A DATABASE FOR A 3D GIS FOR URBAN ENVIRONMENTS SUPPORTING PHOTO-REALISTIC VISUALIZATION

Michael Kofler, Herwig Rehatschek, Michael Gruber
Institute for Computer Graphics
Graz Technical University
Austria
{kofler,herwig,gruber}@icg.tu-graz.ac.at

Commision II, Working Group 2

KEY WORDS: GIS, Urban, Database, Visualization, Three-dimensional

ABSTRACT

As 3D GISs (geographical information systems) become increasingly important, new database technologies are needed. This paper presents ideas about requirements for a 3D GIS database for urban environments. A high emphasis is given to the support of a fast photo-realistic visualization, thus expanding the scope of functions offered by traditional GIS databases. Required functions to allow an efficient visualization are: • Perspective querying (to extract all objects of the database located a pyramid of vision). • Support of the level of detail concept on database level (to extract objects in varying detail according to the size of the objects on screen). • Image compressing/decompressing/caching (to offer a scale-able compromise between speed and data volume).

KURZFASSUNG

Mit der zunehmenden Bedeutung dreidimensionaler geografischer Informationssysteme (3D GIS) zeichnet sich die Notwendigkeit neuer Datenbanktechnologien ab. Herkömmliche GIS-Datenbanken sind insbesondere da überfordert, wo es um die schnelle Visualisierung der Daten geht. Erforderliche neue Funktionen sind: • Perspektivische Abfragen (um alle Objekte zu ermitteln, die momentan sichtbar sind). • Unterstützung des Level-of-Detail-Konzepts (um Objekte je nach ihrer aktuellen Größe am Bildschirm in unterschiedlichen Qualitätsstufen zur Verfügung zu stellen). • Automatische Komprimierung / Dekomprimierung von Bildern.

1 INTRODUCTION / MOTIVATION

An increasing number of users of GISs for urban environments asks for 3D extensions ([SBM 94], [RG 95]):

- Scientists want to simulate noise, heat and exhaust spreading in big cities.
- Telecommunication companies need 3D data to calculate wave propagation in urban environments.
- Architects want photo-realistic models of existing buildings to plan new ones and to visualize the resulting scenery.
- Tourism agencies want to offer virtual reality models of destinations they are offering.

The step from 2D to 3D causes several problems, which can be summarized in three points: building 3D models, storing them and providing a user interface to visualize and manipulate them. This paper is focused on the latter two points. It discusses techniques to store and visualize huge 3D data sets.

Currently, two research groups are working on this topic: on the one side, maintainers and developers of GISs are trying to expand their 2D systems with 3D features. On the other side, lots of efforts come from computer graphics experts, who want to visualize scenes of growing complexity faster and faster.

Unfortunately, there is little interconnection between these two groups. This paper tries to fill this gap. It gives an overview of recent research activities (section 2), points out requirements to a 3D GIS database for urban environments (section 3) and discusses issues of implementation (section 4). The paper ends with a short outlook (section 5) and some references.

2 RELATED WORK

This section summarizes some recent papers on rendering and storing complex geometrical models. All papers present ideas to improve the LOD (level of detail) concept, which will be crucial for any 3D GIS application supporting interactive photo-realistic rendering. LOD is a mechanism used in computer graphics to improve the drawing speed of complex scenes [Cla 76]: Each object is stored several times in different levels of quality (levels of detail). During visualization each object is drawn in the optimal level of detail. The chosen level depends on the size of the object in the current view. Objects that appear small can be drawn in little detail (and therefore very fast) without loosing quality; in contrast, objects near the point of view that cover a lot of space on the screen need to be rendered in full quality.

Although the LOD idea is quite old and lots of applications make use of it, several problems remain unsolved. The most urgent one is probably building LOD models: although several scene editors already support the LOD concept, none is able to build LODs automatically in good quality yet. Therefore, most existing LOD models are still built manually, thus increasing the already high cost of creating a model at all. [HG 94, SS 95] explain algorithms for building lower levels of detail from a given 3D model in high quality. Until now, these algorithms have been restricted to polygonal models and do not deal with textures, though.

In its original concept LODs offer different geometric representations for one object. [MS 95] propose two improvements: (1), the LOD concept can be adapted to store the entire model (e.g. a city) in one hierarchical structure. (2), 'impostors' replace traditional LODs: Impostors are very similar to LODs; they are more versatile as they include additional

informations such as the view angle. For example, the impostor for the front view of a building might consist of only one rectangle. Although this idea might produce higher quality pictures, building an impostor model is even harder than building LOD models.

[FS 93, Fun 93] describe the Sodahall walkthrough system: This system allows real time movement through a complex building. The visualization is based on the LOD concept. To choose the appropriate level of detail, time constraints are used. If the scene gets more complex during a walk through, the level of detail is reduced accordingly. Thus, a nearly constant frame rate (independent of the complexity of the visible scene) can be achieved. The entire model (350 Mbytes) is stored in a database system especially developed for the Sodahall system. This approach guarantees maximum speed but lacks multi user access, standard queries etc. – features that are indispensable for a GIS.

[KLR 95] describe a system called Virtual GIS, which unifies a textured DTM (digital terrain model) with simple GIS data (buildings). The system uses the LOD idea extensively to minimize the amount of DTM data actually drawn and achieves 'real-time visualization' (defined in the paper with a guaranteed rate of 10 to 15 pictures per second). The system is however limited to simple terrain models with a constant triangle size.

3 3D DATABASE DESIGN

3.1 Relational versus object oriented databases

Currently, most GISs are implemented either using relational databases (RDBs) or using non standardized application-specific databases [Sin 93]. The main reason for this fact is that nearly no commercial object oriented databases (OODBs) were available until about five years ago.

The disadvantages of RDBs become obvious with a close look to the nature of GIS data: Broadly, GIS data means arbitrary data types including numeric and short string data, large unstructured data such as textures, complex structured data such as the 3D geometry of buildings and finally compound objects that are comprised of such data.

RDBs lack the mechanism to deal with this kind of data: Their tabular approach does not allow a suitable modeling of complex hierarchical objects. Although most RDBs support binary large objects (BLOBs) to store graphical data (e.g. textures), these cannot be queried in the same way as other data types. Besides, most RDBs suffer from a severe performance loss if many BLOBs are used.

In contrast, OODBs allows building complex hierarchical data models easily. (Most commercial OODBs offer native support for C++ and allow to store any kind of objects that can be created with C++.) OODBs are optimized to deal with binary data efficiently and offer much better performance in this respect. All benefits of RDBs – an easy to use query language, data distribution, networking, versioning – are still available (although sometimes in a modified form; e.g. most OODBs do not support SQL but improved dialects that offer a better handling of object). Thus, OODBs should be an obvious choice for any new database system for a GIS.

3.2 Amounts of data to be expected

In September 1994 a city block with 28 buildings in Vienna was modeled in a pilot effort ([GMB 95], figure 1). It took about 180 Mbytes to store this scenery with uncompressed photo textures at a pixel size of $4 * 4\,cm^2$. More than 99 percent of this data was needed to store photo textures. First experiments proved that it would be possible to compress textures to about 3 percent of their original size without a significant loss of quality.

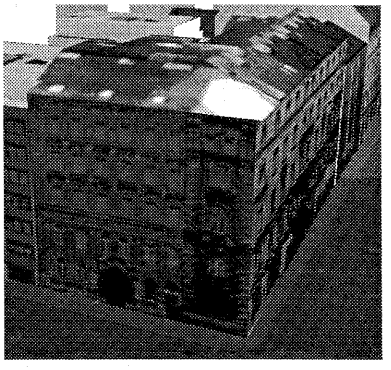

Figure 1: One house of the Vienna pilot project

Since Vienna has 220000 buildings, this data set can be considered to represent $\frac{1}{8000}$ of the entire city. Extrapolating these numbers, using image compression, considering some overhead for indexing in the database etc. it should be possible to store a 3D GIS model of Vienna in 100 to 200 Gbytes.

This first approximation does not consider objects other than buildings (e.g. plants, trees etc.). The interior courtyards of buildings were neglected, too. Another important feature that was dropped in this approximation would be versioning (allowing to view the data at any stage/time since the start of the database). Therefore, it seems a realistic assumption to expect at least 500 Gbytes of data.

3.3 Visualization

During the past ten years a lot of effort has been invested in improving the drawing speed of complex scenes, regarding hardware (z-buffering, texturing etc.) as well as software (e.g. LOD concept). These techniques do not solve the problems that appear when dealing with really big models, though: at the point where the entire scene can not be held in RAM any longer, it is not the rendering the scene which turns out to be the bottleneck, but rather retrieving the 3D data from a database.

To allow interactive working with a big 3D GIS model, it is crucial to minimize the amount of data transferred from the database to the visualization software. It will be a main function of the database to act as an intelligent filter, reducing hundreds of Gbytes of available data to a flow of a few Megabytes per second, that can actually be drawn in realtime. (The exact amount of data that can be handled depends primary on the rendering hardware. One important key number is the number of textured triangles that can be drawn per second.)

3.4 R-trees, bounding boxes and the LOD concept

R-trees have been established as one of the most important data structures for indexing spatial data. Lots of 2D GIS applications rely on R-trees or on one of their variants. R-trees consist of overlapping rectangles that embrace geometrical objects (e.g. all houses of a district). R-trees can easily be

expanded to 3D, now using hierarchically structured boxes instead of rectangles.

In a 3D model, R-trees would be used to store 3D bounding boxes (figure 2): At the root level, one gigantic box embraces the entire city. In this box, several smaller boxes contain districts of the city. Within these city-boxes, smaller boxes might be used to build bounding boxes of streets. The next R-tree level consists of bounding boxes holding buildings or other elementary objects. Finally, the R-tree concept might be used to partition buildings into even smaller objects – roof and body, in the next level windows, doors, chimneys etc. Thus, R-trees are a data structure that is simple to manage and yet allow the storing of data of an entire city in surprisingly few levels.

Notes: The previous paragraph only describes the basic concepts about how R-trees could be used to store 3D graphical data. In reality, some more levels might be useful to limit the number of objects within one level. Please note also that the R-tree is organized in a three dimensional way from the beginning. Of course the z-coordinate is of little interest in the higher levels of the R-trees, as most big cities are more or less flat (e.g. the z-coordinate range is very small when compared to the range of x- and y-coordinates). The z-coordinate gets more and more important in deeper levels of the R-tree, where buildings or other objects are split to sub-objects. This consequent 3D design makes it possible to use the database for objects other than cities, e.g. to build a VR-model of a museum, an airport or of a big shopping center.

As the title of this subsection suggests, R-trees can easily combined with the LOD idea. Each level of the city-R-tree simply *is* a level of detail! The only point where the R-tree-concept must be expanded is the data stored within one box: While traditional R-trees up to the second lowest level only contain a list of sub-boxes (or sub-rectangles in 2D), LOD-R-trees also include some graphical information that is sufficient to visualize the interior of the box in a coarse quality. If this quality is not sufficient for a certain point of view, a more detailed graphical description can be obtained from the sub-boxes within the current box.

3.5 On our flight to the Stephansdom

The best way to explain this concept is an example: imagine, you are flying in a helicopter towards Vienna. Suddenly, getting around some hills, you see Vienna for the first time. You are still quite far away – thus Vienna is only a more or less flat area covered with buildings and plants and divided by the Danube. To visualize Vienna from this viewpoint, a very simple DTM-model of Vienna with some coarse textures of aerial images is sufficient. This data is stored in the outermost R-tree-box; there is (yet) no need to traverse the R-tree to any deeper level.

As you slowly come nearer, some outstanding objects – perhaps the radio tower, the UNO city building complex or whatever – must be visualized in more detail to preserve a photo-realistic quality. The graphical data needed for rendering can be obtained from the R-tree on the next deeper level. On your flight to the center of Vienna, this process will constantly be repeated, getting exact graphical models of objects near the helicopter out of lower R-tree-levels and less exact models of objects far away form higher R-tree-levels. The amount of data needed to draw the entire scene will stay approximately stable; objects near the viewpoint will contribute most of this

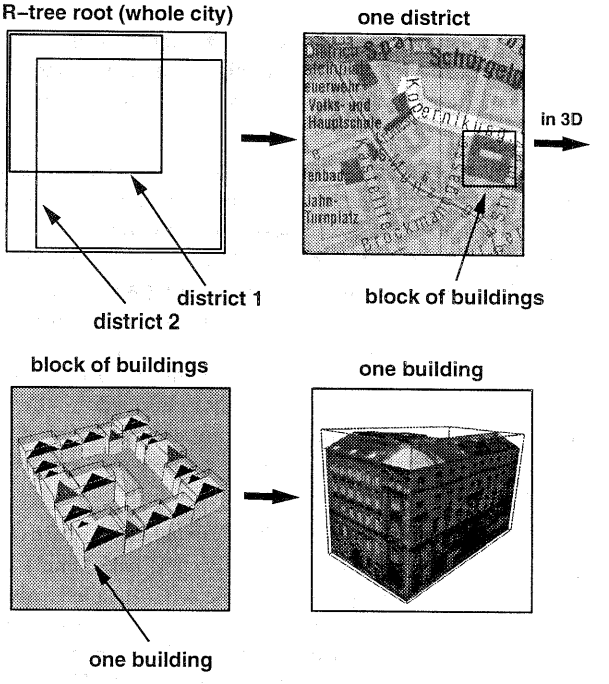

Figure 2: Examples for some R-tree levels (all levels are 3D, although some are shown only as 2D projections)

data.

When you finally land in front of the Stephansdom (a big cathedral in the central city), this extraordinary building will be drawn in the best possible quality. While standing and staring at the picture it will improve progressively as more and more data in growing levels of detail is loaded from the database and used to refine the picture. Technically speaking, you are now in at the bottom end of a very small sub-tree of the entire R-tree of Vienna.

3.6 3D spatial access, perspective querying

As the data structure is in a high degree hierarchical, spatial queries will be organized in a hierarchical way as well. Searching for an object in 3D-space means traversing the R-tree. As the whole tree is based on bounding boxes, a few small queries in each level of the R-tree are sufficient to find objects in 3D-space. (Small means that each box of the R-tree only contains a rather small number of objects (sub-boxes), e.g. all houses of one (part of a) street.) Searching in 2D R-trees for a point or for all objects within a rectangle or box does not impose any difficulties.

What is really new when compared to traditional R-trees applications is querying for all points within a pyramid or cone of vision. It will be that kind of query that will be needed most frequently while visualizing a scene. A straightforward realization of a perspective query would result in complex geometric formulas that cannot possibly handled by any query language however sophisticated. Besides it would result in a huge number of objects retrieved, although most of them would contribute little information actually needed for visualization.

A more intelligent scheme is needed. To simplify the query the pyramid of vision can be split into axially parallel boxes. This splitting has an obvious disadvantage: it results in a con-

siderably higher volume, as the boxes need to fully embrace the pyramid of vision. On the plus side, queries for objects within the resulting boxes can be done much faster – all you need is simple bounding box tests. Besides, it is possible to process queries for different boxes in different ways (figure 3): The first query would only consider the box nearest to the point of view. With the resulting objects a first visualization can already be done. In the meantime, objects for boxes far away from the point of view can be queried. As these objects distribute less information (foreshortening), the query can be stopped at a higher level in the R-tree. (This is equivalent to a lower level of detail.) Thus, perspective query does not mean simply retrieving all objects within a pyramid of vision; the LOD concept is also involved – otherwise its implementation on database level would not make sense.

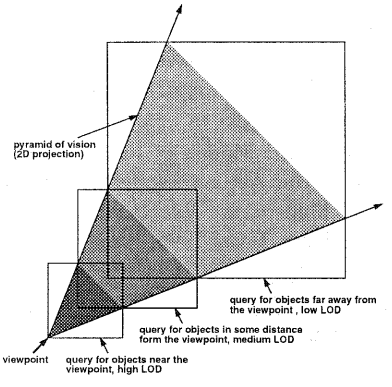

Figure 3: Successive queries for objects within the pyramid of vision; the figure shows a 2D projection of the 3D pyramid and its bounding boxes

3.7 Handling textures

As was mentioned above, in a first pilot project, uncompressed photo textures consumed more than 99 percent of the models memory. Simply compressing this textures with JPEG or another algorithm results in considerably smaller memory demands. To avoid a tradeoff in speed, the textures used most recently should be cached uncompressed by the database. A simple LRU (least recently used) algorithm together with a variable cache size limit might be used to decide which bitmaps may stay uncompressed in the cache.

To support LOD, textures will be stored as multi resolution bitmaps (bitmap pyramids). As the cache size is always a limiting factor, it is neccesary to cache bitmaps in a low LOD if memory restrictions do not allow to cache them in higher quality. This would allow a fast visualization at low quality. If higher quality is needed, compressed bitmaps need to be reloaded and decompressed again.

4 IMPLEMENTATION ISSUES

4.1 The CyberCity project

The above described database will be one cornerstone of the CyberCity project. In this project the Institute for Computer Graphics in Graz is developing techniques to store, manipulate and visualize three dimensional data of urban environments. Important research tasks apart from the database are (semi-) automatic 3D object reconstruction from available 2D GIS data and aerial images, texture extraction from aerial images and facade photographs [GPL 95], object refinement (geometric modeling) from facade photographs, development of a line camera to do facade photographs more efficiently [MD 96] etc.

4.2 Software and data format standards

Of course it is tempting to restart software development from scratch: This would result in an optimal performance and in software modules fitting together. However, this would also mean spending years of manpower to write code that has already been written in a similar form, defining another data format incompatible to all previous ones. To avoid these drawbacks, CyberCity will rely as much as possible on existing standards:

- Graphical data as well as GIS data will be stored using a commercial object oriented database system (possibly ObjectStore or O_2).
- The database will be able to import and export Open Inventor files. Therefore, any program supporting Open Inventor can be used for modeling or for visualization.

Although the development of the database has a GIS application in mind, the database might also turn out to be a good starting point for a more general VRML database.

4.3 Current work

At the moment, the OODBs ObjectStore and O_2 are being used for experiments in perspective querying. A test program to benchmark bounding box queries in an LOD-R-tree already exists (figure 4). No final decision has been made yet about which database system to use, though.

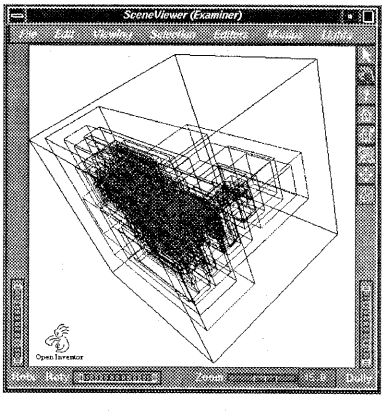

Figure 4: A test database consisting of 3000 empty bounding boxes organized in a r-tree was queried for cubes in the front octant. The result was visualized using Open Inventor.

At the same time, software is being developed to build a low-detail model of Vienna (figure 5). The model is based on a manual analysis of aerial photographs of Vienna. For each building or building complex, a polygon defining the eaves and one base point is available. Out of this data, a converter builds 3D objects for each building and digital terrain model (DTM). Incorrect objects (e.g. duplicate points, intersecting polygons etc.) are either corrected or skipped. A test dataset consisting of more than 10000 buildings will be used to test the first implementation of the database.

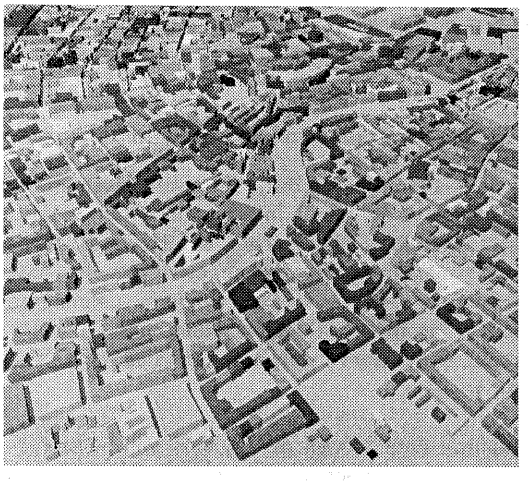

Figure 5: A low-detail model of approximately 2000 buildings in Vienna

5 SUMMARY / OUTLOOK

This paper outlined a concept for storing big 3D datasets of urban environments. Special focus was given to fast visualization. It was proposed to organize the data as one big LOD model. For any part of the data geometric models in several quality levels can be used to render the scene.

On the database level, the LOD model can be stored in an R-tree. The nature of the data (hierarchical organization, mainly geometrical or binary data) makes the choice of an object oriented database obvious.

The implementation of this system is the first step toward a complete database system for a 3D urban GIS. The next step would be a graphical user interface which allows interaction with the data.

5.1 Building an urban Geo-Server

To make the data available to a geographically dispersed community, techniques to distribute the data between several database servers and to update datasets over the network need to be developed. The implementation could be based on the concept for a distributed central / local server concept proposed by [Reh1 96]. Although this server (GDSS, Graz Distributed Server System) is intended to achieve a unified access to remote sensing image data, there are more or less the same requirements as for the Cybercity project: network managment, transparent data exhange and database queries, user interaction etc.

To achieve acceptable response and transmission times, a high speed network backbone (probably ATM) must be used. ATM will most likely become an integrationg network in the near future. It is designed to carry all existing and future data services including traffic with a time relation – e.g. voice and video – as well as data with transmission speeds up to 2.4 Gbit/sec [Pry 94, Reh2 96].

Finally, the entire dataset would be stored only at one or a few central servers; reduced datasets could be stored locally, e.g. all data of district Xyz in the municipal administration office Xyz, all data about telephone lines in the office of a telecommunication company and so on. This concept of an urban Geo-server would lead to a new dimension of GIS application.

REFERENCES

[Bro 92] Brooks F. et al: Six generations of building Walktroughs. Technical Report TR92-026, University of North Carolina, Department of Computer Science, 1992.

[Cla 76] Clark J.H.: Hierarchical Geometric Models for Visible Surface Algorithm. In Communications of the ACM, 19, 10 (October 1976), pp. 547-554.

[FS 93] Funkhouser T.A., Séquin C.H.: Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. In Siggraph 93 Computer Graphics Proceedings, pp. 247-254.

[Fun 93] Funkhouser T.A.: Database and Display Algorithms for Interactive Visualization of Architectural Models. Ph.D. thesis, University of Berkeley

[GMB 95] Gruber M., Meissl S., Böhm R.: Das dreidimensionale digitale Stadtmodell Wien. Erfahrungen aus einer Vorstudie. In: VGI (Österreichische Zeitschrift für Vermessung und Geoinformation) 1+2/95, pp. 29-36

[GPL 95] Gruber M., Pasko M., Leberl F.: Geometric versus Texture Detail in 3D Models of Real World Buildings. In: Proceedings of the Ascona Workshop 95 on Automatic Extraction of Man-Made Objects from Aerial and Space Images. Birkhäuser, Basel 1995.

[HG 94] Heckbert P.S., Garland M.: Multiresolution Modeling for Fast Rendering. Proceedings of Graphics Interface '94, Banff, Alberta, Canada, 1994.

[KLR 95] Koller D., Lindstrom P., Ribarsky W., Hodges L.F., Faust N., Turner G.: Virtual GIS: A Real-Time 3D Geographic Information System. GVU Technical Report 95-14, Georgia Institute of Technology, 1995.

[MD 96] Maresch M., Duracher P.: The Geometric Design of a Vehicle Based 3 Line CCD Camera System for Data Acquisition of 3D City Models. In.: Proceedings of ISPRS, Vienna 1996.

[MS 95] Maciel P.W.C., Shirley P.: Visual Navigation of Large Environments Using Textured Clusters. 1995 ACM Siggraph Symposium on Interactive 3D Graphics, pp. 95-102

[Pry 94] Prycher M.: Asynchronous Transfer Mode. Prentice Hall, 1994.

[Reh1 96] Rehatschek H.: Preliminary Design of a Distributed Planetary Images Data Archive Based on an ATM Network. In: Proceedings of the Visual 96 Congress, pp. 68-78, Melbourne 1996.

[Reh2 96] Rehatschek H.: A concept for a Network-Based Distributed Image Data Archive, Proceedings of ISPRS, 1996.

[RG 95] Ranzinger M., Günther G.: Changing the city: datasets and applications for 3D urban planning. In GIS Europe, 3/95, pp. 28-30

[SBM 94] Schneider B., Borrel P., Menon J., Mittleman J., Rossignac J.: BRUSH as a Walkthrough System for Architectural Models. IBM Research Report RC 19638, 1994.

[Sin 93] Singer C.: Relationale Datenbanksysteme als Basis geographischer Datenbanken. In Geo-Informations-Systeme 6/1993, pp. 9-16

[SS 95] Schaufler G., Stürzlinger W.: Generating Multiple Levels of Detail from Polygonal Geometry Models. In Proceedings of 2nd Eurographics Workshop on Virtual Environments, Monte Carlo, Monaco, 1995.