation sideways, joining two buildings, changing the shape of the trussed roof, etc.[2]

In the design studio, we intended to represent two transformed consecutive states of the selected buildings, both described by a number of geometric and topological parameters such as height, width, length, position, and orientation. To that effect, it was necessary to extract useful information from the available resources to formulate a transformation hypothesis and visualize it using VRML. Using one building as an example, we will describe the available documents (text and graphics), the steps in the mining of information, and how the semantic hierarchy could enhance the representation of the targeted architectural know-how. We will conclude with a discussion of the results and a reflection on some of the lessons that were drawn from the project. Finally, we will outline some future perspectives relative to the application of such representation in the design studio.

The Scene-Graph

The scene-graph is a data structure used to represent, visualize, and manipulate information about our world. It is a system of nodes and arcs that can be described according to its structure and functions. In the design studio, we decided to work with the VRML scene-graph version. Although the above-mentioned standards have much in common, we will exclusively refer to the concepts of VRML in the present discussion.

A VRML scene-graph contains information about the graphical description of the scene such as appearance and geometry of the objects, lighting and viewing conditions, possible user interaction, and generation of events.[3] When the scene-graph is considered from a structural point of view, it is seen as a set of relations defining the syntax of the programming code, which expressions are allowed and which are not, where they fit in the structure, and how their sequences must be arranged.[4] Each graph has one root node and arranges the description of the scene in a branching layout of nodes, representing the objects, and arcs, representing the relations among them (fig. 1). That layout is a basic semantic terminology to define the types of possible directional relations among the nodes: the parent node and

Hassoun Karam is a Ph.D. candidate in computer assisted architectural design (CAAD) at the School of architecture - Université Laval, Québec. As a member of the Laboratoire de recherche sur l'identité par modélisation architecturale (LIMA), he is interested in architectural transformations, their uncertain digital representations and relationships with architectural identity.

JSSAC / JSÉAC 27, nos 1, 2 (2002) ; 29-37.

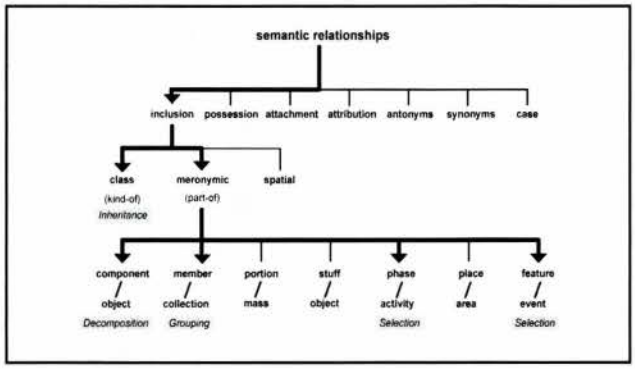


Fig. 2. A semantic hierarchy.
(adapted from Storey 1993)

the child node. All the nodes can have a certain number of parents and children with the exception of the root node (it cannot have a parent) and the leaf nodes (nodes that do not have children). Each branch starts from the root and covers a path of nodes and arcs leading to a leaf node; it contains the complete information to describe the leaf node in terms of shape, appearance, and behaviour. Consequently, the predefined properties of the nodes influence the graphical modeling of the observed objects (naming the objects, grouping them) and provide the students with a common base to analyze the architectural elements and understand their complexity and varying details.

The apparently rigid top-down hierarchy is "flexibilized" by a layer of *Routes*. *Routes* are nodes that constitute a network linking the components of the nodes; they can be compared to a web of wires connecting command centres and supporting the flow of information among them. We consider the *Routes* network as a functional representation of the scene-graph. Thus, the larger interaction among the components that is supported by the functional aspect of the *Routes*, overcomes the apparent rigidity of the sequential arrangement of the nodes. Moreover, both the functional properties and the relational hierarchy of the scene-graph complete its definition as a "system." That definition induces the dimension of emergence within the representation and, as we will discuss later, that dimension in turn supports the validation of the architectural transformation hypothesis.

Another systemic dimension of our approach is the possibility to include the subjectivity of the observer, i.e. the student, in the representation formalism. During the modeling phase of the studio, the students used a comprehensive index of architectural details[5] to identify the elements to be modeled within the context of the project and to name them accordingly. They analyzed the collected architectural information (from site visits, texts, photographs, and drawings), identified the constituent parts, and decided which were relevant to the representation or were not, hence the subjectivity. In the context of knowledge representation, that selection process is considered to be a part of the conceptualization of the target knowledge and to subsequently create the universe of discourse, i.e. "the set of objects about which knowledge is being expressed."[6]

Within the scene-graph, that procedure is actually an abstraction operation supported by the modularity of the nodes. We consider the representation of the subjective contributions of each team of students as a restitution of the architectural and historical contexts that affected the state of the neighbourhood; restitution used to enhance the theoretical input in the design studio, as Mitchell described.[7]

> In a given context, the set of objects to which we refer, and about which we can express comments, is the universe of discourse. In discourse about a building the universe invariably includes the parts of the building to which we choose to give attention. Different people may decompose a building into parts in different ways and so include different collections of parts in the universe of discourse.

The Semantic Relations

The support of the scene-graph to a given representation is implemented, in effect, within the nodes of the VRML. These nodes can contain graphical and spatial attributes as well as others more abstract such as geometric constraints, parent-child relations, etc. The positional organization of the nodes in the graph is managed by a set of specialized nodes. Their role is to define relations among the other nodes and constrain the behaviour of the graph. In the field of knowledge representation, such relations are chosen from a large set of meaningful descriptions of the interaction pertaining to the objects observed.[8] They are referred to as semantic relations and used to model constraints, inheritance, operation propagation, or specialized query capabilities.[9] Basically, semantic relations formalize meaning and structure of the represented knowledge by using abstractions such as inclusion, aggregation, association, possession, attachment, and attribution.[10]

Besides, we define the term "semantic relations" within the context of artificial intelligence and in relation to coding algorithms. We understand it as the outcome and effect of the application of algorithms and procedures.[11]

While selecting our relations, we relied on strict differences among the nodes; differences that exclusively map each node to a functional outcome. For instance, more than one type of specialized nodes offer the possibility to organize nodes in a group; yet we have chosen the most basic type (*Group*) that does nothing except define an element-set relation and maintain this strict membership among the children nodes. Finally, because from the onset of the studio, the work of each team of students was viewed as an abstraction input, a summary of the group's analysis, into the others', and supported by the uniformity of the syntax of the scene-graph, it was possible for the representation to contain the set of potential interpretations of the observed neighbourhood, thus contributing to the semantics definition of the model.[12]

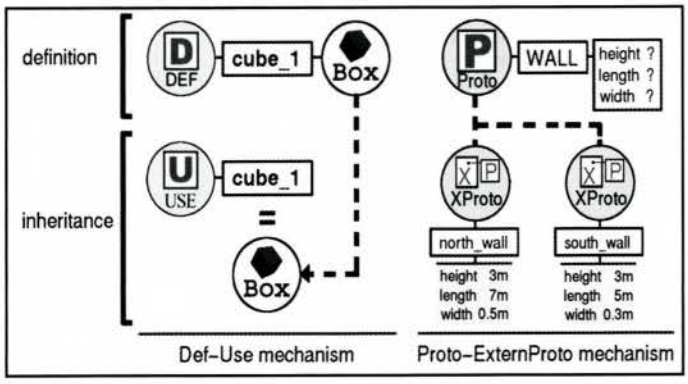


Fig. 3. Inheritance mechanisms.

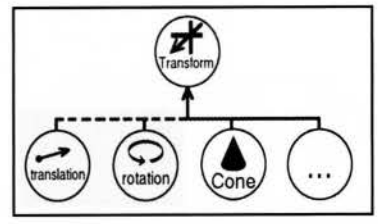


Fig. 4. The Transform node containing spatial operators: translation and rotation.

We have identified four VRML categories of semantic relations to represent processes of architectural transformations: inheritance, decomposition, grouping, and selection. Essentially, our manipulation of architectural objects can be situated within the general semantic category of inclusion as described in figure 2, drawn from Storey.[13] Although that hierarchy was compiled from various disciplines such as computer science, linguistics, logic and cognitive psychology, it truly supports our description of architectural transformations and shows how the chosen relations fit into that structure (gray-shaded in fig. 2). We use it to represent the controlled flow of transformations down the graph, the stable interaction among the elements of the scene, and a set of explicit and permanent constraints.

In the following paragraphs we develop the definitions of our semantic relations within the specifications of the scene-graph and refer the reader to Storey[14] for the description of the general categories.

Inheritance

Inheritance is the transmission of a set of properties from an ancestor to a descendant. In the VRML, it deals with (1) the propagation of attributes such as position in space, dimensions, orientation, colour, shape, and scale; and (2) the reuse of the code. It is implemented in two ways: the basic naming mechanism of DEF-USE, and the more general ones of *Proto* and *ExternProto*.

Those mechanisms differ when applying two distinct levels of inheritance derived from object-oriented programming (OOP): (1) in DEF-USE, the *addition* introduces new attributes and behaviours to the target representation; and (2) in *Proto/ExternProto*, the *substitution* replaces the initial values of an attribute or behaviour with a new one.[15] Consequently, the DEF-USE mechanism implements an imperative inheritance; the user cannot change the properties passed down the graph, and the objects are instantiated as specified by their ancestor (fig. 3), while the *Proto/ExternProto* mechanism is similar to the class-object inheritance mechanism in OOP and it uses *substitution* along with *addition* to manage inheritance.

When the modeled objects were converted (in the studio) from the original graphical format to VRML, it became tedious to redefine general classes based on those converted properties. The use of *Proto* then became necessary to recuperate the geometrical information and allow the addition or substitution of the properties needed to visualize the architectural transformation. Those properties were mainly related to the appearance of the objects (texture and colour), inherited by the DEF-USE node and unaffected by the geometrical transformations.

Decomposition

Decomposition is the inclusion within a given node of all the information it needs to function within the scene. Although it is the equivalent of the encapsulation property of OOP, we gave it a different name to designate its usage within VRML, especially because it is implemented by a defined node, *Transform* (fig. 4).

If an object is composed of a set of parts containing the information to be modified by the user, these have to be explicitly declared in the code as such. The basic definition of the geometry of an object in VRML is the SHAPE node, but that keyword does not allow any relation with other objects: it can only be displayed on the screen. The *Transform* node allows the user to interact with the object. That node implements the notion of encapsulation in VRML, on top of a SHAPE node, and it indicates a first level of possible interaction with the basic geometrical description of an object. In addition, it provides a nesting of shape information and, consequently, supports the detailed decomposition of an object into its parts. *Transform* is also a separator that defines the limits of the interaction of a given object with others in the same scene and it is the only means to pass rotation, translation, and scale transformations down the graph.

Although we have identified decomposition as a separate relation, as noted by Stefik and Bobrow,[16] it effectively supports the inheritance mechanisms (fig. 3) and the abstraction of the represented knowledge by its granularity dimension. Granularity supports the limited and the local aspects of the modifications applied to a separated part while maintaining its relations with the others.

Grouping

Grouping states that a number of separate objects should be treated as one. In VRML there are a number of grouping nodes: *Transform, Switch, Inline, Billboard, Anchor, and Group*. The *Group* node simply concatenates the objects and allows them to be processed as a single entity; it maintains a strict element-set relation and offers no interaction among them, i.e. the object does not acquire additional characteristics or perform any function within the group. Logically speaking and to some extent, it is equivalent to the AND operator. In the hierarchy, *Group* always defines a higher level of manipulation and semantically

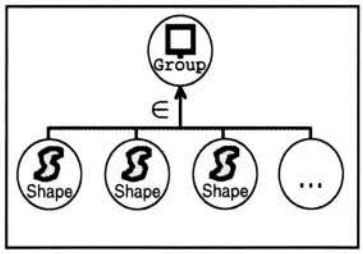


Fig. 5. The membership relation implemented by the Group node.

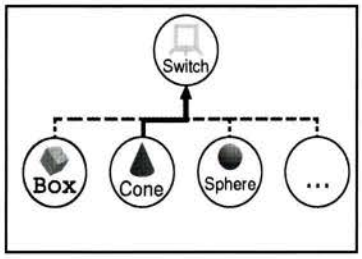


Fig. 7. A Switch node selects one element from a set.

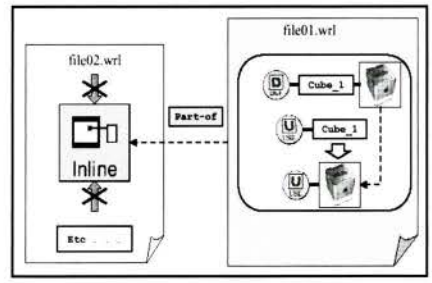


Fig. 6. The Inline node includes a child file into a parent file.

expresses the imperative requirement of a whole in terms of parts in an aggregation operation.

Inline is another grouping node and it is used to insert a VRML file (or a set of files)–a child file with all its nodes and properties–into another one, a parent file (fig. 5). It supports the construction of a main scene-graph from other sub-graphs residing in other files. An interesting property of that node is its scooping capacity. When defined within a child file, a DEF-USE node cannot be accessed from the parent file. The child file protects its components from external manipulation and the whole inserted sub-graph is treated as a single object. Translated into existential graphs and first-order logic, that node defines the logical implication: the *modus ponens*. For instance, given two entities P and Q, and P entails Q, if (and when) P is true, then Q is necessarily true. Besides, that node establishes a relation to the child file components. Any interaction defined in the parent file and implemented upon the "*Inline*-ed" child file is semantically equivalent to the application of the interaction to the components in the child file (fig. 7).

Selection

The previous relations remain static in the sense that, when they are defined within the hierarchy and without user input, they represent simple constraints in the graph, each according to its context and requirements.

VRML comprises a set of interaction nodes to translate the decision of the user into an operation on the scene. And since the scene is represented by a graph, it is useful to have a path selection mechanism that can isolate a relevant part of the scene (we call it a local situation) and allow the user to interact with it.

Choosing a particular path is made possible with the *Switch* node. Depending on the user input (a click on a object, a simple pointing operation, etc.), that node identifies a part of the scene, constructs an implementation path throughout the graph reaching that specific part, and implements the required actions or operations (fig. 8). Combined with the Routing mechanism (see § 2), *Switch* is equivalent to a dynamic version of the logical operator OR. It implements a control interface (such as *if… then…*) and semantically expresses the dependency of two nodes by the propagation of the chosen operation along the path. Various *Switch* paths can be created throughout the graph adding a spatial representation of the exclusiveness dimension[17], which, in turn, describes regions of applicability for each operation.

The circumscription of a local situation (or more) has been used in artificial intelligence as a means to construct ontologies.[18] By supporting the identification of the frequent situations, i.e. common architectural transformations, and conserving them as explicit nodes, the scene-graph provides the students with a practical way to enhance the abstraction of the observed information, and, at a later phase, the restitution, within the project, of the abstracted architectural properties of the site.

The Elaboration of the Semantic Relations

The elaboration of the semantic relations can be summarized with two phases of the modeling process:

– The *initial outset of a scene-graph*: On one hand, the qualitative description of the graph can be deduced from the topological properties of the relations (their location in the graph and their mutual links). On the other hand, the existent sets of transformation processes define the quantitative attributes to be manipulated anytime during the interaction. In the graph, the state of the building is described by a number of architectural elements and the transformation of a state is viewed as any combination of the addition, subtraction, or modification of an element. When the qualitative and quantitative properties are mapped to a state of the building, a series of possible design representations of the same building are obtained, with small modifications of those properties.

– Its *subsequent modifications by the user*: When the user modifies the scene-graph, its state remains qualitatively the same until there is a major modification of any property; then it shifts to the next state. That phase supports the various interpretations of the described scene and creates new emergent properties for the target nodes; the perceived effect of the interaction among the relations represents the emergence process in the graph, enhances its abstraction capacities, and completes its systemic description, with emergence defined as the process of "making properties which were not explicitly represented at the outset become represented explicitly."[19]

– The *application of the internal control mechanisms among the nodes*: The control mechanism has to maintain an information consistency of the graph to allow the bi-directional exploration of the representation, from state 1 to state 2 and back. It is important to note that, at a conceptual level, that exploration is not strictly circular, i.e. always coming back to the same starting point, it rather spirals: a first step of exploration from state 1 to state 2 is coupled with an emergence of a particular property of state 1 which,

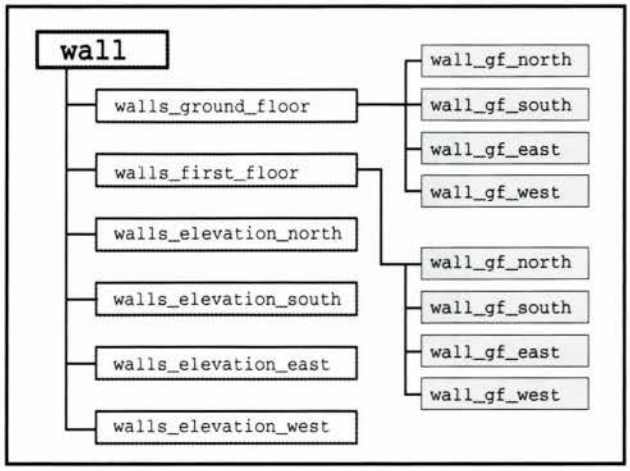


Fig. 8. A three-level hierarchy.

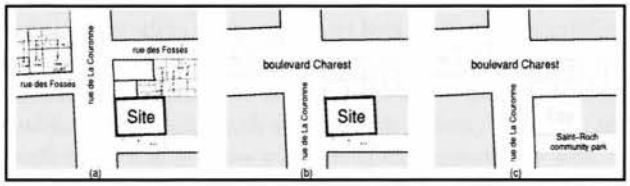


Fig. 9. The states of the site transformation: a- initial state, b- widening of Des Fossés street, c- actual state.
(Documents from the Archives of the City of Quebec).

when returning from state 2, would be re-integrated in state 1, increasing its interpretative potential.[20]

The Relational Hierarchy

Our semantic relations being defined, they have to be structured into an interactive hierarchy: this is based on the definition of generic objects with default features that can be changed during the construction of the graph and/or the user's interactions. For instance, it is possible to define for each craft (masonry, roofing, carpentry, etc.) a set of basic *Proto* nodes that could be instantiated into visual objects on the screen. In addition, interactivity is defined in terms of a set of operations allowing the user to modify the scene-graph and a set of constraints controlling and propagating those modifications among the nodes of the graph, i.e. the objects in the scene.

The top level of the hierarchy contains a general abstract *Proto* (fig. 8, 12); it defines the general characteristics of objects that will eventually be instantiated, viewed, and manipulated. In our example, it is the *Proto* "Wall" that encapsulates the default properties and behaviours of all the walls that we wanted to represent and manipulate. The next level contains intermediate *Proto* nodes that inherit the default properties from the previous level and add new ones. For instance, having inherited the basic attributes of length, width, and height from the parent *Proto*, the *Proto* "horizontal_walls" adds to them the possibility of being translated sideways, on a horizontal plane. At the bottom level of that tree-like hierarchy, are instances of those Protos, i.e. physical objects such as walls. Each one has inherited its properties from its parent *Protos* and behaves accordingly (fig. 8).

Moreover, we considered that the transformation operations were applied at two architectural levels: the floor and the elevation, and we chose to map them both to the behaviour of the instantiated objects. That choice is made explicit by the naming protocol of the objects and the defined geometrical and visual transformations. For instance, the texture of the northern ground floor wall could be changed from wood to brick without affecting the northern wall on the first floor, and all the walls on the eastern elevation of the building could be eliminated from the scene-graph without affecting the others.

An Architectural Transformation Case

The proposed relational hierarchy supports the representation of the transformation processes that shaped the built landscape of the Saint-Roch neighbourhood. One of the objectives of the design studio was to help the students understand the changes in the neighbourhood and invest that knowledge in the design of a project. Our site is an architectural example covering three aspects of the morphological and urban transformations that occurred through the city: the infrastructure (allotment, street patterns, etc.), the building mass (the elementary unit of the landscape), and the elevation surface providing the visual evidence of the changing identity of the urban landscape.

The case study of the building chosen constituted a challenging task of representation: (1) to integrate the information collected from various sources in order to complete the detailed history of the transformation hypothesis; and (2) to compensate, by using the scene-graph, for the missing architectural information on the demolished buildings. The available documents were photographs taken during the construction phase of the building, after its completion, and after its transformation. A study by Noppen[21] described and documented the sequential phases of the transformational processes of the site and provided the historical background for our model (fig. 9).

The Given Hypothesis

The following hypothesis covers the three levels of urban transformations as noted previously.

The building was located at 52 La Couronne Street, in the Saint-Roch area of Quebec City (fig. 9). Two buildings from Des Fossés Street that would later become Charest Est separated it. Such a configuration had the effect of imposing on the buildings to have their main elevation on La Couronne. As a direct result of that widening, our building acquired an access to the main street, Charest Est, and thus the transformation became inevitable. The building had to have its main elevation on Charest Est, but La Couronne was still thriving and the projected transformation had to take into account that second element.

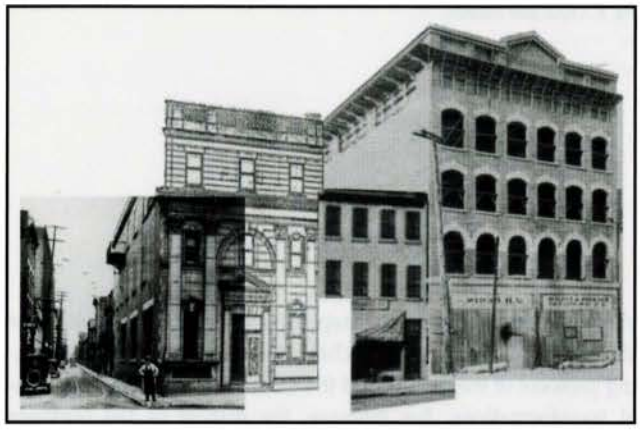


Fig. 10. The reconstructed site. (Archives of the City of Quebec).

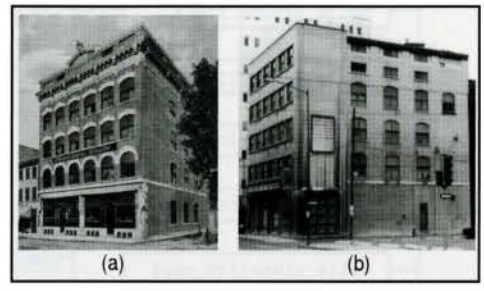


Fig. 11. The transformed states of the chosen building: (a) initial and (b) transformed. (Archives of the City of Quebec).

Consequently, both elevations on Charest Est and La Couronne had to be refurbished and re-established.

The information available on the photographs and the documents had to be cross-examined in order to validate the hypothesis. The intended validation did not imply an exclusive identification of a unique solution, but rather a plausible explanation of a possible architectural configuration.

The site did not contain any landmarks or architectural remains of its previous state; the building was completely demolished in 1994 and replaced by the Saint-Roch community park. The hypothesis was thus addressed from two different angles.

The first was the assembly of a 2D-graphical scene of the initial state of the building. We relied on documents available from the archives of the city of Quebec and on the study by Noppen[22] to produce a composite representation of the urban landscape and the chosen lot. The representation mapped the textual descriptions of the transformed building to the sparse photographs, maps, and architectural plans and provided a visual support to understand the sequence of the transformations as described by the documents.

The second approach was a 3D VRML model, with the objective of having the visualization of the transformation dynamical and temporally controlled by the user: the graphical display shows a sequential replacement of the transformed architectural elements and the user can stop and re-run the visualization at will. Moreover, the interactive movement of the user within a VRML model provides various viewpoints from which the transformation process can be observed against the surrounding urban environment. Consequently, one can easily grasp the qualitative properties of that architectural intervention and evaluate its impact.

The Transformation Process

To implement the validation approach, we needed to visualize the transformation process between two well-defined states of the building. An initial one that lasted until the widening of the main street of the neighbourhood, and a final one that lasted until the demolition of the building. The transformation processes considered are the results of sequences of the construction operations and can be broken down to their elementary actions defined within a range of know-how and skills: extending, moving or adding a wall, filling in a window, creating an opening in a wall, and so forth.

There are three phases in the transformation history of the site that can be summarized as follows: (1) the construction of the building itself as illustrated by a photograph of the construction site (fig. 10-a); (2) after the widening of Charest Est, the total refurbishing of both front elevations on La Couronne and Charest Est (fig. 10-b)[23]; and (3) the complete demolition of the building in 1994. The photographs showed the addition of a fourth floor, just under the roof, portrayed by the opening of new windows. On the southern elevation, a large opening was made to allow the circulation of large stocks of goods in and out of the building.

To complete the description of that process, we suppose that the eastern and southern elevations remained structurally unchanged and that their transformation was mainly at the textural level. Other changes in the building most certainly resulted from by the refurbishing of the north and west elevation. Nevertheless, we assume that transformation operations related to other components, such as floors and internal walls, can be omitted without affecting the integrity of our approach, and that the modularity of the scene-graph will allow us to add those transformations at a later stage.

The Transformation Graph

The scene-graph includes two states of the given architectural elements in the building (fig. 11). The first one describes those elements before the widening of Des Fossés Street, and the second one illustrates the operations carried out afterwards. To demonstrate the case study, we chose to represent the replacement of the western and northern elevations.

Both states were modeled using graphical software that does not support explicit knowledge representation in its data structure and they were then converted into VRML format in order to represent the transformational process. The converted files contained encapsulated information; each node described the visual, spatial, and geometrical properties of the corresponding architectural element. That resulted into the impossibility to extract the transformation operations already embedded within each object and we had to use both VRML files (one for each state) in the scene-graph. The advantage of that was the ease with which we replaced one initial state with its transformed state by using the selection relation; the disadvantage however was that important information had to be manually extracted and re-coded in VRML.

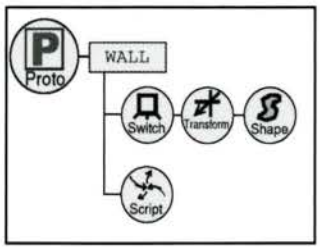


Fig. 12. A sub-graph describing the modular prototype of a wall.

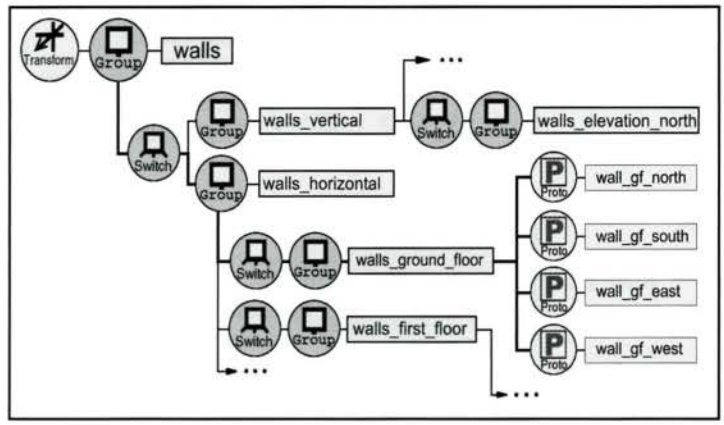


Fig. 13. The scene-graph of the transformed building.

We represented the transformational process as a network of positional relations coupled with a second functional network to selectively link those elements and provide the transmission of commands and control through the system. We used the *Route* nodes as the main component of the functional network to wire the nodes of the structural network. As mentioned earlier, routing is a powerful feature of VRML and, eventually, the internal manipulation of the graph and the user's interaction will be conveyed by those wires.

We built a *Proto* node containing the description of the target walls (geometrical properties and behaviour) and placed it at the top of our hierarchy. That *Proto* node served as a general class from which we instantiated all the basic objects in our scene, i.e. the walls. It comprised the following three elements. First, a *Switch* node to allow the individual selection of the child object among a group of objects; that node also enhanced the flexibility of the hierarchy and controlled the inclusion of this object in it. The second component was a *Transform* node to add a local coordinate system and allow the spatial transformations to take place. The third component was a SHAPE node to describe the geometrical properties of the wall; it explicitly refers to the converted VRML model as discussed in the previous paragraph. The initial SHAPE component, describing the initial state of a given wall, will be replaced with the transformed state of the object. Since we were particularly interested only in geometrical transformations, no appearance nodes were used to keep the file size small and facilitate its manipulation. In addition to those nodes, the *Proto* included various scripts to control the user's interaction and maintain the consistency of the graph.

Each of the instantiated walls could belong to two separate groups: the floor group and the elevation group. On one hand, when we want to visualize the phases of the construction process, the floor groups are shown successively, starting with the ground floor up to the roof. On the other hand, to show the transformation of the north elevation by replacing the initial phase with the transformed one, all the walls in the various floors need to be selected and replaced; that is done by selecting the north elevation group.

These two levels of interaction are complexified by adding two other levels: the horizontal one, combining the floor groups and the vertical one, combining the elevation groups. One use of the latter is exemplified when we want to completely erase the building from the scene; to do so, we select the vertical walls group (or the horizontal one) and the graphical representation is erased from the screen.

The arrows pointing to the dots replace the repetitive parts of the scene-graph that were eliminated for graphical clarity. In the "walls_vertical" group, the arrow points to three other groups containing the corresponding elevation walls for the south, east, and west. These groups are similar to the one shown in the graph (walls_elevation_north). In the "walls_first_floor" group, the arrow points to four corresponding *Proto* nodes of the first floor, similar to those of the group "walls_ground_floor."

Discussion

The validation of the given hypothesis (i.e. the sequential transformation of the building) is demonstrated by the actual representation of the architectural transformation within the scene-graph. It relies on two semantic (i.e. effective) properties of that representation. The first one is mapping the extracted declarative knowledge to a graphical proof of a possible interpretation expressing the construction processes that could have been used. The second, more abstract, deals with the integration of that interpretation into the architectural and urban history of the building site, and argues for the exemplarity and the re-usability of such a representation. Moreover, the translation of the documented transformations into a set of structured abstractions (aggregation, inclusion, etc.) and geometrical constraints encapsulates the general constructional know-how used by the craftsmen, in that particular period of time.

Despite the fact that the formalism used is a scene-graph, we consider the informational content of the applied operations as a conceptual 2D plane (fig. 14). It contains the sequential organization of the elements of construction processes such as the elimination of a wall, addition of a window, and the stratified existence within the graph of the transformed states: initial, intermediate, and final. The model shows a dynamic and heterogeneous stratification through time; the construction processes are not passed uniformly from one state to another and thus become, in the graph, a representation of the building's temporal changes. At a more abstract level, that model defines a probable design space in which possible combination sets could be identified, selected, and integrated as design aids in the design studio. Those heuristic operations, supported by the semantic relations, make it possible to generate and validate plausible

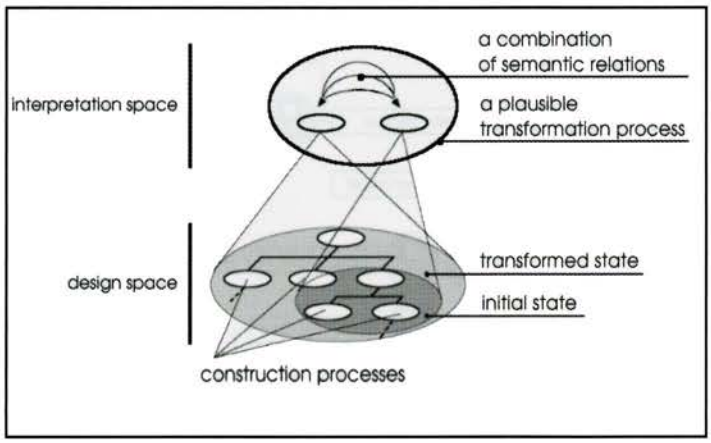


Fig. 14. The conceptual model of the universe of discourse.

interpretations of architectural transformation processes and thus complete our representation's universe of discourse.

In addition, the number of dimensions of both spaces actually depends on the chosen attributes of the architectural object(s) studied: for instance, we chose to represent the transformed states and sequences of the construction processes, so our design space is two-dimensional; when the semantic relations are added, we get a three-dimensional interpretation space (fig. 14).

The discussion of the heuristic dynamics of that interpretation space extends beyond the scope of this paper but it is important to note that, in this model, the scaling of both axes, i.e. the level of detail of the represented elements, influences the resulting interpretation of a given hypothesis. We could, for instance, add more intermediate states between those we have already mentioned, or more sequences of construction processes, and then re-examine the qualitative properties of the hypothesis. Consequently, that feedback would support the reusability of the represented knowledge, and, as a direct result of mapping the design space to the interpretation space, both the encapsulating generalization and the coding of the transformation history would facilitate the application of those blocks of code to the representation of other buildings in the neighbourhood, which were transformed by the widening of Des Fossés Street.

Conclusions and Perspectives

In this paper, we presented a knowledge-based representation of an architectural transformation in an urban environment. In the design studio, we chose a neighbourhood that shows a coherent pattern of transformational operations and discussed a case of operations as applied to a given building.

The framework of our model relied on the scene-graph formalism of VRML, widely used data-structure standards for 3D document exchange over the Web. It comprised two basic elements: a set of semantic relations describing the collected information, and a relational hierarchy organizing the acquired knowledge at an abstract level and defining architectural objects and transformation operations.

We considered the architectural transformation of the building as an amount of information that represents two consecutive states. That informational content, structured by the scene-graph in a set of construction sequences (elimination, replacement, extension, modification, addition, etc.), represents the range of possible know-how processes in a given craft and a set of constraints whose combinations lead to a space of possible designs. When coupled with a semantic hierarchy, the design space is extended into an interpretation space consisting of sets of plausible hypotheses. Consequently, the application of the scene-graph heuristics to the latter validates a chosen hypothesis and induces it in the conceptual phase of the project in the design studio.

The VRML scene-graph is a powerful tool to represent architectural knowledge and its mapping to the domain of object-oriented programming could support the development of educational approaches in design studios. That formalism supports the design studio by offering a computable means to (1) represent a design space in a modular and extensible model, (2) define its mapping to an abstract and cognitive interpretation space, and (3) construct the required interaction tools. There is much needed development to de done at the level of the core of VRML to make it more design-studio-friendly and our view remains dependent, in part, on the further technical development of that language and its extension beyond the simple 3D data exchange format.

Furthermore, we have identified the following perspectives for the representation of architectural transformations using the scene-graph:

– The development of a set of VRML meta-nodes based on the use of *Proto* and *ExternProto* as a basis for an interactive design system: capturing the user's interaction, representing a constraint-propagation mechanism to control this interaction, and modeling an inference mechanism to implement the resulting transformations on the scene-graph. Particularly, this system would include the definition of a set of user-oriented operators to manipulate the relational hierarchy. Practically, we are interested in developing a set of standard operators that would provide efficient manipulation of the scene-graph with respect to geometric constraint problems.

– The development of the semantic definitions: the actual set of semantic relations that we have defined can be extended to cover topological spatial relations such as inclusion, tangency, disjunction, containment, overlapping, and equivalence. Consequently and as a result of the extension of the semantic axis in the interpretation space and the development of the scene-graph heuristics, we would enlarge our universe of discourse and therefore enhance the qualitative representation of spatial information.

Notes

1. Wulf, William A., Mary Shaw, Paul N. Hilfinger, and Lawrence Flon, 1981, *Fundamental Structures of Computer Science*, Reading, MA, Addison-Wesley Publishing company, 621 p.

2. Noppen, Luc, 1996, *Patrimoine du Quartier Saint-Roch : L'identité architecturale, usages, formes et monuments*, Rapport pour la Division du design et du patrimoine, Centre de développement économique et urbain, Ville de Québec.

3. The Virtual Reality Modelling Language, International Standard ISO/IEC 14772-1:1997, [http://www.vrml.org/technicalinfo/specifications/vrml97/index.htm], date accessed: June 13, 2001.

4. Heck, Michael M., 1997, *VRML 2.0 for Open Inventor Programmers: A Technical White Paper*, Template Graphics Software (TGS).

5. Laframboise, Yves, 1975, *L'architecture traditionnelle au Québec : glossaire illustré de la maison aux 17e et 18e siècles*, Montréal, les Éditions de l'Homme, 319 p. (see p. 285).

6. Genesereth, Michael R., and Nils J. Nilsson, 1987, *Logical Foundations of Artificial Intelligence*, Los Altos, CA, Morgan Kaufmann, 405 p. (see p. 9).

7. Mitchell, William John, 1990, *The Logic of Architecture: Design, Computation, and Cognition*, Cambridge, MA, MIT Press, p. 22.

8. Genesereth and Nilsson : 11.

9. Liu, Li-min, and Michael Halper, 1999, « Incorporating Semantic Relationships into an Object-Oriented Database System », In Proceedings of the 32nd Hawaii International Conference on System Sciences, Wailea (HI), January 5-8, Institute of Electrical and Electronics Engineers (IEEE), Inc., p. 1.

10. Storey, Veda C., 1993, « Understanding Semantic Relationships », *Very Large Data Base Journal*, vol. 2, n° 4, p. 455-488. (see p. 456-457).

11. Wulf *et al.* : 101-106.

12. Simon, J.-C., 1984, *La reconnaissance des formes par algorithmes*, Paris, Masson, 251 p. (see p. 35).

13. Storey.

14. Storey.

15. Stefik, Mark, and, Daniel G Bobrow, 1986, « Object-Oriented Programming: Themes and Variations », *The AI Magazine*, vol. 6, n° 4, p. 40-62 (see pages 46-47).

16. Stefik and Bobrow : 47.

17. Liu and Halper : 2.

18. Lehman, Fritz, 1992, « Semantic Networks », *Computer Math Applications*, vol. 23, n° 2-5, p. 1-50 (see p. 20).

19. Jun, Han J., and John S. Gero, 1997, « Representation, Re-Representation and Emergence in Collaborative Computer-Aided Design », In Mary Lou Maher, John S. Gero, and Fay Sudweeks (eds.), 1997, *Formal Aspects of Collaborative Computer-Aided Design*, Sydney, Key Centre of Design Computing, University of Sydney, p. 303-320 (see p. 5).

20. Jun and Gero : 4.

21. Noppen, 1996.

22. Noppen, 1996.

23. The brick walls of those elevations were demolished and replaced with stone walls. However, a part of the original state had survived until the complete demolition and was visible on the southern elevation (fig. 11-b).