MetropoVis: Time-Dependent Real-Time Rendering of Large and Photorealistic Virtual Cities

Peter Borovský[1], Gerd Hesina[2] and Robert F. Tobler[2]

[1]Department of Applied Computer Science, Comenius University, Bratislava, Slovakia, [2]VRVis Research Centre, Vienna, Austria

1 ABSTRACT

A high quality visualization system for changing urban environments is a powerful tool for city planners. In order to accommodate the time-dependent nature of a city in such a system, we have built a four dimensional information space which facilitates both real-time rendering and database management. Our system consists of a Database Server, which solves GIS-related problems, and a fast visualization tool called AveViewer based on the Advance Visualization Engine (AVE) developed at the VRVis Research Center.

As a basis for data exchange between the viewer and the database, as well as for importing data, we chose VRML – an open 3D graphics format. The database server, utilizing MySQL 4.1 satisfies the OpenGIS standards, and facilitates the management of large amounts of structured geometric, raster and metadata content extended by timestamps. AveViewer uses this time-dependent information for a number of unique visualization methods, which are especially suited for urban planning.

Our system was tested on the very first 3D model of the Bratislava historical city centre (result of the MetropoVis data case study), as well as on other valuable datasets of Graz and Vienna.

Figure 1: Large-scale modeling example – Bratislava Castle and nearby Old Town textured from aerial images.

2 PROBLEM STATEMENT

Recent advances in the rendering speed of commercially available graphics cards make it possible to render large photorealistic models on commodity hardware. Hence, mechanisms for storing large models are required. This paper describes our efforts to build a databse server which is optimized for storing GIS realated data and meets general requirements for storing the kind of time-dependent data that is precent in data bases used for urban planning.

3 FROM RAW DATA TO VIRTUAL MODELS

A variety of different modeling techniques is used for assembling a virtual city. The majority of these techniques deal with processing digital terrain models (DTM), aerial images, laser-scanned data, digital photographs, cadastral or other ad-hoc quantitative information. For an overview of a low-cost data acquisition and modeling techniques that maintain high-quality results we refer the reader to the overview by Ftacnik et al. [FTACNIK04]. High-quality modeling software for creating city models has been described by Karner [KARNER]. Dudek and Blaise [DUDEK03] demonstrate the importance of data time classification and show appropriate visualization solution..

This paper is not concerned with the analysis of the modeling methods used in urban plannen, but demonstrates a system for fast and visually valuable interaction between the user and a large virtual space. Nevertheless, any meaningful information visualization system strongly depends on the quality of the input data, which leads us naturally to a rough taxonomy of available virtual urban models, based on their applicablity in a photorealistic rendering system. The next three figures show examples of three photorealistic categories we recognize: large, medium and small-scale. They differ in the highest possible visual detail, the human effort to build

them and the amount of data they allocate (and thus the visualization speed complexity as a trade-off based on the amount of detail that is present in the model).

Large-scale models are sufficient for data-load demanding flythroughs, where an observer is expected to view the objects from farther than about 30 meters. One typical application for such model is a city web presentation, limited by low network traffic requirements. A large-scale web presentation gives virtual visitors the most attractive view to the city – the bird eye's perspective, but does not allow them to access resources that are too detailed (i.e. costly to retreive).

Figure 1 shows a photogrammetrically constructed model, textured by a quick ray-casting method (described by Ftacnik et al. [FTACNIK04]). Aerial image resolution is 10 cm, sufficient for 30 m close-up in virtual space (at a field of view of 90 degrees).

While there are many automatization techniques known for large-scale city modeling, small-scale models require a lot of effort to create (essentially these models are mostly hand-made). They have to include all features the viewer can notice in the real object, allowing him to view them from the closest possible distance in the virtual space. Full-feature requirement prohibits to use the aerial data as an only data source (problems with overlapping roofs, etc.). Small-scale methods are mostly restricted to one or a limited cluster of buildings, since the creator has to capture the essential information directly from the physical object location. Figure 2 shows the model of St. Martin's Cathedral in Bratislava assembled from high-quality digital photographs.

Figure 2: Small-scale modeling example – St. Martin's Cathedral in Bratislava created in Caligari trueSpace 6.5 by Marek Zimányi.

Compromises of the large and small-scale approaches with both huge domains and acceptable details are often found in many virtual city projects. Typical usage of the medium-scale model is a virtual walkthrough [VOP01] – the user is allowed to move freely even in the narrow streets, but must not expect very smooth details in close-ups smaller than couple of meters (unless there is a model substitution in a higher level of detail).

Figure 3: Medium-scale modeling example – French Institute in Bratislava textured by Ján Lacko.

4 METROPOVIS DATA CASE STUDY – OLD TOWN IN BRATISLAVA

In the MetropoVis project, we have developed software solutions capable of dealing with large amount of very detailed geometrical and other resources. For testing purposes, we needed to have much finer than just low-cost and large-scale models accessible, thus we have established a team of creatives, building an original virtual city model. Due to the lack of such models in Slovakia, we have chosen its Capital and set a goal to virtually reconstruct the current historical centre of the Bratislava Old Town. With limited time and human resources available, we decided to produce a medium-scale model (Figure 3 displays a fragment), enriched with a couple of selected buildings in the highly detailed small-scale versions (Figure 2).

Figure 4: MetropoVis Case Study domain – historical centre of Bratislava Old Town.

36.5 hectares of historical buildings were photogrammetrically processed from aerial images by the Eurosense company. Roofline vectors of hundreds of objects together with a detailed terrain model formed the basis for further improvements. We found out that geometry of a roofline box model, as demonstrated in Figure 6, is almost satisfactory for the medium-scale presentation. Most complex wall formations can be incorporated into textures, which saves several hours of geometry refinement and texture mapping per building (e.g. the flattened porch entry to the French Institute in the Figure 3). The only exceptions are some essential structures bulging out of flat façades thereby preventing the virtual user to see them from a side. The biggest possible model improvements thus are concerned with obtaining adequate texutres. In our case study, we are investigating some of the known texturing methods (the ultimate method is a combination of multiple approaches). Final results including a cost and quality analyzis can be found in the reports published on our project web page [MetropoVis] .

5 VRML AS A COMMUNICATION MEDIUM

Obviously, any software modules of a larger system handling large amounts of data need to cooperate with each other and the outer environment via reasonably standardized means. From a lot of 3D file format specifications contemporarily available, we chose the best known and accepted open standard for scene description – the Virtual Modeling Language [VRML97]. Our proposed system accepts new models as the VRML scenes, which are loaded by and transferred into the inner format of the Database Server. Next, the Database Server replies to the AveViewer geometry queries in the VRML as illustrated in the simplified dataflow diagram below. The rest of the paper shows our proposals for each part of the dataflow, subsequent to the modelling phase, indeed.

Figure 5: Basic MetropoVis data flow.

The big advantage of the VRML is its wide internet usage and the support of plugins available for common web browsers. Figure 6 demonstrates a part of the MetropoVis case study geometry rendered with the Cortona VRML viewer [Cortona]. A number of drawbacks associated with such plugins prevent them from being a viable choice for large-scale visualization. This is one of the reasons, why we are developing our own visualization tool, capable of real-time visualization of huge amounts of data. Nevertheless it is possible to use Cortona and other similar software to render large virtual cities, though not in real-time and lacking the advanced rendering optimizations and interective features of the AVE (Advanced Visualization Engine) Viewer developed at the VRVis Research Center used in this project.

Figure 6: St. Martin's Cathedral surroundings rendered without textures in the Cortona viewer.

6 PUTTING DIVERSE DATA INTO ONE PIECE

Our system specification proposal naturally follows the modeling database ↔ rendering software division, relieving the database from the real-time visualization and the renderer from the modeling complexity affairs. A simple HTTP protocol was chosen as a communication link between the two subsystems, leading to a particular tradeoff of advantages and drawbacks. The AveViewer acts as an internet application, giving the user a freedom of physical location. Multiple program instances can run simultaneously with one Database Server. On the other hand, HTTP does not guarantee any bandwidth or roundtrip time. Therefore, the AveViewer must be independent from the Database Server, to some extent. In a real setup, the AveViewer tries to load as much data as possible after the connection was established (vital connection persistance is generally not assured over the internet) and after the most crucial data are delivered, less important information is requested in smaller chunks.

The main goal of the Database Server is to unify the data formats provided by its various input sources supplying geometrical models, textures, metadata and non-geometrical information. For this purpose we have developed a standalone application for importing VRML scenes into oure database. Due to the versatility of VRML, all of the currently available wide-spread modeling software packages offer an option to export virtual scenes into this format. However, there are some reasons for choosing other data formats than VRML if the modeling software provides additional features (non-standard object representation, complicated multitexturing, etc.). In this case, we let other developers write their own software for importing data into the Database Servers internal structure, which is published in the interface specifiction document [MetropoVis]. At present, we provide aditional programs for imprting ESRI shapefiles and digital terrain maps.

ESRI files usually serve as a repository for GIS systems, full of data suitable for fast visualization of complex urban scenes. Nevertheless, they often miss one property crucial to photorealistic rendering: color textures. We discovered that software system natively handling with ESRI formats commonly does not allow to store scenes in VRML and hence we introduced our own software solution called Piecer. It is an application capable of combining various vector formats into one VRML scene (prepared for further refinement and texturing in conventional modelers). Additionaly, it allows to quickly break a large city geometry into small pieces (general-purpose modelers are rather awkward in this task), which is very helpful for modeling a large city by a team of modellers (Figure 4 was generated in Piecer as a scene composed of standalone building blocks).

One obvious problem arises when different geometry sources are combined: coordinate systems compatibility. VRML scenes are specified in the well-known 3D orthogonal system, but can differ in orientation, scale and position of the central point. Instead of storing some extra information about the coordinate system for each object, we developed a command-line convertor of VRML file coordinates, that makes it possible to transfer them into a unified system specified by the database administrator and store the transformation in a 4×4 numerical matrix (this useful for transferring the whole database content into the coordinate system of the original VRML scene).

7 DATABASE SERVER OPERATION

The Database Server should store the content intended to fast rendering in the arrangement most appropriate for instant reaction to the AveViewer demands. On the other hand, it should organize data transparently in order to provide multipurpose interface for common GIS analyzis. Core of the Server thus should be a well-arranged spatial database accessible by geometrical, quantitavite, metadata and other SQL queries (most common query language). Main dataflow diagram of our solution is shown in the next figure.

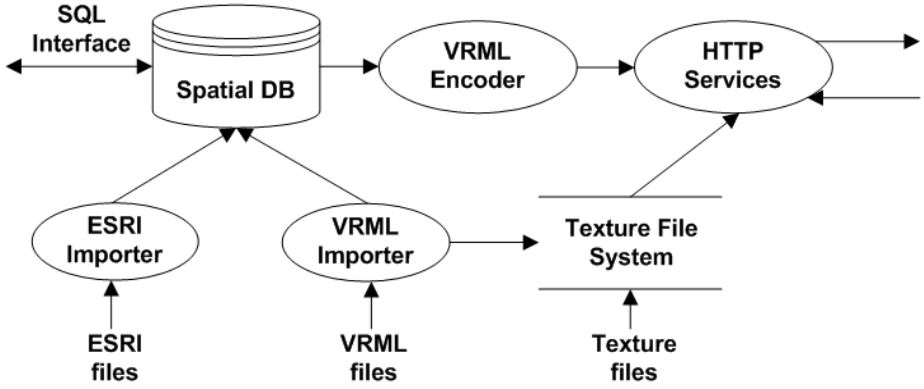

Figure 7: Main context of the MetropoVis Database Server dataflow.

On the input side, the spatial database is fed by the VRML, ESRI or potentially other importing software. As customary in a number of applications, all the texture images are held separately from the database files, which is especially suitable for VRML scenes, since a VRML viewer (not necessarily the AveViewer in this case) primarily loads the VRML geometry file, which refers to the textures by URL strings (www addresses). Thus the textures can be loaded on demand, partially, or not at all, depending on other constraints.

The spatial database design (specified in [MetropoVis]) follows the necessary GIS standards and recommendations for developers (which were summarized in a consise manner by Rigaux et al. [RIGAUX01]). Geomterical structures are stored in hierachical order (scene → layer → object → material → face → vertex) with the single point as the lowest atomical entity. Therefore the objects originally delivered in one piece may be split into separate fragments, if demanded from AveViewer. We have achieved smaller information redundancy in our spatial tables compared to the plain VRML file (where one point joining different materials is divided into more vertices).

Wit the exception of the geometrical and textural information, each object refers to its own metadata record. The most important non-visual metadata item is a timestamp of the (real-world) object creation and possibly its termination (end of existence). Thus it is possible to retrieve the state of each part of the city as it was at a specific time. By loading the data available for whole time intervals (thus requesting an interval in both time and space), it is possible to visualize the changes in the buildings of a city as they happened over time. This time-dependency of the city data is essential for the long-term viability of a city data base.

Other metadata include object specific information (e.g. location name) and author specific information (e.g. name, time of the model creation) information. The same meta data is available for object layers as well(buildings, traffic signs, vegetations, civic networks, etc.). The metadata record specification is open to additions, and a tool for quick specification of a new item has been implemented for simplifying further extensions.

We have already mentioned the time constraint AveViewer can use for specifying the demanded data. Other natural constraint is the space to load geometry from. Apart from the time and layer specification, no other metadata restrictions were incorporated into the DB Server ↔ AveViewer protocol (detailed protocol specification can be found on the project web page [MetropoVis]). Our motive was to achieve a fast communication, which requires static query formats (i.e. the possible query fields are known in advance) rather than dynamic queries that would lead to additional time overhead during message parsing and particularly during the result composition on the DB Server side.

In order to avoid geomtery redundance, we introduced so-called prototyped objects into the network protocol. Prototyped objects (traffic signs, trees,…) are geometry entities with multiple occurences of the same (or proportionally changed) instances. Prototyping is also possible in VRML, but the prototyped object cannot be shared by two separate VRML files, which is not a case of our system. Each prototype has a unique identifier associated and once the AveViewer loads the prototyped geometry, only the instance properties like position and height are transferred with the prototype identifier through the network. Having reasonable classes of trees (oak, pine, maple,…) prototyped, a virtual city creator can save huge amounts when supplying models for parks, orchards or woods.

From the huge nomber of availble choices for a spatial DBMS (database management systems), the famous open-source MySQL in its 4.1 version was chosen as a solution for our Database Server. This version of MySQL is the first to support spatial tables, exactly in the OpenGIS SFS style (defined in [OpenGIS]). MySQL user can now formulate structured geometrical queries for the database, which made it possible to reserve the database connections for non-visual SQL analyzis, as illustrated in the Figure 7. An additional advantage of the OpenGIS spatial tables is the ease of porting the DB Server source code to any other OpenGIS-based DBMS. Due to performance considerations, we are using the C API for interfacing MySQL, rather than ODBC or other slower but DBMS-independent connection.

From the point of view of a computer graphics developer, OpenGIS has one major drawback – it only operates in a 2-dimensional space. The third dimension has to be an attribute of a 2 dimensional object. We overcome this limitation by restricting the AveViewer queries to two dimensions, which is sufficient for queries in a city model. The query results come out as fully 3D files, thus the AveViewer user can freely operate in all three dimensionsm, as well as the time dimension.

8 RESULTS AND CONCLUSION

We have developed a general system for visualizing time-dependent urban data that can easily be extended for visualizing a number of aspects of urban development. The use of standards at various interfaces in our system makes it possible to replace a number of components without a huge effort (e.g. viewer, modeler, database). Our in-house viewer facilitates viewing large city-scapes in real-time at a very high, near phtorealistic quality. In the future we will extend the database side to handle more available metadata, to facilitate the work of urban planners, and provide non-photorealistic data-views that visualize various aspects of the data available in city models.

9 REFERENCES

[FTACNIK04] FTÁCNIK, M., BOROVSKÝ P., SAMUELCÍK M., Low-cost High-quality Virtual City Models, CORP 20004, http://corp.mmp.kosnet.com/CORP_CD_2004/archiv/papers/CORP2004_FTACNIK_BOROVSKY_SAMUELCIK.PDF

[KARNER] KARNER K., MetropoGIS: A City Modeling System, 1st International Conference on Virtual City and Territory, Barcelona 2004, http://www.vrvis.at/TR/2004/TR_VRVis_2004_033_Full.pdf

[DUDEK03] DUDEK I., BLAISE J. Y., New Experimentations of a Generic Framework for Architectural Heritage Data Visualisation, WSCG 2003 proceedings, http://wscg.zcu.cz/wscg2003/Papers_2003/E23.pdf

[VOP01] The Virtual Old Prague Project, http://www.cgg.cvut.cz/vsp/

[VRML97] Web3D Consortium: VRML Archives, http://www.web3d.org/x3d/vrml/index.html

[Cortona] ParallelGraphics, Cortona VRML Client homepage, http://www.parallelgraphics.com/products/cortona/

[MetropoVis] MetropovVis project home page, http://www.vrvis.at/research/projects/MetropoVis/

[MySQL] MySQL 4.1 Downloads, http://dev.mysql.com/downloads/mysql/4.1.html

[RIGAUX01] RIGAUX P., SCHOLL M. O., VOISARD A., Spatial Databases: With Application to GIS, Morgan Kaufmann, 1st edition , 2001

[OpenGIS] OpenGIS® Simple Fatures Specification for SQL, http://www.opengeospatial.org/docs/99-049.pdf