

## 3D URBAN GIS ON THE WEB: DATA STRUCTURING AND VISUALIZATION

Siyka Zlatanova\*, Michael Gruber\*\*  
ITC, 7500 AA P.O. Box 6, Enschede, the Netherlands\*  
ICG, TU, Muenzgrabenstr. 11, A 8010, Graz, Austria\*\*  
zlatanova@itc.nl, gruber@icg.tu-graz.ac.at

### ABSTRACT

The maintenance, visualization and query of 3D urban data has always been problematic due to the complexity of geometry, diversity of attribute information, large amounts of data, demanded comprehensive analysis and queries requiring new techniques for visualization and query. The approach presented here aims visualization and query via Internet and utilizes established standards and techniques to access and retrieve remote data: Virtual Reality Modeling Language (VRML) and Hypertext Markup Language (HTML) to visualize respectively 3D geometry and text information, Javascript and Common Gateway Interface (CGI) scripts to control the interaction and query of the three dimensional Geographic Information Systems (3D GIS) on the Web.

The paper focuses a couple of aspects related to the data base structure on the GIS server: clarification of data needed to create VRML and HTML documents, and data structuring according to an elaborated classification of the information stored per object, i.e. *attributes*, *connections* (relationships) and *functions* (behavior), considering its thematic and geometric aspect. A data structure capable to serve spatial analysis and to supply data for a VRML document creation with ability to introduce and control dynamics of objects, is presented. Issues related to fast navigation through the 3D model, i.e. a mechanism to build LOD without storage of extra information are discussed and some results of the usage of a R-tree structure to create Levels of Detail (LOD) are reported.

**KEYWORDS:** 3D GIS, VRML, WWW, GIS-server, R-tree, LOD.

### 1. INTRODUCTION

The growing interest in 3D urban data of wide range of local and remote users, on one hand, demands a 3D GIS providing extended techniques for data query, visualization and interaction with 3D GIS data, as well as, user-friendly, easy-to-use, standardized Graphics User Interface (GUI). Improved possibilities to access documents on remote hosts and establishment of standards (e.g. HTML to organize text, movie, image data and VRML to visualize and interact with 3D models, scripts to introduce and control dynamic), on another hand, have brought the opportunity to work out a strategy for a 3D GIS on the Web.

The coordination between all the standards and synchronization of the “query-response” process contributes to the establishment of specific requirements and influences data organization. The paper discusses these issues in the following sequence: first, a short description of the approach for query and interaction is presented; second, database requirements are discussed, third a description of the data structure is presented and, finally, some implementation results are reported.

### 2. SYSTEM ARCHITECTURE

The system architecture for visualization and query, presented here, is a typical client-server architecture: the persistent data are stored on the server(s) and the client accesses them from remote stations (see Figure 1). The

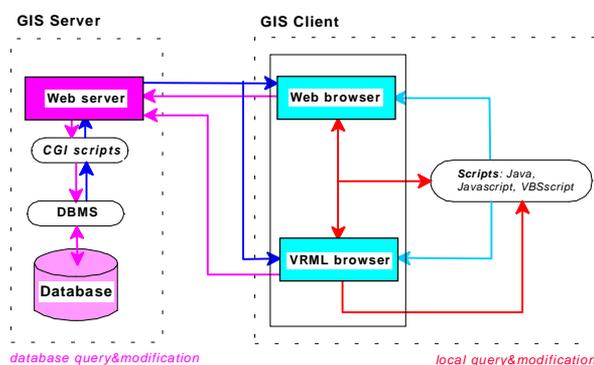
access to the data on the server is controlled by software, i.e. Web server and a number of CGI scripts. The client has to be supplied with HTML (Web) and Virtual Reality (VR) browsers at the client station to be able to observe and interact with text and graphics data. Text, 2D graphics, images, movies are organized in HTML files (documents) and 3D graphic data (models, worlds) are available in VRML files (documents). While the remote retrieval of HTML documents is already well established everyday practice on the Web, the use of VRML for modeling 3D models of real objects is still not so popular. However, some papers presented recently show increasing interest in the language, as well as its suitability to describe and interact with models of real objects (Lindenbeck, 1998; Schickler, 1997; Coors, 1998).

The system architecture is very similar to the one presented in (Lindenbeck, 1998), however, the goal is more broad. VRML in our approach provides not only means supporting visualization of 3D geometry, but mechanisms for interactive query of geometry. Compare to the approach presented in (Coors, 1998), our approach uses the GUI of the VR and HTML browsers to visualize and interact with data and CGI and Javascript scripts to manipulate and query data respectively on the server and the client station.

The process of client-server communication can be described shortly as follows: the client sends a request for information to the server using the VR or HTML

browsers, the Web server processes the request and returns the demanded data to the client station (see Figure 1). Depending on the request and the type of the data received, four fundamental phases can be distinguished: 1)identification, 2)query, 3)visualization and navigation, and 4)manipulation (see also Zlatanova et al, 1998).

The identification phase clarifies users rights and displays the starting HTML or VRML document. The information needed is specified in the query phase. The query can be formulated either in a HTML document by filing out a form or in the VRML document by an user action, e.g. "click" with the mouse on an object from the 3D model. The Web server sends back to the browser either a document existing on the server, or a document created on the fly after completed query from the database. The requested information, in form of HTML or VRML document, e.g. a table with text data or a 3D model of a particular neighborhood, is visualized in HTML or VR browsers at the client station. In the second phase, the user only retrieves information.



**Figure 1: Client - server 3D GIS**

The third phase refers already the manner of introducing changes in the data base or in the document, which is active on the screen at the client station. The modifications have two aspects: how to start and where to execute them. The alteration can be initiated in two ways (similarly to the data query): 1) inside the Web browser by filing out a sequence of forms or 2) inside the VR browser by interaction with the graphics. Concerning the location, two types of data modifications can be differentiated: locally on the client station and remotely in the database on the GIS server. The local changes are based on: 1)the VRML document capability to bring information up to the client station without its immediate visualization and 2)the execution of Java (Javascripts or VBS) for changing certain parameters in the 3D model. The data base changes can be completed by a sequence of HTML forms for introducing new values, CGI scripts to communicate with the Data Base Management System (DBMS) and database operations (see Figure 1).

The major advantages of the approach compare to the alternative standalone 3D GIS, can be summarized as

flexibility, extensibility and interoperability. The system can be easily tuned for a variety of outcomes. The documents created are standard documents that can be freely distributed across the Web. The use of VR and Web browsers, HTML fill-out forms familiar from many other applications on the Web, prevent the development of special GUI, which is a time consuming task. The approach followed does not require special hardware. In case of slow connection to Internet, the documents can be displayed after creating a copy at the client station.

### 3. REQUIREMENTS TO THE DATABASE

The analysis of the system architecture described above, reveals the need of data restructuring: 1) from the data structure used for data storage to the structure of VRML and HTML documents and 2) from VRML/HTML to the internal data structure of the VR and HTML browsers. The focus here is on the organization of data in the database. The second transformation is automatically done by the HTML and VR browsers and therefore is a responsibility of the developers of browsers.

Clearly, the data structure in the data base should be appropriate for both GIS analysis (thematic and spatial) and composition of documents (HTML and VRML). Since the visualization of 3D geometry is based on VRML, data to be visualized and the manner of their structuring are closely related to the concept of the VRML data structure (see Ames A.L, 1996). The documents created in most of the cases on the fly have to consume as less time as possible. Bearing in mind these two factors, we can summarize the following requirements for a data structure:

- **Information about appearance.** The VRML document contains two groups of data to describe appearance of objects on the screen: 1)data to define shape and position in the modeling space and 2)parameters to describe the physical appearance of the surfaces of the objects. The information about the first group of data is usually provided by set of ordered points and their coordinates. The information about surface of the objects can be expressed by color variations or textures to cover the geometry with image files. Since the complexity of shapes in urban areas is usually quite high, we need improved techniques to increase the realism and thus facilitate navigation through the model. The most successful solution is the use of real images for wrapping the geometry (see Gruber et al, 1997). If real images are not available, an artificial texture and materials or only color combinations have to be used. This is to say that the information about surfaces of objects (named *geometric attributes* later in the text) has to be available in the database.

- **Information about behavior.** The behavior of objects, here, is limited only to the geometry domain, i.e. how the object reacts during interaction with the VRML world. For example, what will happen if the user “click” with the mouse on the a building. One possible reaction is the begin of rotation around the building. Many GIS and CAD systems provide such dynamics without storage of any special parameters in the database, but the range of operations is rather limited. For example, a “click” on a building will cause always display of some thematic information. VRML, however, offers a technique to outline several behaviors per object. The language defines behavior of objects by 1)set of “sensors”, detecting user actions, time changes and changes in the status of objects (e.g. successful display of a HTML document), and 2) set of parameters altering objects and scene description (e.g. lights, viewing position). The mechanism works on the basis of prior information “what action activates what reaction”. Additional data are required but in return quite complicated behaviors can be designed.
- **Fast traversal of the data base.** In general, the approach to access the database has some waiting time for communication between the browsers at the client station and the Web server at the server station. Additional period of time is needed for the documents created on the fly. The time for transporting the document via Internet cannot be influenced, therefore a fast traversal of the database should be ensured, which refers the speed efficiency of the data structure.
- **Ability to create LOD.** The visualization using several LOD is well known computer graphics technique for navigation through large amounts of data. The most detailed description of the object (i.e. the most data for processing) is displayed on the screen only when the object is very near to the observer. Urban 3D textured models are quite large and the use of LOD for real-time navigation through VR worlds is almost compulsory. VRML supports LOD, i.e. permits several different predefined geometry descriptions to be included in the VRML document, as each LOD is in a separate VRML document. They are used by the VR browser as an indication for switching between more detailed and less detailed versions depending on the distance from the observer to the object. The technique requires, however, either the LOD to be stored in the database or easy creation of LOD on the basis of existing data to be ensured.

The requirements listed above concern only the visualization process with respect to the chosen approach for visualization and interaction and extend the

traditional GIS requirements for maintenance of topology and thematic information.

#### 4. CONCEPTUAL SCHEMA

This paper is limited to data structuring in geometry domain, therefore issues related to thematic domain will be discussed only for completion of definitions and expressions.

##### 4.1 Spatial object

The fundament of the VRML concept, as well as, the concept for interaction with VR worlds, is the object oriented approach. However, what is an object? Which parameters give a full description of an object? How can we structure the information per object? Since these basic questions should be answered prior the database organization, a clarification of the components contributing to the complete description of a spatial object will be presented in the following chapter.

As a starting point for the spatial object definition we use the Coad’s broad understanding (Norman, 1996) of an object, i.e. “the object can be anything: feature, action, process, which is of interest for the user”. The answers to the questions “what the object knows about itself”, “who the object knows” and “what the object does” (named *object responsibilities*) and time-related component named *scenario* aim to a complete characterization of the object. In other words, an object **O** can be everything (feature, process, action, etc.) that can be represented by two components **OR** (object responsibilities) and **S** (scenario):

##### **O (OR, S)**

We will use terms *attributes*, *relationships* and *behavior*, instead of the questions part of **OR** in the Coad’s definition. *Attributes (A)* comprises all the information, which can be collected answering to the question “what the object knows about itself”, *relationships (R)* and *behavior (B)* are related respectively to the question “who the object knows” and “what the object does”. Thus we can write that the object responsibilities **OR** have three components:

##### **OR (A,R,B)**

A substitution of the **OR** components in the notations for an object **O** we obtain the full set of components describing an object, i.e. *attributes*, *relationships*, *behavior* and *scenario* :

##### **O(A,R,B,S)**

The term *scenario* stands for all the information, which helps to follow, record and understand the evolution of the object in long term period of time.

A spatial object easily can be described by the four components. For example, a building has:

*attributes*: hotel, made of bricks, somewhere in the middle of the town (coordinates of the building are available);

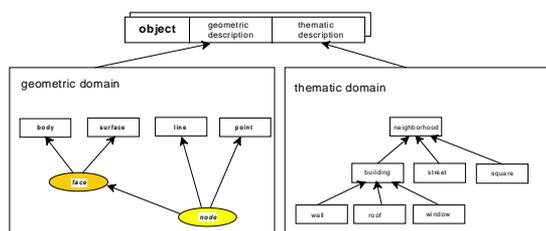
*relationships*: attached to the building of the theater, part of chain of hotels;

*behavior* : possible walk through the hotel and sightseeing from the roof terrace, 10% of the profit go somewhere;

*scenario*: building is reconstructed four times, last used as a hospital.

It clearly can be seen that some of the descriptions refer geometric domain (e.g. “made of bricks”) and other are pure thematic (e.g. “10% of the profit go somewhere”). Therefore, we will further elaborate on *geometric (GD)* and *thematic (TD)* domain of a spatial object (see also Molenaar, 1992):

## O(GD, TD)



**Figure 2: Geometric and thematic description of an object**

We distinguish *attributes*, *relationships*, *behavior* and *scenario* of spatial objects in thematic and geometric domain:

**GD (GA, GR, GB, GS)**

**TD (TA, TR, TB, TS)**

where **GA, GR, GB, GS** - *geometric appearance, geometric relationships, geometric behavior, geometric scenario*; **TA, TR, TB, TS** - *thematic attributes, thematic relationships, thematic behavior, thematic scenario*

The substitution in the notation for a spatial object will give us the following:

**O((GA, GR, GB, GS), (TA, TR, TB, TS))**

Despite the thematic and geometric specialization in the components, it is still possible to maintain different in nature objects (people, buildings, documents, etc.). The components are not compulsory. If the geometric components does not exist the object can be maintained only according to its thematic description. Similarly, not

all the components within one domain are obligatory. For example, geometry domain may be represented only by **GA** and **GR** or even only by **GA**. In general, the information which is maintained in nowadays GIS’s corresponds to the information containing in **GA**, **GR** and **TA** components.

The establishment of clear differentiation between thematic and geometric information facilitates the process of information structuring and permits creation of separate thematic and geometric hierarchies (see Figure 2). In case of switching from one geometry representation (e.g. boundary representation) to another (e.g. voxel representation) only the necessary components (e.g. **GA** and **GR**) have to be replaced.

Different associations between thematic and semantic hierarchies permits a concept for a multi-resolution description per object to be developed. For example, an object named building can be represented by a “box” (i.e. **GA<sub>1</sub>**) in **GD** and can have the properties of an administrative building as **TA** in **TD**. Another application, however, can require the same building with the same **TA** to be represented as a point (i.e. **GA<sub>2</sub>**).

### 4.1.1 Geometric domain (GD)

The components in thematic domain (**TD**), will be not studied further because 1) their high dependence on the purposes of the information system and 2) a variety of approaches and methods to structure thematic information (Norman, 1996). Therefore, we will keep the notations of an object, here, as follows:

**O((GA, GR, GB, GS), TD)**

#### 4.1.1.1 Geometric appearance (GA)

**GA** is notation for *geometric appearance*, not *geometric attributes*. The use of other name is done due to the more complex meaning of *attributes* in geometry domain. Information about shape, position, size, color, texture, etc. has to be structured in this component. This data are related to the visual appearance of the object in 3D space. Shape, size and position of the objects are dependent on the manner of geometry description chosen, (e.g. boundary representation, constructive solid geometry) and the level of abstraction applied. Variations can be quite significant. Color, texture, material are determined by some physical properties (e.g. material used for covering roofs) of real objects and are not influenced by the geometry representation. The roof of the building is red regardless the geometric representation, i.e. a cone or a set of triangles. Therefore we introduce two new components: *geometric description (GDsc)* and *geometric attributes (GAtt)* as a part of **GA**:

**GA(GDsc,GAtt)**

=> **O(((GDsc, GAtt), GR, GB, GS), TD)**

Despite the three dimensions of every object, modeling process still requires certain abstractions of real objects to be build. The historical human experience with maps and 3D CAD models has contributed to establishment of four abstraction types of objects, i.e. points, lines, surfaces and solids. We will use the terms *point, line, surface and body* and will give them the a common notation *geometric objects (GO)*. The next necessary step is to distinguish between *geometric objects* and *constructive objects (CnsO)*. *Geometric objects* are elementary *nD* objects ( $n = 0,1,2,3$ ), which can be associated with thematic meaning, while *constructive objects* are used to compose *geometric objects*. For example, a house can be build of many *cubes* with different size and position in the space. The *cube* is a **CnsO** and the construction of *cubes* is a **GO**. The *geometric description*, in fact, is a function of *constructive elements*:

**GDsc(GO(CnsO))**

Then the *geometric appearance* is represented by two components *geometric description* and *geometric attributes*, where the *geometric description* is expressed by *geometric objects (GO)*, which are function of *constructive objects (CnsO)* i.e.

**GA(GO(CnsO),GAtt)**

and the object notation is extended with the components containing more detailed information about **GDsc**:

**O(((GO(CnsO),GAtt), GR, GB, GS), TD)**

#### 4.1.1.2 Geometric relationships (GR)

The second component cares about relationships, which in geometry domain will refer to spatial relationships such as containment (*building inside parcel*), touch (connected buildings, adjacent parcels), etc. The component is not compulsory for visualization process but facilitates spatial analysis. The way of representing spatial relationships is again related to the method of description. If the **GDsc** component does not provide all the needed spatial relationships, some of them can be explicitly formulated. However, **GR** and **GD** are very much dependent and will not be discussed in details.

#### 4.1.1.3 Geometric behavior (GB)

The third component, denoted *geometric behavior*, contains information about the permitted operations on the object during navigation and editing. In the light of the VRML concept, we distinguish the following types of *behavior*:

- **Operations on geometry (OG)**. The type refers permitted operation on an object such as: 1) *deleting (OD)* an existing object or some of its components, 2) *updating (OU)* some values of components of an existing object and 3) *adding (OA)* a new object or a new component of an object. Operations on geometry can be presented as a set of three components **OG (OD, OU, OA)**. Further, we specify which particular components are accessible by the user for modification. For example, we can forbid any changes in the components **GDsc** and allow changes only in **GAtt**.

The control of the operations on objects can be successfully used to protect the information on the GIS server. Since the tendency of our approach is to provide a broad range of users with access to the information, a strong security system against mistakes and unscrupulous actions has to be developed. Protection of data can be build up on two levels: server and database. The server level controls and restricts the user rights to modify the data in general. The database level protects a particular object from a particular action, e.g. a building cannot be deleted by any user via Internet.

- **Reactions of objects on events (GE)**. This type of *behavior* aims to the establishment of a strategy for describing the user interactions with the object and the possible reactions. In this context, we define two components: *initial event (EI)* and a corresponding *reaction (ER)* of an object. *Initial event*, i.e. the action that can be detected by the system and processes, which are supported by VRML are: 1) user action i.e. click with the mouse, drag and drop with the mouse, pass over object with the mouse; 2) absolute and relative time (i.e. some event can be initiated at a predefined in VRML document moment, counted by an internal clock) and 3) events, caused by other applications (e.g. a display of a document, successful connection to the server, which are detectable by a special *field values* in the syntax of VRML. The reaction can be either executing of a an existing on the GIS server HTML or VRML document, or running a script file (CGI, Java, etc.), or starting a predefined action (animation, rotation, shifting of object), which can be included the current VRML document. Last case requires the **ER** component to be refined for the parameters necessary to describe the action. For example, if we want to define: “after two clicks with the mouse start an animation showing rotating building. Some parameters, e.g. center of rotation can be computed from the data in the **GDsc** component, but other, e.g. speed of rotation might be stored. The **GE** component is represented as **GE(EI,ER)**

- **Reactions on interactions with other objects (GI).** This type of *behavior* is devoted to interaction between object inside the model. For example, if we have an object car and we start movement with the car trough the town, we can specify what will happen if the car touches one or another building. Note, we can specify different reaction: the car could crash or pass trough the object of interaction. To make possible this kind of behavior, we define three components: *initiator*, i.e. the object caused the interaction (**IO**), *initial event* (**IE**) and *reaction* (**IR**). Then the short notation for this type of *behavior* can be written as **GI(IO,IE,IR)**.
- **Degree of immersion (GM).** Last possible *behavior* is with respect to specification of a detailed investigation of an object, e.g. entering a building. The information is quite relevant for composite objects. For example, suppose a building is a aggregation of rooms, walls, stairs, etc. The information about the interior of the building is not necessary for a simple “walk trough the town”, therefore a VRML document only with the walls of the building can be created. If the user wants to enter the building, a new VRML document should be created and submit to the client station. A possible way to display the interior of buildings is the usage panoramic images and appropriate viewers. Useful information ordering the files with panoramic images can be organized in the **GM** component.

Finally, the complete set with all the components describing and structuring *behavior* of objects s:

**GB(OG, GE, GI, GM)**  
**=>GB((OD, OU, OA), (EI, ER), (IO, IE, IR), GM)**

In fact, the classification of *behavior*, listed above, can be realized in the VRML document applying different mechanisms, which can lead to combination of some parameters on implementation level.

The last component of the **GD**, i.e. *geometric scenario* **GS** pursues maintenance of information about changes in long periods of time. For example, appropriate data can give an idea what are the changes in the shape of the building for a period of ten years, or what are changes in the vegetation in a town in five years, or even what is the pollution propagation in an hour. However, the **GS** is far beyond the scope of the paper and will not be discussed in details.

At the end, the notation for an object including the elaboration in *geometric domain* (**GD**) is presented as follows:

**O(((GO(CnsO),GAtt), GR, ((OD, OU, OA), (EI, ER), (IO, IE, IR), GM), GS), TD)**

#### 4.1.2 Composite objects

The definition and structuring of composite objects always have been related to a number of difficulties, e.g. the composite object does not have the properties of the creating objects, the decomposition into composing pieces sometimes is not possible, need of well defined system for inheritance of parameters. Despite the problems, the composite objects are necessary for:

- dynamic modeling, e.g. to move composites relatively to one other
- increasing storage economy by references to already known objects
- easy update propagation, i.e. modification of a “parent” object will be propagated to “children” object

Two basic techniques have been applied in computer graphics for creating composites of 3D cells (solids): spatial set operations (union, intersection and difference) and joining pieces along their boundaries. The first technique is more suitable for 3D objects represented as solids, while the second is more often used for surface representations. An advantage of the first method is easy way of decomposition and disadvantage impossibility to model separate faces. Second method usually does not support back partition into composing object. A variety of mixed structures are implemented in order to benefit from some advantages and avoid drawbacks. The complexity of the problem increases when a spatial object is considered due to the thematic and geometric component.

VRML allows grouping of objects as the principles are aggregation (geometry, appearance and behavior of objects), inheritance (transformations) and encapsulation.

A *composite object* (**CO**) will be defined here as a set of *objects* (**O<sub>i</sub>**) and *composing rules* (**Ru<sub>i</sub>**) for composition and can have its own components in *geometric domain* (**GD**), *thematic domain* (**TD**), i.e.

**CO(O<sub>i</sub>, Ru<sub>i</sub>, TD, GD)**

The *composing rules* are per object and refer each component of the object, i.e. *attributes*, *relationships*, *behavior* and *scenario*:

**Ru(RuA, RuR, RuB, RuS)**

where, **RuA** is rules for composing attributes, **RuR** - rules for composing relationships, **RuB** - rules for composing behavior and **RuS** - rules for composing scenario.

Clearly, the *composition rules* are different for *geometric* and *thematic domain*. For example, a composite building can be organized as aggregation of several small buildings in geometric domain, while classification principal can be applied in thematic domain. Therefore the **Ru** components per object has the notation:

### **Ru(RuGD, RuTD)**

where **RuGD** and **RuTD** are rules for composition in geometric and thematic domain.

=>**Ru((RuGA, RuGR, RuGB, RuGS), (RuTA, RuTR, RuTB, RuTS))**

We can write the notation about composite object as:

**CO(O<sub>i</sub>, Ru<sub>i</sub>, GD, TD) => CO(O<sub>i</sub>, (RuGD<sub>i</sub>, RuTD<sub>i</sub>), GD, TD) =>CO (O<sub>i</sub>, ((RuGA<sub>i</sub>, RuGR<sub>i</sub>, RuGB<sub>i</sub>, RuGS<sub>i</sub>), (RuTA<sub>i</sub>, RuTR<sub>i</sub>, RuTB<sub>i</sub>, RuTS<sub>i</sub>)), GD, TD)**

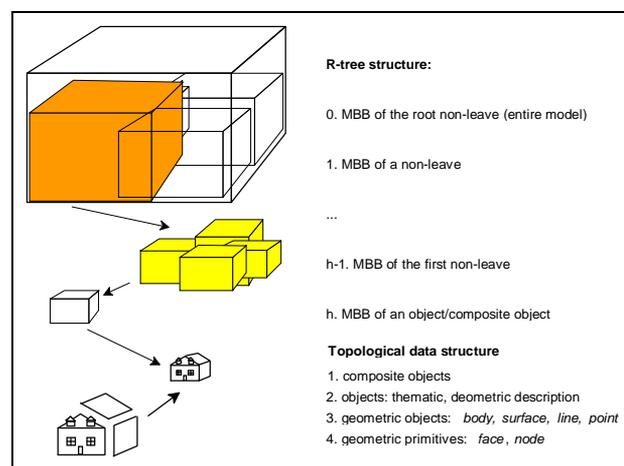
Some simplification of the notation can be achieved if a the rules for composition are unified and the same rules are applied for all the components in a certain domain. For example, the **RuGR** component from *geometric domain (GD)* can be dropped off the notations because the in most of the cases the spatial relationships are related to the **GDsc**, defined on an object level.

### 4.1.3 Spatial indexing: 3D R-tree

Presented structuring of the information per object in the previous sections clarifies the data needed to create the VRML document and facilitates their organization. Recall from chapter 3 reveals that fast access of database and organization of LOD are still not ensured. One way to achieve fast traversal of the database and respectively short time for creation of VRML documents on the fly, is a smart organization of data in **GDsc** component, i.e. utilization of structures implemented in VRGIS for real-time navigation. Most of them, however, aim speeding up of the visualization process rather than performance of spatial analysis. Therefore, we prefer a geometric representation maintaining topology, which will facilitate spatial analysis and apply additional techniques to reduce the traversal time (see Figure 3). Such a technique is the implementation of an indexing schema. We concentrate on spatial indexing schemas (e.g. Quad tree, R-tree) due to aimed ability to complete spatial and network analysis (e.g. “show the way to the center of the town”), not only retrieval of data (e.g. thematic attributes) about given object, which can be realized by linear indexing. Some properties of R-trees, e.g. arbitrary sell boundaries, multi-dimensional search space, make them some of the best structures for spatial searching. The R-tree intended for spatial indexing is a slight

modification of the classical R-tree presented in (Guttman, 1984).

The major properties of the classical R-tree can be summarized as follows: the R-tree is a collection of tuples as each tuple has unique identifier; leaves contain a tuple of the form (MBR, O<sub>i</sub>), where MBR is the minimum bounding rectangle of an object O<sub>i</sub>. Non-leaves are presented by another tuple (MBR, R<sub>i</sub>), where MBR is minimal covering rectangular in the lower entries. The R-tree is a balanced tree with maximum height  $\log_m N - 1$ , where  $N$  is the number of the objects for indexing. Reported implementations show that the optimal number of entries  $i$  per node is between 3 and 4.



**Figure 3: 3D R-tree indexing**

The R-tree organization of objects aims not only speeding up the operations with the database but supplying data for organization of LOD on the fly. Therefore, we modify the bounding rectangles to bounding 3D boxes. The properties of the classical R-tree are preserved as MBR is renamed to minimum bounding box (**MBB**) (see Figure 3).

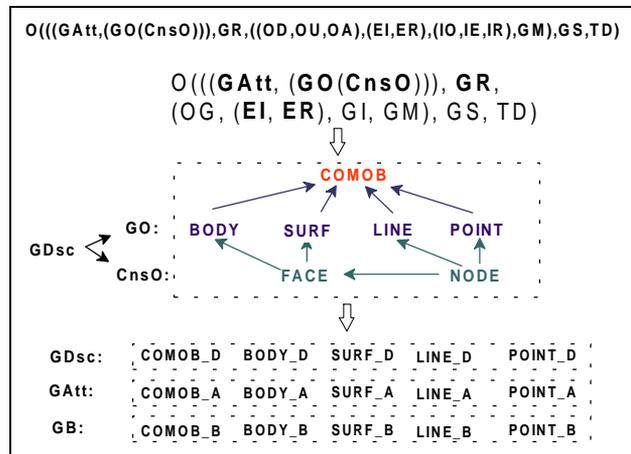
The VRML documents with the LOD has to be created separately from the basic file with the full-detail data but simultaneously within the time for data query. The back/forward switch operation between VRML documents with different LOD is handled automatically by the browser during the navigation through the model. In general, two approaches to maintain LOD can be followed: preceded creation and storage of LOD in the data base (Schickler, 1997) or dynamic creation on the basic of existing data. First approach gives the freedom to design LOD, i.e. to select which components of **GDsc** will be used for a particular LOD. In fact, the LOD are different *geometric descriptions* per object. Second approach leads to more coarse LOD but needs less data for storage.

In our approach for visualization, we use the leaves of the R-tree, i.e. the component **MBB** to create LOD on the fly as follows:

- LOD0: the **MBB** of the root of the R-tree, i.e. one box
- LOD1: the **MBB** of the first non-leaf level
- LOD2: all the **MBB** of leaves: bounding boxes per object
- LOD3: full geometry resolution without photo texture.
- LOD4: full geometry resolution with photo texture

## 5. IMPLEMENTATION ISSUES

The data structure presented here is not complete implementation of all the components of the object defined above. We concentrate mainly on **GA**, **GR** and **GB**, assuming that they represent the most important information to create VRML documents and have crucial impact on the successful communication between client and server (see Figure 4).



**Figure 4: Implemented components: GDsc, GAtt, GB**

Chapter 4 was devoted on a schema for structuring of information per object, however, the manner of geometric description of the objects was not discussed. We will As it was stated above, **GDsc** is influenced by a number of factors: the purpose of the application (e.g. environmental analysis or town planning), the method for data collection, the rendering engine used for visualization, etc. Boundary representations (B-reps) seem to be the best suited description for urban modeling due to 1)the prevalent attention to the surfaces of the objects, and 2) mostly surface measurements to build geometry. On another hand most of the rendering engines (VR browsers in our approach) are based on B-reps. Some recent developments in 3D reconstruction of man-made objects draws the attention up to the constructive solid geometry (CSG) structures, which in practice is realizable in VRML. We consider, however, the storage of real measurements a crucial requirement for our data structure. Since it is more difficult to be realized in CSG structures, we concentrate on B-reps description without dipper elaboration.

The **CnsO** in B-reps are points, lines (arcs) and faces, which are used in different combinations in CAD and GIS structures (Baer et al, 1979). The model presented here is based on existence of two of these **CnsO**, i.e. points named *nodes* and *faces*. This is to say that a *surface* and *body* object will be described as a set of *faces*, while *line* object will be a set of *nodes* and *point* object will be the *node* containing the coordinates (see Figure 2 and 5).

The most significant spatial relationships for the VRML document among *geometric objects* and their *composing objects*, are the relationships *object-face* and *face-node*. The query for creation of VRML documents starts always from the object level and ends at the lowest *constructive object* level, i.e. *nodes*. This requires an *object-face-node* traverse of the data base, which is assured by storage of boundary relationships, e.g. *face* is a list of *nodes*, *body* is a list of *faces*.

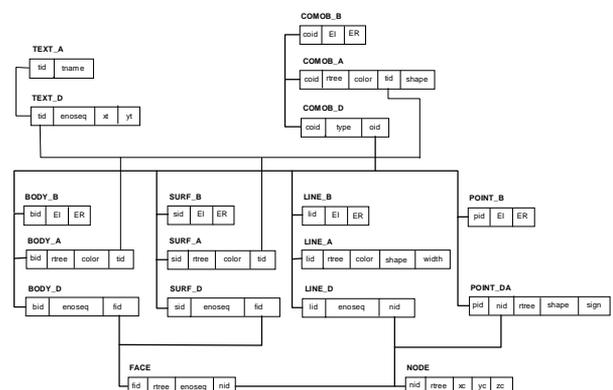
In order to simplify the *composing rules* to create *composite objects*, we establish the following rules for composition of objects in geometry domain:

**geometric objects (GO(CnsO)):** The major principle is aggregation, i.e. the *composite object* is a child, containing all the primitives of parent objects.

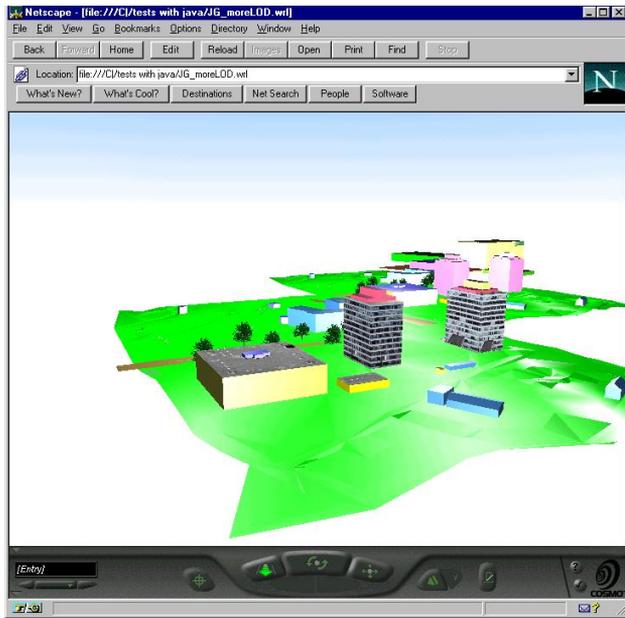
**geometric attributes (GA):** The rules are aggregation and inheritance. If the *composite object* has its own geometric attributes they are dominant, e.g. a *surface*, presented as a composite object of a street, canal and parcel, can have a *geometric attribute* texture covering the three parent objects.

**behavior (GB):** Maintenance of behavior of composite objects is similar to the geometric attributes. In general, two distinct cases are possible: 1)the child have an individual behavior, which is dominant for the *composite object* and 2)the child does not have behavior, i.e. the parent behavior can be activated individually.

Bearing in mind the assumptions about *composing rules*, the *composite object* can be expressed as a list of composing objects.



**Figure 5: Relational data structure**



**Figure 6: LOD: full resolution, geometry without texture, MBB of a non-leave of the R-tree.**

The data are organized in a relational data model as each component (**GDsc**, **GAtt**, **GB**) is represented by one relational table (e.g. SURF\_D, SURF\_A, SURF\_B) (see Figure 4 and 5). Thus each **GO** (*body*, *surface*, *line*, *point* and *composite object*) has three relational tables. An example of the information stored in the relational tables can be seen below:

**SURF\_D**

sid	enoseq	fid
1	1	-2
1	2	+1
1	3	+3

sid - surface identifier; enoseq - sequences of faces in a surfaces; fid - face identifier where the sign +/- is indication for the orientation of the face;

**SURF\_A**

sid	rtree	color	tid
1	245	234	2
2	238	123	2
3	156	234	3

sid - surface identifier; rtree- the path in the r-tree; color - RGB color of the surface; tid - texture identifier for wrapping or mapping one texture on a surface;

**SURF\_B**

sid	EI	ER
1	onckl	"pp.wrl"
1	onpas	"ch.java"
2	onckl	"rp.wrl"

sid - surface identifier; EI - event initiator; ER - event response.

The R-tree leaves and non-leaves are organized in relational tables containing information about **MBB** (minX, minY, minZ, maxX, maxY and maxZ coordinates of a R-tree box ) and the identifiers of the three children leaves (non-leaves). The number of

entries, i.e. three, was chosen among experimented 2,3,4 and 5 entries per non-leave. The aim was to achieve such groups of objects in the height  $h-1$ , i.e. the first non-leave level which can be used to create one LOD. This means that the **MBB** has to give an idea about the shape of the three grouped objects. As an experimented criterion for grouping was used: the minimal oblique distance, the minimal horizontal distance, and the min-max angle, between weight centers of the objects. The best results were achieved with criterion min distance and min-max angle between mass centers of the objects. An additional column with the position in the r-tree was included in the *attribute* tables of each **GO** and **CO** and **CnsO** in order to facilitate the traversal of the FACE and NODE tables. Note, the FACE and NODE table contain all the *faces* and *nodes* in one 3D model.

The method for creating LOD for visualization is expected to perform satisfactory results for buildings (see Figure 6), however, large surface objects will cause visualization artifacts. Suppose the DTM is one object, its bounding box will cover the entire model and in case of rough relief will hide large parts of the model. Apparently either these LOD should not be applied to such surfaces, or the surfaces should be subdivided further into smaller parts.

**6. SUMMARY AND FURTHER RESEARCH**

We presented one possible solution for a 3D GIS on the Web using HTML and VRML documents for visualization, navigation, query and modification of data. Specific for the system architecture database requirements were clarified and systematized, and a concept for structuring the information per object was introduced. The data structure maintains information describing behavior of objects allowing variety of even-driven operation to be formulated. From the user point of view this means ability to query objects in the VRML world for diverse information (text, movie, image, VRML), modify objects and their components.

The data structure derived is mapped on a relational data model. The implementation components of the system are: APACHE server as a Web server, Netscape as a HTML browser, COSMO Player as a VRML browser, MySQL as a DBMS. The programming languages are Perl and C++ for creating CGI scripts and Javascript for manipulations of the model at the client station. The full functionality of the system is not reached yet. The query-response process is limited to sending back to the client station HTML and VRML documents existing on the server. The next step is the creation of documents on the fly after completed query from the database.

## References

1. Ames, A.L., 1996, VRML 2.0 Sourcebook, John Wiley&Sons, Inc., New York, USA, 654 p.
2. Baer, A. C. Eastman and M. Henrion, 1979, Geometric Modeling: a Survey Computer Aided Design, Vol. 11, No. 5, pp. 253-272
3. Coors, V. and V. Jung, 1998, Using VRML as an Interface to the 3D Data Warehouse, Proceedings of VRML'98
4. Gruber, M., M. Kofler and F. Leberl, 1997, Managing large 3D urban data base contents supporting phototexture and levels of detail, Proceedings of the Ascona Workshop'97, AEMMOASI, Birkhäuser Verlag, Basel, pp. 377-386
5. Guttman, A., 1984, R-trees: A dynamic Index Structure for Spatial Searching, ACM SIGMOD 13, pp. 47-57
6. Lindenbeck, Ch. and H. Ulmer, 1998, Geology meets virtual reality: VRML visualization server applications, WSCG'98, Vol. III, Plzen, Czech Republic, pp. 402-408
7. Molenaar, M., 1992, A Topology for 3D Vector Maps, ITC Journal, 1992-1, pp. 25-33
8. Norman, R.J., 1996, Object-Oriented Systems Analysis and Design, Prentice Hall International, Inc., New York, 431 p.
9. Schickler, W., 1997, A Virtual reality model of a major international airport, Proceedings of the Ascona Workshop'97, AEMMOASI, Birkhäuser Verlag, Basel, pp. 367-376
10. Zlatanova, S. and K. Tempfli, 1998, Data structuring and visualization of 3D urban data, presented at the AGILE conference, Enschede, the Netherlands