Supporting Dynamic Information Visualization with VRML and DataBases

Thomas Risse, Martin Leissler, Matthias Hemmje, Karl Aberer
GMD - German National Research Center for Informaation Technology
Integrated Publication and Information Systems Institute (IPSI)
Darmstadt, Germany
[risse, leissler, hemmje, aberer]@darmstadt.gmd.de

Introduction and Motivation

Due to increasing amounts of often complexly structured data collections within global networks, appropriate visualization methods to support information retrieval, i.e., browsing and search tasks [Hemmje et al. 94][Massari et al. 98] as well as, e.g., data mining tasks for naive end users of information systems gain growing importance. The available graphics performance of network client systems is steadily growing, too. Therefore, methods for three-dimensional visualization can be applied more and more easily.

Such systems can be used in several application areas. One example is information visualization within information systems [Rorvig & Hemmje 98]. Another important application area is electronic commerce where it can be used to, e.g., generate virtual shopping malls, virtual libraries, virtual galleries, or to visualize trade fair and congress information [RISSE & HEMMJE 97]. A virtual-reality overview of a trade fair hall can, e.g., display information about all companies exhibiting their products on stalls in the hall. The possibility of walking through a three dimensional visualization of the trade fair hall, enables visitors to plan their routes through the trade fair more easily, long before the actual visit takes place. The benefits for exhibitors are, e.g., that they get an impression of their stand and its position among competitors stalls. Furthermore, the visualization of the hall and stands could be used as a metaphor to access further information about the exhibitors and their products, or to send requests for meeting appointments to staff members of the exhibitors.

The Current Situation

The Virtual Reality Modeling Language (VRML) is widely used on the Internet to describe three-dimensional scenes[CONSORTIUM]. However, today VRML is more or less only used to generate static scenes, mostly for demonstrational purposes. Such static scenes lack properties like persistence, scalability, and security[VRMLSPEC].

Therefore, after loading the scene, users start their VR exploration always in the same state of the choosen VR world [VRMLFEAT]. All changes to scenes are not persistent. Furthermore, the provided virtual worlds have limited sizes because the browsers that are used to visualize them must hold the scene description completely. Even in the most advanced versions of VRML, the scenes have always the same complexity regardless of the capabilities of the browser [VRML97][VRMLCHANGE]. This means, that the scenes are not scaleable, have a limited size, and that security and communication control mechanisms are not available. Finally, the dynamic generation of scenes depending on different access rights of different users, is not possible. In consequence, VRML is mostly used to only visualize static VR environments and dynamically generated VRML scenes do not exist in many
places because the development of tools and applications to support their generation is a costly effort [AMES97].

**An Approach Towards Supporting Dynamic Scene Generation**

Due to the fact that much information is stored within databases it could be visualized by means of a dynamic generation of three-dimensional scenes. Today the dynamic generation of two-dimensional HTML pages is a standard functionality of all database systems. Therefore, the idea outlined in this paper is to generate VRML scenes in a similar way.

To achieve this goal, traditional Database Management Systems (DBMS) are used as a basis to provide the foundational functionality for VR information management and information visualization. By this means, base functionality for security and persistence is already available. Therefore changes of values are persistent for future use and are also available to other users [COD70]. But this requires that the client can directly communicate with the DBMS to access VRML and other data stored in its databases.

Furthermore, DBMS are also considered helpful to achieve scalability of VRML scenes. Providing a DBMS with VRML data management capabilities make an optimized streaming of any sub scene possible. Also a user defined logical scaling w.r.t., e.g., application requirements is possible because the structure of the overall scene w.r.t. to its composing components is accessible.

In addition to enabling logical and physical scaling, the security system of the DBMS allows controllable access to objects, scene parts or whole scenes. Therefore it is possible to hide information for some users and provide others with full access. In the “trade fair hall” scenario mentioned above, it is, e.g., possible to display the rent of a stand only to authorized staff members or customers but not to visitors.

While persistence and security are basic properties of a DBMS, it also has to enable permanent storage of VRML scenes or parts of them together with related metadata in a well structured way. In addition the DBMS has to provide functionalities for retrieving and manipulating VRML data. If this is achieved, it becomes possible to work with VRML scenes in DBMS supported applications as with any other data type. For example all stands of a trade fair can be stored as VRML scenes together with the company name and the position within the hall. Finally it is possible to select a subset of all stands with special properties and merge them together to one new scene to, e.g., display only stands which are already rent to customers.

However, due the fact that most stored information is not of a three-dimensional nature, the system must also be able to generate new VRML scenes out of, e.g., operational business data represented by traditional data types or, e.g., multimedia represented by specialized media data types. For example within the trade fair system statistical numbers about expected visitor flows in a hall could be visualized with smaller or bigger arrows.

Beside the above outlined persistence properties, an event handling system is necessary, too, to support interaction between the VRML scene and the DBMS or the VRML scene and users. Therefore if, e.g., users or application programs change some data which should results in changes of the scene, the database should send an event to all corresponding users of that VRML scene. The VRML event handling within the scene can then react in an appropriate way by, e.g., changing the scene. Such a
reaction can be to reload a part or the whole scene. Another possibility is only to reload the data to be visualized instead of a VRML scene. In this case the client must be able to generate VRML data out of the operational data that is to be visualized. This means, that the client has to access the database directly and has to calculate the new parts of the scene from the received data result during the runtime of the scene.

**VRML extensions**

Our implementation approach of the previously described mechanisms is based on extensions to the database system as well as on language extensions to VRML itself [DBWORK] [VAG].

First of all, VRML will be extended with a database oriented node which is used as a so called ‘server side include’ (SSI) mechanism. This means that the VRML scene is parsed within the DBMS to expand the SSI node and replace it with other, standard VRML nodes. Therefore, the SSI nodes themself will never be send to the client. Each SSI node contains a SQL statement [SQL3] [SQLREF97] which will be executed during the expansion of the scene at run time. The result of this expansion can be processed in three different ways. The first option is to set a field value of specific VRML node directly from the query result. To achieve this, the syntax of the SQL SELECT command has been extended in the following way:

```
```

The extension to the normal SELECT statement is that the destination node and field of the result is defined with the INTO clause. Another possibility to process the results of a query into visual meta objects is textual replacement. Within the SSI nodes some variables can be defined. If the INTO clause of the SQL statement contains one of these variables, it can be replaced by the corresponding value of the query result.

The third method to integrate query result values into a scene description is to instantiate a template. Such a template contains several nodes whose values can depend on the query result. In this case, a new instance of the template is generated for each row and the resulting instances are grouped together into one new group node of the expanded VRML scene description. Following this approach, the template will always be expanded to replace the original node even if it’s empty.

Finally, to enable DBMS trigger mechanisms, one also needs another new node which is implemented as a VRML prototype. It receives events from the DBMS and forwards them to the nodes of the scene. To achieve this, the event routing is done applying the standard VRML routing mechanism.
Database extensions

Besides providing the basic features of persistency and security, a traditional DBMS cannot directly handle VRML scenes. Therefore it needs to be extended. To achieve this, object relational database systems are a good choice because they can be extended with new data types and operations in a similar way like object oriented DBMS but still provide sufficient support for handling relational data in the traditional way [STO96]. As most structured information is still stored in relational DBMS, this choice of development platform is consequent.

The basic part of the necessary implementation work is the definition of the VRML data type and its related functions. An object of the VRML data type stores complete and valid VRML scenes. This is achieved by implementing at least basic support for a simple VRML scene node. On the VRML data type, some functions are defined to be used for type conversion, merging, import and export of predefined scenes as well as to support transformations on the whole scene. The merging of scenes can, e.g., be used in selections. If the result of a selection is a group of single VRML scenes they have to be converted into one whole scene before sending them to the client. Before sending any scene to the client each SSI node must be expanded as previously described.

Finally a trigger mechanism is implemented in a way that triggers events which are routed over an event server to the clients. The event server manages the active VRML clients and distributes the trigger events. By this means, the database is not influenced if communication problems occur.

Conclusion and future work

With the described implementations we have achieved support for most of the above mentioned requirements for dynamic information visualization support. Security and persistency are provided as a basic property of the DBMS. With the VRML data type and its corresponding functions the storage, retrieval and manipulation of VRML data is possible and implements server side persistency of VRML data. Furthermore, SSI mechanism implements support for the dynamic generation of VRML scenes out of operational data as, e.g., traditional relational data, multimedia data or VRML data itself. Multi user environments and interaction oriented applications can use the trigger mechanism for the communication between database and the clients. These components can be used for server based information visualization. To close the interaction loop by enabling clients to also interact with the server some further development is currently on its way. At this time, the described DBMS extension components can make changes to VRML scene values persistent but for true
persistence the client changes to the scene must be send to the DBMS to initiate an update in the corresponding database. To achieve this, a so called VRML SQL runtime node is currently under development. It will also have the ability to execute all sorts of SQL statements and to generate new VRML nodes out of the corresponding results. The result processing mechanism that will be applied is the same as already implemented for the server side include node. Together with the trigger mechanism, these developments will provide a sound base for the creation of multi user VRML environments for the above mentioned application areas. Finally, server side supported scalability is also an approach which is currently tackled and will be implemented soon. It will enable the generation of large VRML worlds regardless of the client's hardware platform.

Literature


[Rorvig & Hemmje 98]


[VRMLFEAT] VRML 2.0 Features, [http://www.vrml.org/Specifications/VRML2.0.old/FINAL/Overview.html#Summary](http://www.vrml.org/Specifications/VRML2.0.old/FINAL/Overview.html#Summary)


[VRMLCHANGE] Differences between VRML 2.0 und VRML97, [http://cosmosoftware.com/developer/moving-worlds/spec.DIS/index.html#Summary](http://cosmosoftware.com/developer/moving-worlds/spec.DIS/index.html#Summary)