VRML For 3D GIS

Siyka Zlatanova[1]
ITC, Enschede, The Netherland

Abstract

The three dimensional Geographic Information System (3D GIS) is an area of intensive investigations. A 3D GIS should be a system capable of maintaining and analyzing 3D spatial and thematic properties of real geographic objects. Most of the research efforts concentrate on stand-alone solutions. However, recent innovative technologies developed on the World Wide Web allow client-server alternatives of 3D GIS to be studied.

The paper presents an approach to query, modify and interact with remote spatial databases, which aiming at providing functional capabilities of a 3D GIS. The approach is based on dynamically created VRML documents to portray 3D graphics (spatial data), HTML documents to formulate SQL queries, CGI scripts to access the database. The techniques proposed are examined by a prototype system. Working examples of spatial queries are described in detail and performance tests are discussed at the end.

Keywords: client-server, CGI, spatial queries, SQL, HTML

1 Introduction

The increased complexity of tasks in many applications seeks for an integration of 3D spatial and thematic data and mutual relationships. Existing systems either fall short to deal with 3D geometry (2D GIS) or lack extended spatial and thematic analysis (CAD). For example, queries such as “show which buildings have a common wall”, ”show which pipe goes through this building”, “show all the offices, which are on the second floor” still can not be accomplished by any of the commercially available systems. In this respect, many authors consider a 3D GIS, which maintains 3D topology, spatial and thematic information, the successful solution (see [11],[15]).

One of the challenges for the 3D GIS development is remote access across the Internet. The issue has three central aspects 1) organization of data on the server, 2) means to formulate queries and 2) visualization of the resulting information (basically 3D graphics and text) on the client station.

The Web has already shown a great potential in improving accessibility to spatial information hosted in different computer systems over the Internet. The commonly implemented approach is the provision of Internet access to existing systems (CAD or 2D GIS), e.g. Intergraph with GeoMedia, Autodesk with MapGuide, ESRI with ArcView Internet Map Server. However, the spatial information supplied is 2D (i.e. raster or vector maps) and the scope of queries is limited to the ones predefined by the vendor (see also [8]).

Until recently, 3D models were not directly accessible on the Web. The Virtual Reality Modeling Language (VRML) was designed to fill the gap. Since December 1997 when it was approved as a Web standard, it has gradually gained popularity for visualization of urban models (see [3],[10],[13],[14]), geological structures (see [9]), historical or architecturally significant building constructions (see [6],[12]), tourist information (see [4]) etc. Most of the VRML models created so far are pre-designed files stored on the server. Dynamically created VRML documents are reported only by Coors et al and Gahedan (see [7],[9]). The language, however, has the potential to describe the behavior of objects, provide links to other documents on the Web, represent interrelations that can be used to retrieve and visualize 3D spatial information and thus serve as an interface to 3D GIS..

The approach of a 3D GIS on the Web presented here, relies on a Database Management System (DBMS). The data are organized according to a conceptual schema, which integrates 3D topology, spatial and thematic data. Practically, the database supplies information for both 3D spatial analysis and visualization queries completed with the help of SQL statements. The access to the database is controlled by CGI scripts. Dynamically created VRML and HTML documents provide the Graphic User Interface (GUI) to complete queries, visualize results and explore 3D models. The VRML documents contain extended information for A similar approach with direct access to an integrated database via a Spatial Data Manager is presented in [2]. However, the space is restricted to 2D and the user interacts with the map through Java applets.

The paper is organized in the following order: first a short overview of the VRML syntax is given, second, a detailed description of the system architecture is presented, third the most significant requirements for the conceptual schema are mentioned and finally, several examples of 3D spatial queries are discussed. The performance results obtained of two data sets form urban areas are given to qualify and illustrate the approach presented.

[1]zlatanova@itc.nl

2 VRML and VR browsers

VRML is a high level object-oriented language for the description of scenes and the behavior of objects. Initially based on the SGI Open Inventor file format for exchange of data, the language has passed through several stages, i.e. VRML 1.0, VRML 2.0 VRML97, before its endorsement as a Web standard. The syntax of VRML is based on objects (*nodes)* with parameters (*fields*). A number of *nodes* are responsible for the design of the *scene*: description of geometry (regular and irregular shapes, grids, text), illumination of the model (directional, spot, point and ambient lights), materials and textures (draping and mapping of JPEG, GIF, PNG image file formats). Combinations of another *nodes*, i.e. *sensors*, *routes* and *interpolators* introduce dynamics. *Sensors* detect viewer actions (e.g. mouse move, click, drag), time changes and viewer positions (visibility, proximity, collision). *Routes* direct captured events to *interpolators* to alter some *fields* (color, position, orientation, scale). While appropriate for direct animations, the mechanism is insufficient for descriptions of complex actions, e.g. the control of sequential clicks with the mouse on an object. In case of complicated movements and manipulations, the *script* node referring to Java applets and JavaScripts, may be employed. The *proto* node supplies the user with a tool to design his/her own *sensors* and *interpolators*. Common Gateway Interface (CGI) scripts embedded in the body of the VRML document allow establishment of connections to any application on the server. All the VRML *nodes* can be aggregated in various complex hierarchical composites and altered together. More details about the syntax of VRML can be found in [5].

The scene designed according to VRML is stored in an ASCII file. Specific visualization software, i.e. Virtual Reality (VR) browser is necessary to display data on the screen. The role of VRML document and VR browsers is different. The VRML document supplies the parameters for scene design and the dynamics of objects while the VR browser takes care of scene rendering and the interface to navigate through and interact with the model. Initially, the basic function of the VR browser, besides visualization, was only real time navigation through the model, i.e. provision of virtual reality techniques: *examine, fly-over, walk-trough, pan, zoom*. The second edition of VRML granted the VR browser with new responsibilities, i.e. detection of user interactions with objects if they were described in the VRML file. Plenty of freeware and trial versions of VR browsers can be downloaded from the site of the 3D Web consortium (see [1]).

The potential of VRML and VR browsers for 3D modeling is still underestimated. The understanding about the couple VRML-VR browser is often that it was a system for visualizing of 3D graphics on the Web allowing real-time exploration. This impression is created mostly by CAD and GIS vendors, which offer export of their models in VRML. The models created are static 3D worlds lacking dynamics and point-and-click capabilities. In the next sections, it is demonstrated that VRML and VR browsers can serve the more sophisticated tasks that are needed for a 3D GIS, i.e. query of the objects of the model to obtain thematic or spatial information.

3 The approach

The client-server architecture presented here is based on the CGI mechanism to access remote data. The main components of the system are a Web browser with a VRML plug-in on the client site, a Web server and DBMS on the server site (see Figure 1). A set of CGI scripts acts as an intermediate communication unit between the client and the DBMS.

The entire process of information extraction can be categorized into several groups, i.e. identification, query, data modification and exploration.

User identification and database selection. At the first stage, users are given options to chose a model (e.g. a town or even a neighborhood might be in a separate database) and the operations desired (query and visualization or updating). The database can be selected inside an initial HTML document (by filling out a form or selecting an area of interest in a 2D map) or a VRML document (pointing on a 3D map). Among the variety of approaches to specify the scope of operations, the most convenient one is the provision of the often used operations (e.g. query and visualize) for all the users. Thus any user can freely send queries and receive responses without restrictions until an attempt to change data in the database.

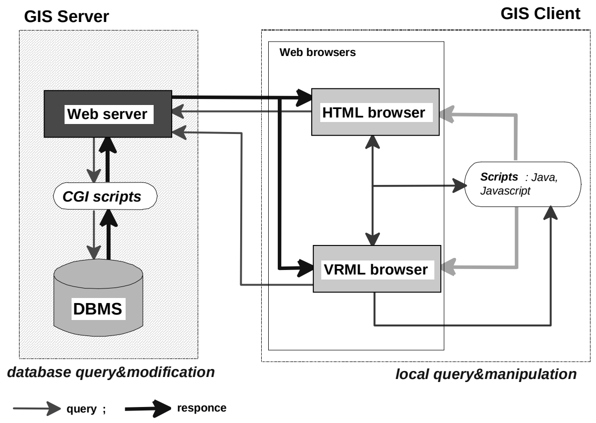


Figure 1: System architecture

Query. The request for information about an object (e.g. a building, an owner, a parcel, a street) is broken down into two steps: object identification and information specification. For example, the simple query “who is the owner of this building” will require means to 1) point to and select the building and 2) select the attribute “owner” among all the building characteristics. The first step can be formulated either in a VRML or an HTML document. In an HTML document, the object can be specified by typing its identifier (ID). Most likely, the user is not always aware of IDs in advance. A more attractive alternative is the recognition of the object in a VRML document. The VRML file in this case has to be created such as a link between the ID in the database and the object on the screen (i.e. the VRML *node* in the VRML document). The user points the object of interest, observes its ID and composes the query.

The identification of an object activates a CGI script on the server. The script returns an HTML form, where the user further specifies the parameters. The CGI script processes the form parameters, extracts the needed data from the database and creates a document on the fly, which is sent back to the client station by the server. Since the CGI client/server architecture is stateless, each new query will initiate the same process and the result will be a completely new document. If the parameters of the previously completed form are involved in the new query, they should be memorized on the client station.

The two-step schema described above is appropriate only to query information (non-spatial and spatial) about a particular object. Many queries and analyses cannot be fitted in the schema because it is difficult to clarify the objects in advance. Examples of such queries are “show the highest building in the town”, “show all the administrative buildings”, “show the common walls”, “who are the owners of the buildings along this street”. Section 6 elaborates on several examples of spatial queries discusses both cases query of individual objects and complex queries.

Data modification. Since VRML in its current stage lacks *nodes* for direct database access, editing of data can not be organized implementing the standard VRML syntax. The modification, completed by the user inside the current VRML document cannot be transferred to the server either. For example, the operation “drag-with the mouse” described in a VRML document by a couple of standard *nodes* that allow the user to move the object inside the browser, will effect neither the original VRML document nor the object in the database. In the here presented approach, CGI scripts conduct the editing process. Similarly to the operations described above, HTML completed forms and VRML documents participate in input-output process. *Example 2,* Section 6 explains the sequence of operations. An alternative, which makes use of Java applets and the Common Object Request Broker Architecture (CORBA), is presented in [7]. The approach, however, requires extensions of the TCP/IP protocol, which are not widely implemented on the Internet yet.

Exploration and local modification address discovery and (or) temporary changes in the model on the client station. The exploration can be interpreted as a local query of some object properties, which are delivered by VRML documents, but can be activated and/or observed only on user request. Many examples of VRML worlds guiding the user through rooms, towns, scientific models, computer systems, etc. are available for access on the Web (see Congress Center, Airport Schipol, Twente Music Centrum). A combination of appropriate VRML *nodes* may facilitate even the decision-making process. For example, an urban planner may want to compare several architectural projects for a renovation of building facades. She/he can operate on the several new views, e.g. switch them sequentially and observe the effect. The new views can be organized as different texture, kept in separate image files on the server (see Section 6, *Example 2*). Other typical examples are: design of urban green areas and evaluation of the tree growth in a certain period of time. Different types and sizes of tree models can be prepared and sent to the client in one file. Only one of all the possible solutions is visible at a time, therefore, a perception of design is created.

In general, documents can be displayed at the client screen in a common window subdivided in frames, several new windows or combinations of them. In the prototype system, preference is given to a common Web window (split into several frames). Although separate windows provide the user with more freedom to resize and adjust observed models, the control over the windows and information inside is rather complicated.

The data delivered at the client site is displayed either by an HTML browser (text, 2D graphics, etc.) or a VR browser (3D graphics and text). For example, the query “show the way between the hotel and the nearest shop for shoes ” will result in a subset of objects (streets and surrounding houses). An appropriate animation can even route the user from the hotel to the shop. The result, in this case, can be displayed in one VRML document. A number of queries, e.g. “show the way between the hotel and the nearest shop for shoes and the prices there”, however, may require HTML and VRML documents to be created simultaneously. Unfortunately, due to restrictions in the CGI mechanism for dynamic creation of documents, the delivery of only one document per session is possible. The first line sent by the server is the MIME type of the document, which gives an indication to the browser which plug-in (in this example VR browser) to activate. This limitation can be compensated only at the price of a new client-server session. Hence, the example above will be subdivided into two steps: first, the VRML document describing the geometry will be displayed and second, a new user action, e.g. click with the mouse on the shop will create an HTML document containing the prices.

4 VRML for visualization and interaction

As indicated above, the VRML document can be created either as a simple document for visualization and navigation only or as a complex, point-and-click and dynamics enabled document. In the fist case, only the functionality of VR browsers, i.e. *fly-over, walk-trough, examine, pan, zoom* can be employed. In terms of GIS, simple VRML documents are applicable only for end visualization, i.e. no further information is to be queried .

The second type extends the ability to interact with the model almost unlimitedly. For example, each object in the current VRML document can be a clickable object invoking Java applets, CGI or Java scripts. Consequently, a new query to the database (on the server) or query of the VRML document arrived (on the client station) could be the next action. The new query could result again in a complex VRML document. To illustrate the extended possibilities of such approach, an user who needs accommodation information will be considered. In the initial VRML document, the first click on the door of a hotel (Javascript, the same document) will allow the user to enter. The second click on the reception desk (CGI script, new HTML document in a new window) will show prices and available rooms. The third click on a button of the lift will open the door (*Interpolators* and r*outes*, the same VRML document). The fourth clicking on a board to move the lift (CGI script, new HTML document in a new window) will ask for the number of the floor. The fifth click on the send button (CGI script, new VRML document) will display the corridor on the 4th floor, and so forth. In such a way, network analysis might be realized. In the example, five clicks of the mouse activate three CGI scripts (three new documents are dynamically created), one Javascript and a couple of VRML *interpolators*.

An interesting issue is dynamic composition of such a complex VRML document. The first basic operation, i.e. identification of a certain object, is already problematic. As was mentioned before, the VR browser is not a complete GUI, e.g. point-and-click operation is not a responsibility of the browser. The browser reacts on user actions (other than navigation) only if they are initially and explicitly described in the VRML document. A particular *sensor* has to be attached to a particular object before the user is able to interact with that object. The next step is the composition of the response. What does the user want to achieve selecting this object: text, graphics, image, spatial analysis, attribute information, data about the selected object or about other objects? In this approach, the decision on the type of *sensor*, the target object and the resulting event (CGI script or Java script, or appropriate VRML *nodes*, or files on remote servers), has to be taken by the CGI script during the dynamic creation of the document. Clearly, the set of CGI scripts becomes a critical element in the system and could cause problems on the server due to:

- drastic increase of the number of CGI scripts in case of complex sequential queries, which will complicate the script management
- sophisticated algorithms, which will require longer time for the dynamic composition of VRML documents
- long VRML documents, which will give rise to negative effects in two directions: occupation of the server and long waiting time at the client site.

So far, only a user action has been considered as a possible input event to initiate an action. As was mentioned at the beginning, VRML is capable of sensing two other types of events: 1) dynamic interactions among objects and 2) time related changes. For example, a collision between two moving objects, a collapse of a building after a certain period of time or due to a contact with another object (e.g. plane), a crash of a plane if touches the ground, etc.

Apparently, VRML has the potential to describe complicated static and dynamic spatial interrelations. The dynamic creation of such complex VRML documents, however, is not an easy task.

To avoid or reduce the undesirable effects of CGI scripting and facilitate management of dynamic interactions, the proposed system stores appropriate supplementary information about *behavior* of objects in the database. An object is described by its *attributes*, *relationships* and *behavior* (see [16]). The *behavior* in the geometric domain defines dynamic changes and interactions related to characteristics of objects such as shape, position, colour, etc. However, *behavior* can be extended to comprise changes and interactions in the virtual world. Thus, a variety of parameters, scripts, small VRML files, animations, etc. per object that facilitate and simplify the work of CGI script can be captured in the database. The result is a possibility of CGI scripts standardization, which consequently decreases their number and reduces their size. Large worlds (i.e. long VRML files) can be partitioned into several smaller ones by assigning *behavior* to specific objects (doors, windows, etc.). The world can be reconstructed afterwards on user request as only one script is sufficient to deliver the entire file. The hotel example given above will have a separate object “board in the lift” with a *behavior* “moves up”. The *behavior* can be implemented on a database level as “*on-click* activate *which-floor* CGI script” with two fields *event initiator* and *event response*. The field *event response* contains the script, which controls the movement, and some parameters to identify the floor. The number of parameters may vary depending on the type of the interaction. The plane crash on the ground, for example, will need three parameters: to identify the event (touch), the conflict object (ground) and the resulting action (crash, e.g. short animation).

5 Conceptual schema

The database integrates a variety of information in order to serve a large spectrum of tasks: spatial and thematic analysis, 3D visualization (some applications may need realistic), manipulation of objects and introduction and control of *behavior*. The exhaustive list of requirements to the database is quite long, therefore only the most important ones are mentioned here:

- storage of thematic and spatial information per object

- storage of physical properties of objects (material, texture)
- support of spatial analysis (neighborhood, proximity)
- ability to provide data for visualization in VRML, i.e. faces, orientation of faces and points bordering the faces.
- storage of parameters describing the behavior of objects

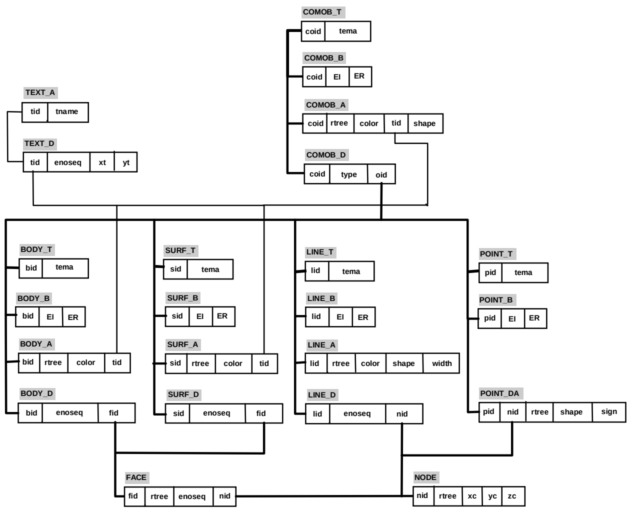


Figure 2: Relational implementation of the conceptual schema

Furthermore, the result of the user query is dynamically created, which asks for response times that are acceptable to the users. The time interval between sending the query and displaying the data is a compound of: 1) the time needed for data traveling between the server and the client, 2) the time for data parsing at the client station, 3) the time for database traversing and 3) the time for document composition. Indeed, one way to speed up the process and improve the performance is a careful selection of client-server hardware and software, in particular DBMS, language for programming and browser for visualization. An additional way, which is discussed here, is optimization of the conceptual schema.

The mapping of the conceptual schema onto a relational data structure is given on Figure 2. Four types of geometric objects, i.e. *point, line, surface and body* are supported. The basic constructive objects are *face* and *node* (not to be confused with *nodes* in the VRML syntax).

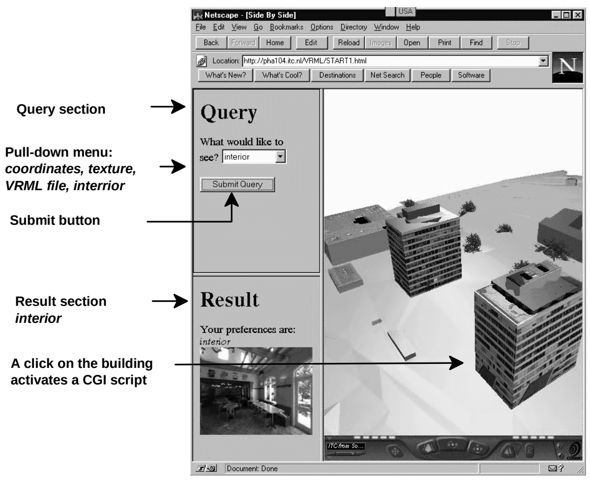
Each object is represented by four relational tables. Tables with extension _D contain the information about shape and position of objects, tables _A keep parameters about physical properties and tables _B give some information about eventual *behavior* of objects. Since the thematic information is not elaborated, tables _T are limited to a simple indication of the object class. Each object has a unique ID. Complex objects are described as aggregations of body, surface, line or point objects. Complex objects may have their own behavior. For example, the building of the hotel can be linked to a script (activated by input event "mouse over"), which provides some thematic information. The entrance door, however, may react independently (activated by "mouse click").

A coding of the geometric objects and corresponding constructive objects based on R-tree grouping is provided (field *rtree* in _D, FACE and NODE tables) in order to speed up traverse of the tables, facilitate the maintenance of the information and extend the spatial analysis toward directional analysis. More details about the conceptual schema and the R-tree can be found in [16].

6 Examples of Queries

Two 3D urban models are created and organized according to the conceptual schema to illustrate the functionality of the system. The first 3D model contains photo-textured buildings, DTM, trees and lampposts of the central part of Enschede, the Netherlands. The model is relatively small but all the geometric objects according to the conceptual schema are represented. The second model consists only of buildings (presented as BODY objects) of Vienna, Austria (see Table 1). In contrast to the first model, the second one is relatively simple but with a size which can be expected from real models. The buildings do not have textures assigned. All the examples presented in the paper can be accessed at http://barley.itc.nl.

Example1: Query of spatial and thematic information. The example is a realization of the two-step query discussed above, i.e. demonstrates an extraction of information about a particular object. The user has access to the VRML document with a point-and-click option and can visually choose an object (e.g. building). A click with the mouse on the building activates a CGI script, which delivers the *Query-Result* sections (see Figure 3)


Netscape - [Side By Side]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: http://pha104.itc.nl/VRML/START1.html
What's New? What's Cool? Destinations Net Search People Software
Query section
Pull-down menu: coordinates, texture, VRML file, interrior
Submit button
Result section interior
A click on the building activates a CGI script
Document: Done

Query

What would like to see? interior

Submit Query

Result

Your preferences are: *interior*

Figure 3: Query interior of a building

In the *Query* section, a pull-down menu offers several choices: coordinates of the building, the image file used to texture the walls of the building (in this example only one image file is used for all the four textured walls), a VRML document, containing only the building of interest, and the interior of the building. The request has to be sent to the server by checking the Submit button. The CGI script creates and sends to the client a new HTML document in the section *Result*. Figure 3 shows the interior of the building as an embedded panoramic movie (accessible through the *SmoothMovie* plug-in). Practically, any standard Web document (movie, sound, animation) is applicable for visualization of results.

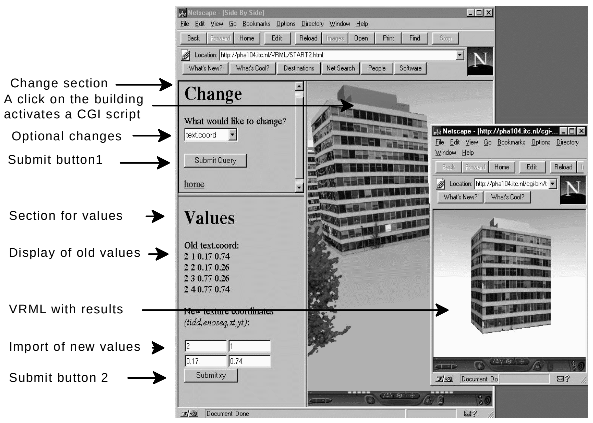


Figure 4: Changing texture coordinates of a building

Example2: Modification of information. The next example presents the sequence of steps to change information on the database level. Again, the initial VRML file has to be equipped with the necessary *sensors* to detect user actions. A click on a building activates a CGI script, which delivers a form with *Change-Values* section. The snapshot shown in Figure 4, shows the interface to change texture coordinates. This refers to a case when the user wants a replacement of an old façade with a new one. VRML syntax requires the name of the new image file and ties points to correctly map the image onto the corresponding face. Here, only the steps to adjust the texture coordinates will be explained. The submission (*Submit button 1*) of an item selected (i.e. *texture coordinates*) activates a search in the database for old image coordinates and then displays them in the section *Values*. The new-typed values (sent by *Submit button 2*) are replaced in the corresponding fields in the database and a VRML document considering the new coordinates is created for validation. The Web user visually inspects the texture mapping and corrects the values if necessary. Usually, several repetitions of the procedure are sufficient for complete adjustment. In a similar way, geometry coordinates and names of image files used for texturing can be replaced (edited) with new ones.

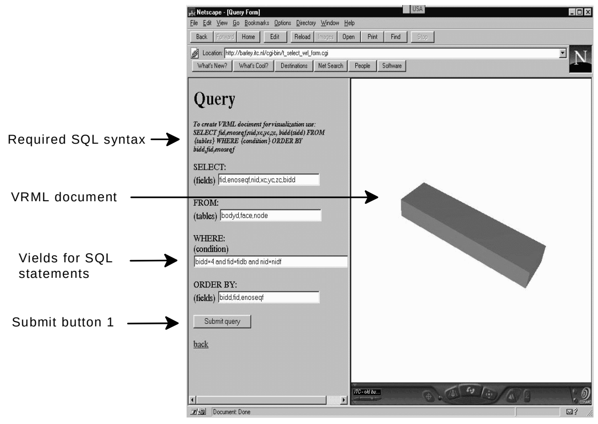
The second textured building in the Figures 3 and 4 is an example of a *local query* as was defined above. A click on that building changes the façade, i.e. the image file used for texturing is replaced with a new one. A combination of *sensors*, *routes* and Javascript provides a “switch-image-on-click” operation. Thus the user may change an arbitrary number of façade images by sequential clicking on the building. However, no connection to the server is made because all the images are included in the VRML document in advance. The mechanism reduces the traffic to the server, however, it has to be used carefully bearing in mind the size of the VRML file.

The examples above represent operations on one object and the query and modification are restricted to the choices given in a pull-down menu. Despite the obvious limitations, the approach is very appropriate for a broad audience of Web users. A special knowledge about the information in the database and the conceptual schema is not necessary.

Example3: SQL queries. Experienced and qualified users can be allowed to send SQL statements to the database and visualize the result of queries in VRML documents (in case of spatial queries). Several SQL forms dealing with different situations, i.e. “free SQL query”, “SELECT”, “SELECT+visualize ” are designed and available for testing in the experimental site. The paper discusses only the “SELECT+ visualize” fill-out form (see Figure 5).

The syntax of the VRML requires a structuring of the geometric data different to the one in the conceptual schema (will be discussed later). A CGI script can re-organize the geometric data obtained from a query only if they are appropriate for a VRML document. For example, the query “which are the walls of building 1” can be represented by the following SQL expression (see also Figure 2):

SELECT fid FROM bodyg, face WHERE bidg=1 and fidg=fid



SELECT: (fields) fid,enoseqt,nid,xc,yc,zc,bidd
FROM: (tables) bodyd,face,node
WHERE: (condition) bidd=4 and fid=fidb and nid=nidf
ORDER BY: (fields) bidd,fid,enoseqf

Figure 5: SELECT and visualize in VRML document

The extracted data (ID of faces), however, is useless for a VRML document. SQL statements have to ensure sufficient data in a strictly defined order (see section 7). In the example, the required SQL expression is given in the HTML fill-out form. An intermediate HTML document (not visible on Figure 5) guides the users in his/her decision whether to proceed further with a VRML document creation. The intermediate step can be avoided by control over fields in the form and data extracted from the RDBMS. Such control, however, will restrict the form to creation only of VRML documents, therefore it is omitted.

The free access to the database provides a mechanism to specify and visualize a wide range of spatial queries. Each request in the spatial domain (formulated by spatial or non-spatial conditions), which can be described in one SELECT statement, in practice, can be visualized in a VRML document. Examples of such queries are: "which are the buildings higher than 20m", "show the buildings in a particular area", "show all the streets", "show all the administrative buildings", etc.

Example 4: Embedded queries. The last examples available on the experimental site are related to spatial queries and analysis, which cannot be expressed by a singe SQL statement. A solution based on a series of specialized HTML fill-out forms and VRML documents (containing the resulting objects) is implemented, e.g. forms to clarify neighborhood relationships (i.e. "common nodes" and "common faces"). The role of the user in such queries is to indicate the objects and relationships that are to be analyzed. The CGI scripts processing such queries assist the RDBMS in the completion of the query, in contrast to the previous examples where they are responsible only for the transfer of parameters and dynamic creation of documents.

7 Performance

All the examples and experiments are completed on a prototype system with the following characteristics: **server** PC Pentium 133MHz, 96MB RAM, LINUX operation system and Apache Web server; **client** PC Pentium 166MHz, 64Mb RAM, Windows'95 operation system, Netscape 3.0 Web browser and Cosmo player VR browser. The freeware MySQL client-server RDBMS is used to host the data. Perl is the programming language for CGI scripting as the supplementary CGI.pm and DBI.pm libraries to create fill-out forms and connect to RDBMS are utilized.

A short look in "the kitchen" of extracting spatial information will be made before reporting the performance tests. Spatial queries intended for visualization pass two compulsory phases. First, the data needed to complete the user query is collected and, second the geometry of the objects, which will participate in the virtual world, is extracted. The two phases can be written as "find the data with respect to the user query" and "find all the data necessary for the VRML file". Indeed, the number of objects included in the VRML document may vary depending on the manner preferred to portray the result (the original scene with highlighted elements or only the objects of interest). Anyhow, the final world created has to contain at least the geometry of the objects elected by the query. The VRML syntax for solid geometry (irregular shapes are considered) requires 1) coordinates of the vertices representing the bordering faces of objects, 2) proper (anti-clockwise) orientation of faces, 3) corresponding texture files and texture coordinates (if they exist). The coordinates have to be listed per object, preferably without duplications. The description of the faces is given by identifiers, which are the current position of the coordinates in the VRML document, starting with 0. Clearly, this structuring differs significantly from the organization of the geometry in the conceptual schema (see Figure 2). However, a particular subset of data extracted in a certain sequence, i.e. *fid, enoseq, nid, xc, yc, zc* and order: *fid, enoseq* can be further re-ordered according to VRML rules.

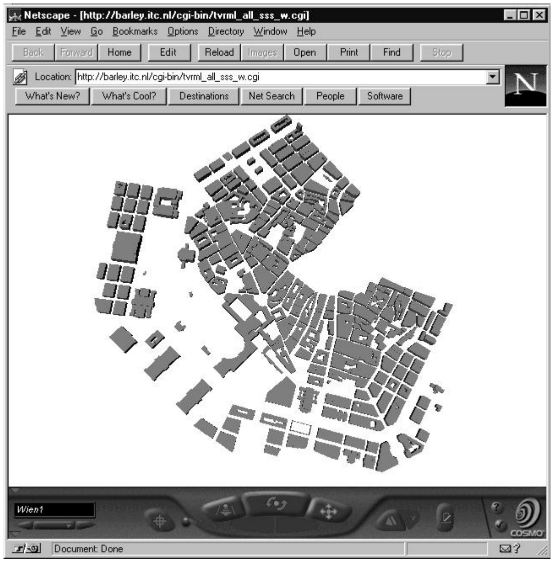

Netscape - [http://barley.itc.nl/cgi-bin/tvrml_all_sss_w.cgi]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: http://barley.itc.nl/cgi-bin/tvrml_all_sss_w.cgi
What's New? What's Cool? Destinations Net Search People Software
Wien1
COSMO
Document: Done

Figure 6: 400 buildings from Vienna (545Kb)

The SQL operator may or may not include the two phases in a single SELECT statement. For example, the query "visualize the buildings inside a certain area" can be expressed by one SQL statement in contrast to the query "check for duplicated points". The representative tests are limited to one-line SQL expressions to ensure equal conditions while comparing CGI scripts and RDBMS queries. Thus the results reported here are based on queries, which can be formulated in the following SELECT statement:

SELECT fid,enoseq,nid,xc,yc,zc FROM <tables> WHERE <condition> ORDER BY fid,enoseq

For example, the VRML document of the BODY object with ID 12, will be obtained by the SQL statement (see also Figure 2):

SELECT fid, enoseq, nid, xc, yc, zc, bidg FROM bodyg, face, node WHERE bidg=12 and fidg=fid and nidf=nid ORDER BY fid, enoseq

Table 1 contains the size of the data sets (Vienna and Enschede) in terms of objects regarding the conceptual schema.

The experiments aimed at estimating the penalty cost of 1) data extraction and 2) creation of a VRML document and transmission to the client station. The first experiments are pure database SQL queries executed on the server by the RDBMS. The column **BD** (Tables 2 and 3) contains the results of time needed by the RDBMS, while the **On the Fly** represents the approximate time for the query, the creation of a VRML document, delivery and visualization on the client station. Hence, the difference between the two results is an estimate of the time needed to transfer the data over the Internet and display in the VR browser. The last three columns give an idea about the size of delivered data, number of **faces** and the **number of records** extracted from the database . The number of records coincides neither with the vertices (not shown) nor with the faces, which is caused by the required special order and particular set of data of the SQL query discussed above.

Table 1: Content of the experimental data sets

	Enschede	Vienna
Composite objects	2	-
Body objects	11	1 600
Surface objects	19	-
Line objects	7	-
Point object	8	-
Faces	1533	92 268
Nodes	960	30 756
Textures	7	-

The Enschede data set was considered too small for performance estimation, therefore R-tree coding and B-tree indexing of the database were not performed. The four objects involved in the experiments, i.e. a building, a surface, a composite object and DTM represented as a surface object (see Table 2) demonstrated the capability to extract all four types of geometric objects. The performance test, however, has proved the importance of optimization issues. Although, individual objects can be extracted in a very fast manner, the object DTM requires rather long time.

Table 2: Enschede data: database query, query on the fly and size of extracted data

Objects	DB (sec)	On the fly (sec)	Faces	Number records
A building	0.2	2	10	48
A surface	0.06	2	1	12
A composite object	0.2	2	15	72
DTM	30	50	1399	4197
Entire model	40	60	1533	4293

The search in the second data set is optimized for speed in two ways. First, a spatial restriction of the query range is introduced by R-tree codes and second, the B-tree indexing provided by MySQl is exploited. The most frequent visited fields, i.e. *fid, nid* in FACE and NODE tables are indexed. The results of the nine representative queries are shown in Table 3.

Table 3: Vienna data: database query and size of extracted data

Number Buildings	DB: (sec)	On the fly (sec)	Faces	Number records	VRML doc (Kb)
1	012	4	13	66	2
2	0:17	4	25	126	5
10	0:30	5	89	414	11
20	0:65	5	223	1 098	28
50	1:70	8	636	3 216	77
200	6:60	45	2 414	12 084	295
400	12:2	80	4 765	23 790	545
600	19:50	140	7 223	36 138	839
1600	52:40	330	18 578	92 268	2 306

As can be seen, the VRML documents smaller than 500Kb can be delivered in 1-2 minutes, which can be considered an acceptable time for a waiting for a Web document. Larger files should be delivered only in exceptional cases. Regarding the complexity of the VRML document, however, a document with a size about 500Kb may contain several neighborhoods or only few textured buildings and DTM (see Figure 6, 7). In this context, a reasonable balance between type of information (detailed vs. schematic) and manner to visualize (textures vs. colors) always has to be pursued.

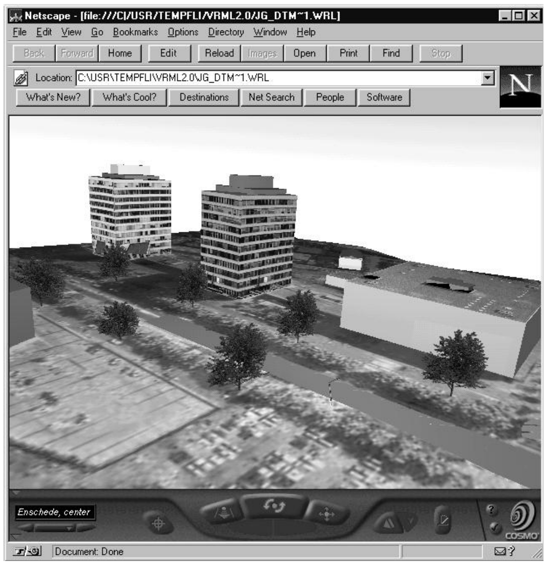

Netscape - [file:///C|/USR/TEMPFLI/VRML2.0/JG_DTM~1.WRL]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: C:\USR\TEMPFLI\VRML2.0\JG_DTM~1.WRL
What's New? What's Cool? Destinations Net Search People Software
Enschede, center
COSMO
Document: Done

Figure 7: Several buildings from Enschede (405Kb)

It should be mentioned that not all the options to optimize the system were explored. The interest was only on the optimization of the conceptual schema due to assumption the that different implementation may produce different results. For example, the RDBMS (MySQL) performs essentially better results when less tables are joined. A split of the SELECT SQL statement into two given as an example above, decreases the time for database traversal with 10-12%.

SELECT FIDI FROM bodyg WHERE bidg=12 and fidg=fid;

SELECT fid, enoseq, nid, xc, yc, zc FROM face, node WHERE fid=FIDI and nidf=nid ORDER BY fid, enoseq

8 Conclusions

An approach for a 3D GIS on the Web has been presented and discussed. An essential role in this approach in granted to VRML. Concerning visualization and exploration, VRML has already shown a capacity to design highly realistic and dynamic worlds. Here, it has been demonstrated that VRML in combination with HTML forms, behaves as GUI to formulate a wide range of SQL queries. The presented examples of 3D spatial data queries, modification and validation of the changes by visual inspection, are few of the 3D GIS operations, which can be organized on the basis of VRML. The Web users, who used to have at their disposal only end VRML documents and expects little or no processing of 3D data across the Internet, can reverse their attitude.

The dynamic creation of complex VRML documents enabling further query, providing enhanced visualization techniques, raises questions for effective solutions. Some of the issues, i.e. adequate database organization and performance optimization were addressed here. The conceptual schema presented maintains 3D GIS information (3D topology, spatial and thematic data) as well as the behavior of objects. As was illustrated the behavior of objects facilitates the dynamic creation of Web documents. Applied techniques for time optimization (i.e. R-tree coding and B-tree indexing) speed up the traversal of the database and thus reduce the waiting time at the client station.

Finally, attention is drawn by the author to the implemented components: MySQL, Apache, Linux, Perl and related libraries. All the modules are freeware software downloaded from the Web. Bearing in mind their easy installation and maintenance, constantly stable work during all the experiments and positive results obtained from performance tests, they are heartily recommend for both research and real applications.

References

[1] 3D Web Consortium, 1999, http://www.sdsc.edu/vrml

[2] Abel, D.J, K. Taylor, Ackland and S. Hungerford, *An exploration of GIS architectures for Internet Environments*, In Computers, Environment and Urban Systems, Vol. 22, No.1, pages 7-23, 1998

[3] *Airport Schipol*, Amsterdam, the Netherlands http://www.schiphol.nl/maps/3d.htm, 1999

[4] *Alpentour Steiermark*, Austria 1998, http://www.alpentour.at/index.html

[5] Ames, A.L., 1996, *VRML 2.0 Sourcebook*, John Wiley&Sons, Inc., New York, USA

[6] *Congress center*, Graz, Austria, http://www.gcongress.com, 1998

[7] Coors, V. and V. Jung, *Using VRML as an Interface to the 3D Data Warehouse*, In Proceedings of VRML'98, New York, 1998

[8] Doyle, S., M. Dodge and A. Smith, *The potential of Web-based mapping and virtual reality technologies for modeling urban environments*, In Computers, Environment and Urban Systems, Vol. 22, No. 2, pages 137-155, 1998

[9] Gahedan, M., *Scatterplots and scenes: visualization techniques for exploratory analysis*, In Computers, Environment and Urban Systems, Vol. 22, No.1, pages 43-56, 1998

[10] Kofler, M., *The Sodahall VRML Jumpthrou*, Project report, UC, Berkeley, 1996

[11] Pilouk, M., Integrated modelling for 3D GIS, PhD thesis, ITC publication, 1996

[12] Schickler, W., *A Virtual reality model for a major international airport*, In Proceedings of the Ascona Workshop'97, Automatic Extraction of Man-Made Objects from Aerial and Space Images, Monte Verita, Switzerland, pages 367-376, 1997

[13] *Twente Music Centrum*, Enschede, the Netherlands, http://wwwseti.cs.utwente.nl/Parlevink/Projects/Muziekcentrum/codecompressed/vmc_zonderschisma.html, 1999

[14] *University of Rostock*, Germany http://www.agr.uni-rostock.de/iggi/cebit_e/, 1999

[15] Templfi, K., *3D topographic mapping for urban GIS*, In ITC journal, 1998-3/4, pages 181-190, 1998

[16] Zlatanova, S. and M. Gruber, *3D GIS on the Web*, In ISPRS, Com. IV, Stuttgart, Germany, pages 691-699, September 1998