Virtual Campeche: A Web Based Virtual Three-Dimensional Tour

Jiri Zara and Bedřich Beneš and Rocio Ruiz Rodarte
CTU Prague,
ITESM, Campus Ciudad de México
ITESM, Campus Estado de México
zara@fel.cvut.cz
{bedrich.benes|caruiz}@itesm.mx

Abstract

We present a web-based application that allows a user to, walk through, see, and interact with a fully three-dimensional model of an old Mexican city. The city itself motivates this work. Campeche is of enormous historical value and has been placed by UNESCO on their list of the World Cultural Heritage sites. A virtual visit can benefit both tourists and scientists.

We show how the model was created and how certain technical issues were addressed. The application runs interactively over a modem-based connection. It first displays primary visual clues and refines the details later. A data prefetcher loads the potentially visible data in advance. The system loads data from the neighborhood by which the user passes using the VRML proximity sensors. In such a way it loads the local neighborhood of the location of the visitor in advance. The application is context-sensitive and can display additional textual and image information.

1. Introduction

Implementing a virtual tour through a model of a city on the web has become fashionable. Even so, there are many problems to be solved. We describe our experience in creating Virtual Campeche; an old city located on the Yucatan Peninsula (see Figure 1). Campeche has great historical value and was declared a World Heritage site by UNESCO in 2001 [11].

There are many reasons for creating virtual tours. The most important is tourism as visitors can previously learn about possible vacation destinations. Another reason is related to business: people can be interested on business, commercial or real estate investment opportunities. Certainly one of the most important aspects is related to culture heritage preservation and our commitment with future generations. This aspect as well as Campeche's remoteness, justifies our goal. The project joins the effort of local authorities' preservation task, by presenting Campeche's distinctiveness in a unique way. By strengthening the identity of the city, we help to preserve its legacy since Culture is an intangible value that is not specifically preserved solely throughout building restoration.

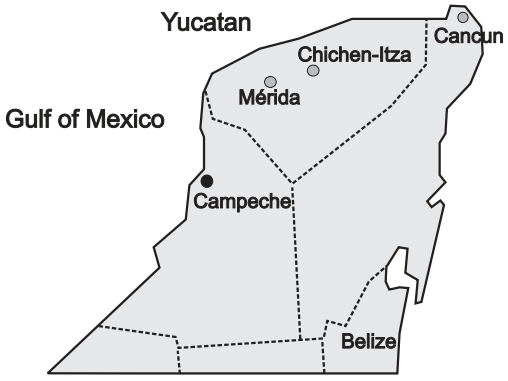

Figure 1. Campeche is located on the Yucatan peninsula on the Gulf of Mexico

Campeche was founded by Spanish conquerors in 1540 at an ancient Maya settlement called Kin Pech. It became the most important port of the Royal Spanish Navy but also the target of numberless pirates attacks for almost a hundred years. To protect the village from looting and destruction, the king of Spain ordered the construction of ramparts around the city. Twenty years of construction were needed to finish a hexagon fortress and it's 2.5 km of walls surrounding the village. Seven bastions were strategically positioned along the six meters high and 3.5 meters thick walls. Ramparts were extended into the sea so ships could dock safely inside and two more forts were built outside the city. Campeche became the best-guarded port in America.

Proceedings of the Fifth Mexican International Conference in Computer Science (ENC'04)
0-7695-2160-6/04 $20.00 © 2004 **IEEE**

Today, some original constructions are gone but the bastions, forts and long sections of the wall are intact.

In our virtual model we have focused on important historic objects: the fortress and the center of the city. We provide a fully three-dimensional virtual tour that can be experienced on any operating system and even runs fairly fast over a modem connection. There is no need to install special software; a common VRML viewer embedded into a standard web-browser is sufficient. We have tested the entire system with the Cortona VRML Client Version 4.2 [5] on different platforms.

The paper is organized as follows. Previous Work, presents different types of virtual tours created for web. Section 3 explains the way we have obtained rough data. Its processing is the topic of the following section. Section 5 describes how we constructed the three-dimensional models. Sections 6 and 7 describe web-based rendering, data structuring and data transmission over the network. The last two sections describe implementation, results, and the final conclusions.

2. Previous Work

There are many virtual cities on the web; Glasgow [12], New Orleans [13], Prague [14], Paris [15], Sydney [16], and Toronto [17], among others. They were created in different ways and can be classified as purely two-dimensional, semi-three-dimensional, and fully three-dimensional.

Two-dimensional models are based on interactive maps and are the most common. An image, textbox, sound, another web-page, or a web-cam can be launched by clicking on sensitive maps. These systems usually provide a search engine to find places of interest.

Semi-three-dimensional representations based on Apple's QuickTimeVR technology [6] are, surprisingly, the most common type of virtual tour on the Internet. These are panoramic views made from realistic renders or digitized photographs which are stitched together for a complete 360 degree view. Users interactively scroll the panorama with their mouse and keyboard to control their viewpoint. QuickTimeVR supports sensitive zones, nodes or hot spots, which allow the user to connect to another view or launch an external application. For example, by clicking on a door, the user calls another QuickTimeVR panoramic cylinder, which produces an effect of passing through that door. QuickTimeVRs are easy to create and do not require advanced technology or programming. Though, their results are limited: by adding connection nodes, the overall weight can increment too much; perspective distortion can be disturbing; scrolling the image provides only a limited sensation of three-dimensional interaction and immersion.

Fully three-dimensional tours are not common. In fact, we know of only Virtual Paris [15] that offers a walk through over a couple of streets and Virtual Old Prague [14] that is probably the first running complex virtual city on the web. There are many issues which make them hard to implement.

Our model of Campeche is fully three-dimensional and is a simplified version of the Virtual Old Prague project [14, 18]. It uses three-dimensional VRML [10] textured models, static photos, web-cams, sounds, and animations.

Common requirements for a virtual city include a realistic appearance, a list of available views and a route planning.

The most important requirement is a fast system response. To provide this functionality our model is divided into small chunks of data that are rapidly moved over the Internet and delivered to the browser. Each data fragment holds additional semantics and allows for progressive refinement of the scene. The model is unbounded and can be scaled depending on the quantity of transferred data, network speed, and rendering quality. [19].

Figure 2. The parts of Campeche that have been included in our model in the first stage

3. Data Acquisition

To reconstruct a city we needed photos of each building taken with the same background light e.g. in the late morning when the sun does not cast large shadows. They were taken from an orthogonal position with no perspective warp. However, at this hour the city was busy so each shot had to be edited to eliminate cars and people. The process is straightforward but monotonous and we needed several visits of Campeche.

Figure 2 shows the part we model in three-dimensions. Actually we have photos of more than 50% of the city center consuming more than 700 MB. We now have three-dimensional models of about 35 % of the city center streets.

4. Object Reconstruction

While data acquisition was laborious, object reconstruction proven even more so. Human intervention was required in all steps although some could be automated.

First, the shots obtained by a non-calibrated camera were cleaned and the important parts selected. This can be partially automated, but human assistance was always needed. The shot histograms must be more-or-less equal and sharpness must be similar. Also, a perspective correction warp that must be applied to the majority of the obtained photos.

Corresponding images were then glued using software for panoramic photographs. One good option was the MGI Photovista [4], but there are others.

Once the shots were ready, the objects were reconstructed into three-dimensional models. It was clear that automatic object reconstruction from a set of photographs is really difficult and that best approach was human-assisted. The automatic approach works only in well defined conditions, e.g., lights are constant and known, if the intensity of the light does not vary, or if two perpendicular walls of the same house are to be visible on the photo.

Figure 3. The two images (up) were used to reconstruct the scene (down)

From various commercial photogrammetric systems we should mention Photomodeller [9] and the University in Berkeley Facade project [8] that is inspired by the successful commercial product Canoma by MetaCreations. These methods need a user assistance. There are other methods that do not require user's assistance, but they produce huge amounts of data (triangles, vertices, textures). Even worse, the data is non-organized (polygon soup) and it is difficult to overlay hyper-links and semantics. The most general way, and the way we used, is to download the models into 3D Studio Max software and make adjustments by hand.

Figure 3 shows a set of images and the resulting three-dimensional VRML model.

As mentioned above, object editing is a monotonous work. To minimize it we have developed a special editor (a snapshot is shown in Figure 4). The main purpose of the editor is to load a VRML model and associate and edit textures on its surfaces. The editor also permits defining levels of detail (LODs) of the model, and assigning levels of compression associated with the textures. The model can be saved in the VRML format for later use in the system.

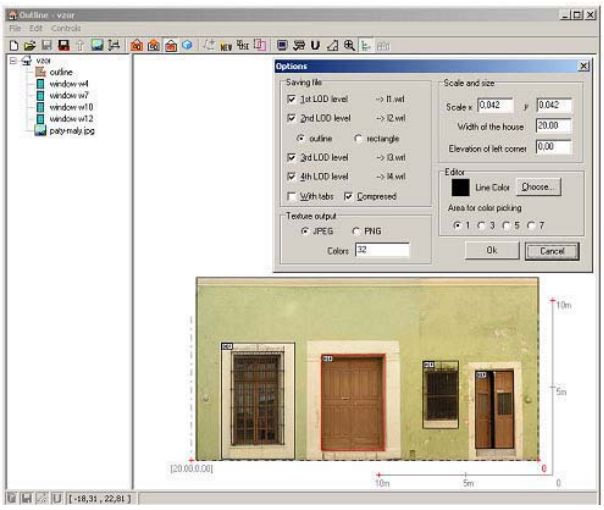

Figure 4. Our VRML texturer

5. Urban Models

While there are numerous papers on large visual databases, this web-based application displays special features:

- Data is structured into small chunks for rapid delivery over the Internet,
- progressive LODs are available,
- a set of impostors is precomputed,
- potential areas of visibility are calculated before implementation,
- the system is open to changes by non-programmers,
- search engines provide historical data, objects names and other background information.

Experience has shown it is not efficient to store the entire structure as one VRML model [19]. Instead the scene is stored as a topological structure whose elements are located

in separate database. See Figure 5. The model is generated on demand by a script that creates chunks of data that are sent over the Internet to the client's browser.

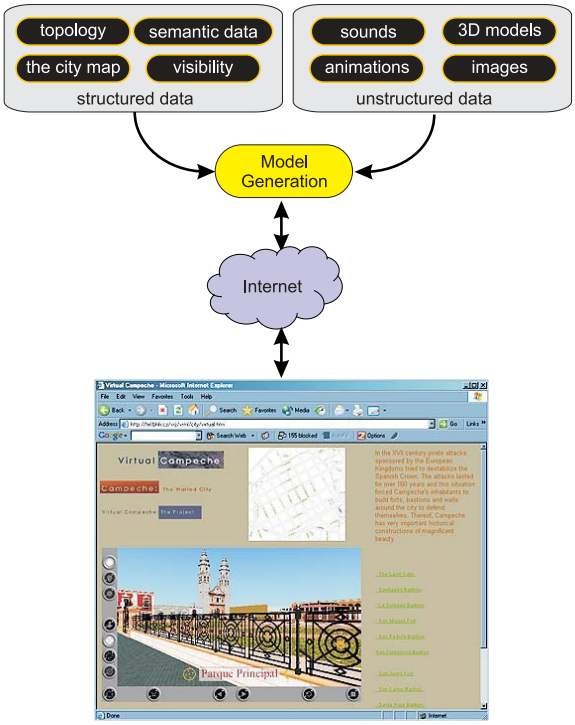

Figure 5. The logical and the topological data structure

Data is stored in either a structured or unstructured format. The structured data reflects the plan of the city. This is further divided into topological information, representing knowledge about the street layout, house neighborhood, etc. The potential visibility areas are stored and, on the top of this, there is semantic information for searching, route planning, etc. All structured data has pointers into the unstructured data.

The unstructured data stores three-dimensional VRML models, photographs, textures, sounds, and web texts. The three-dimensional models are represented in different LODs as explained in the following sections.

6. Rendering

Carefully precalculated scenes allow for efficient rendering. The goal is to immediately display the clues that are visually important, and then add details i.e. display first object silhouettes, then textures, and then less important details. The principle of this approach is displayed on the Figure 6.

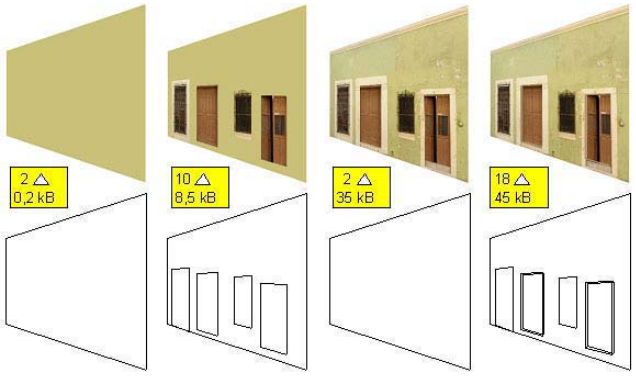

Figure 6. A model of a house in Campeche represented in four LODs for web-based rendering

Dividing the scene into small chunks of data makes this approach possible. Each object is represented in progressive LODs that are sent by the server when the scene is required.

The entire system is stored in hierarchies. The city layout stores the maps together with potentially visible regions (see in the next section). The data structures that represent the streets keep lists of indices that point to the unstructured data. These indices represent the objects that are located on the street. When the user enters the street the objects included in the cell and the adjacent potentially visible cells are sent to the viewer with increasing details.

Textures of door, windows, and some visually important parts are transferred in one composite image from the server. LOD2 is therefore made of a VRML file and a composite texture.

Example in the figure 7 shows several progressive steps of scene as seen over very slow Internet connection.

This display technique is not optimal. The screen image is refined as the user moves and this causes blinking of object textures and object popping. While both can be disturbing, this is the price for using standard hardware and software. Any advanced solution, based on the progressive meshes [3], would involve client side software. We declined this option in favor of a simple implementation that runs everywhere.

7. Data Transmission and Visibility Issues

Special attention has been paid to the optimal balance between the visual quality and amount of data transferred over the web. The LOD principle helps to achieve not only a good rendering speed but allows sending data progressively. This can be seen in Fig. 5 where a number of triangles

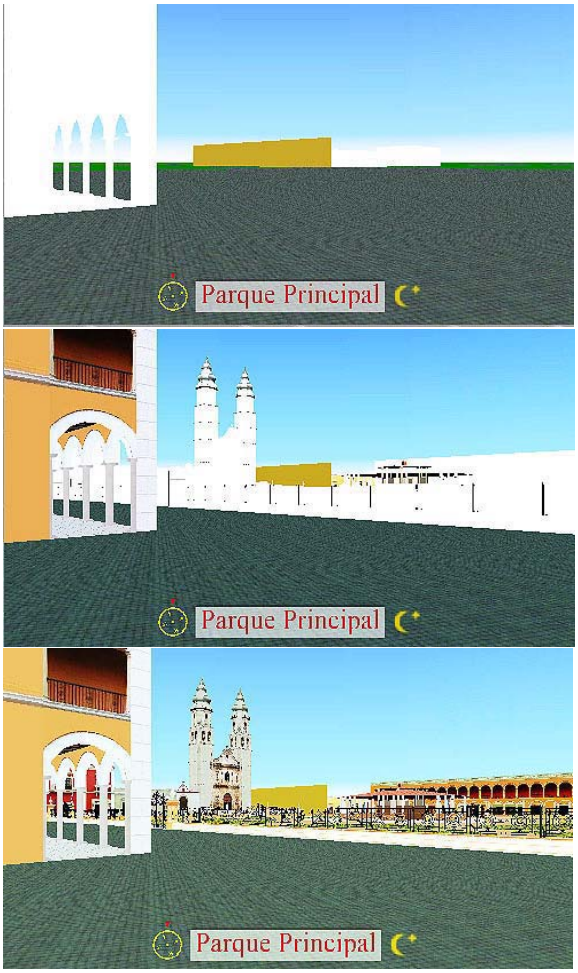

Figure 7. The progressive scene display of Campeche's main plaza (Zocalo)

together with geometrical and texture data for a common building are shown. However, VRML language does not support smooth mesh simplification and creation as known e.g. from [3] and thus unpleasant popping effects are visible. One way to achieve a better appearance would be to set key distances (in which details are switched) to high values. Consequently higher demands to a rendering subsystem would slow down the virtual tour. The traditional tradeoff between a speed and a quality in computer graphics is thus solved in virtual reality - speed is given the clear advantage.

7.1 Data Transmission

Since Virtual Campeche is a web-based application, it was necessary to consider web specific features, namely a stateless communication between a server and a client. A client is responsible for maintaining user's activities. It checks user's position and orientation in a city and is able to recognize that new data has to be downloaded. A proper selection of new data chunks can be computed either by a server or a client. Two different approaches were studied:

1. A client sends a user's position to a server together with a list of already downloaded parts. The server decides what data are missing and sends them to a client.

2. A client has the full information about neighboring and visible parts around a user. It prepares a list of required data and requests them from the server via standard `http` protocol.

It is clear that the first approach increases a load on the server, and is unacceptable for our purposes. We have chosen the second method, where the client has to evaluate topological and visibility information independently on the server. Still, one important question remains - should the client know this kind of information about the whole city in advance or would be possible to build it incrementally as a user walks through the town? In another words - should the complete topology and visibility graphs be transferred to a client in an initial phase (thus causing a delay before the first image is rendered) or could they be split into small subgraphs and transferred part by part to a client? The second technique requires more elaborate organization of data structures but is highly scalable. On the other hand, a client cannot perform advance operations like route planning if the topology graph is incomplete.

In the current version we prefer to keep the initial time to minimum. We prefer to display some data, even in a raw form, rather than to keep the user waiting. The necessary information on neighboring and visible parts is subdivided and attached to every part of a city. Such small data structure could be downloaded from the server as a whole at the beginning or just in time when a user requests a route finding function. To solve the problem of finding a route between places of interest, we plan to derive a simplified topology graph from the whole city map that will include important places only.

7.2 Potential Visibility

As stated before, a geometrical model of a city is combined with structured data. The visibility is not computed in a real-time, but in a preprocessing stage. Since the visibility information is coupled with a specific cell of a city

(e.g. a street, a square), we are interested in computing so called potentially visible sets (PVS) instead of correct evaluation of visible objects from given user's position in a city. This approach is known as from-area visibility [2] or eye-to-cell visibility [7]. The method we use is based on line sampling. Lines originated in random places in given city part are tested against other parts in randomly generated directions.

Virtual users can walk only through the city streets, the options for flying are implicitly disabled in the VRML plug-in and that facilitates the data preparation significantly. We do not need to get precise textures of roofs (simplified by a predefined texture) we can avoid some invisible details.

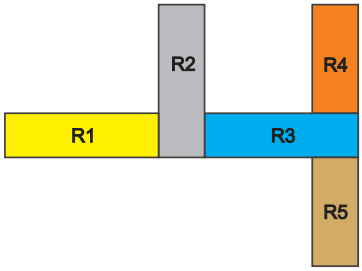

Figure 8. Example of the potential visible scene construction

The city layout is divided into cells (sometimes called regions or blocks). A cell is an area where the visibility does not change significantly. Each cell keeps the list of indices of the objects that are inside. At the moment we enter the cell the list of actually visible and potentially visible objects is generated and the objects are transmitted over the network in different progressive LODs.

Every cell has a precalculated list of visible cells as displayed in the figure 8. This list is stored in the structure of directories on the server, but could be also stored in a global table that could be available all the time. As the user walks he or she is switching from cell to cell and the list of potentially visible objects changes as well.

Let's follow the example from the Figure 8. Let's describe the topology of the cells by the visibility Table 1 that has the following form:

cell	visible neighbors
R1	R2 R3
R2	R1 R3
R3	R1 R2 R4 R5
R4	R3 R5
R5	R3 R4

Table 1. An example of visibility cells

Suppose the user is located in the $R1$. The system will transmit the objects from the cells $R1, R2$ and $R3$. If the user moves to the cell $R2$ the visibility will not change. At the moment the user moves to the $R3$ the system must send all the cells $R1, R2, \ldots, R5$. The important thing is that there the worst case is that the entire scene is transmitted. The average amount of data in reality is reduced to less than 10% as shown in [19].

There is an unfortunate consequence of this approach. The system does not keep any information about the client's status. If this would so, we could know what part of the information that the client already keeps. In this case we could reuse already stored information. VRML does not permit any direct access to the internal structures of the browser. Since we do not want to develop any application that should be installed and we want the system to use the standard browsers and VRML plug-ins, we depend only on the system caches. According to our experience, the cases of rotating and going back are not problematic and the cache hit/miss rate is good if the user moves locally. It would be interesting to optimize the system for these cases.

Applying the static potential visibility approach helps to keep the bandwidth low. Another way how this could be reduced is using billboards and impostors [1]. If the user is far from some objects, they could be replaced by a texture. The large distance will assure there is no, or small, distortion. The texture should be replaced as the user moves closer. The entire set of billboards should be precomputed in advance for the scene.

The data prefetching loading is done by means of VRML Proximity Sensors [10]. Each cell is encompassed by a cube with axis aligned to the coordinate system. Every entrance/exit to the proximity sensor causes loading/releasing of predefined data packages. The cubes are intentionally overlapping so the data is loaded in advance.

8. Implementation and Results

Virtual Campeche was motivated by a successful web application Virtual Old Prague [14] but has been technically simplified. High number of components used in the original application (HTML, Java, PHP, mySQL, JavaScript) lead to a platform dependency and could only be viewed on line.

Virtual Campeche uses a minimal set of technologies/components with the goal of running mostly off-line, e.g. from a CD ROM or a DVD. We avoided Java applets that are not very stable when connected to VRML browser through an EAI interface. Thus only three technologies remained: HTML, VRML, and JavaScript.

The Table 2 presents the data available for a web visitor in the current version. The city is subdivided into 134 parts that correspond naturally to streets and squares.

Virtual Campeche is captured in a reasonable size that is

Media	Number	Size [kB]
HTML text	31	84
Images used in HTML	52	1702
VRML models	137	250
Textures	161	2105

Table 2. The amounts of data used

acceptable even for users connected to the Internet via modem. It should be stressed that we processed 700 Megabytes of raw input data coming mostly from photographs. Virtual Campeche represents not only a qualitative jump from static images to interactive virtual world but a significant compression of information that still preserves the unique atmosphere of the real Campeche.

Figure 9. Real photo (up) and a screen snapshot

We have tested the system in various configurations and environments. The fastest response times occurred when the system was installed locally. Response times were immediate and the inevitable popping of the objects was small.

A remote implementation was also tested. The system is on the broadband Internet in Prague and was tested from Mexico. We have two connections, one is 100Mbs^{-1} and the other was 48kb^{-1} modem connection. The response of the first case was excellent and the system was perfectly dynamic. The modem connection was significantly slower but usable. The primary system feature is that there is continual action and the user is never waiting for a response. The worst case occurs when walking among white walls with successive texture loading.

So far we have implemented the main park, cathedral, Zocalo (center of the city), and the neighboring streets. The system has sensitive areas providing textual information about the place where the user actually stands. A connection with web-cam should be available soon. Figures 9-12 show snapshots from a virtual visit.

Figure 10. Screen snapshots

Figure 11. Screen snapshots

9. Conclusions and Future Work

We have presented the way we have modeled and reconstructed a section of the historical center of the city of Campeche. The result is a three-dimensional system that runs efficiently even over slower modem connections.

Especially labor intensive was the 3D-modeling of objects from photographs. Many parts of Campeche are still to be completed.

In the future we plan to add background data for each object e.g., house names, their history, hotel booking info, telephone numbers, and web-pages.

Figure 12. Screen snapshots

We also plan to enhance the virtual city by videos and web-cams.

There will be technical challenges. We are exploring an aerial view of the scene that will allow the user to move within the model by simple clicking, and also indicate their new coordinates.

Virtual Campeche is part of a cultural project for the National Institute of Anthropology and History, which deals with digital inventory and virtual museography. Some bastions of the fortress are now museums that exhibit Mayan artifacts which are currently being digitized. Within Virtual Campeche, Internet visitors will be able to enter a bastion-museum and interactively admire three-dimensional VRML reproductions of archeological artifacts.

10. Acknowledgments

We are grateful to the students of the Czech Technical University in Prague that handled data and photography processing, and who implemented the VRML parts of the project. Special thanks go to Rostislav Bundil, Martin Hroník, Petr Zobal, Pavel Mařík, and Miroslav Michálek. Thanks also to student Elena Reyes from ITESM CEM in Mexico who made the web pages where the VRML world is displayed.

We deeply appreciate the comments of the second referee of this paper.

This work has been partly supported by the Ministry of Education, Youth, and Sports of the Czech Republic under research program No. Y04/98: 212300014 related to information technologies and communications.

References

[1] T. Akenine-Moller and E. Haines. *Real-Times Rendering, second edition*. A K Peters, 2002.

[2] J. Bittner, P. Wonka, and M. Wimmer. Visibility Preprocessing in Urban Scenes using Line Space Subdivision. In *Proccedings of Pacific Graphics (PG'01)*, pages 276 – 284, 2001.

[3] H. Hoppe. Progressive Meshes. In *Siggraph Conference Proceedings, Annual Conference Series*, pages 99–108. ACM Siggraph, 1996.

[4] MGI Photovista 2.0. www.mgisoft.com.

[5] Parallel Graphics, Cortona VRML Client. www.parallelgraphics.com.

[6] QuickTime VR. www.apple.com/quicktime/qtvr/.

[7] M. Slatter, A. Steed, and Y. Chrysanthou. *Computer Graphics and Virtual Environments: From Realism to Real-Time*. Addison Wesley publishers, 2002.

[8] C. Taylor, P. Debevec, and J. Malik. Reconstructing Polyhedral Models of Architectural Scenes from Photographs. In *Proccedings of ECCV'96*, pages 659 – 668, 1996.

[9] The PhotoModeler 3D Modelling Tool. www.photomodeler.com.

[10] The Virtual Reality Modeling Language. ISO/IEC 14772-1:1997.

[11] UNESCO World Cultural Herritage. whc.unesco.org/heritage.htm.

[12] Virtual Glasgow. iris.abacus.strath.ac.uk/glasgow/.

[13] Virtual New Orleans. www.planet9.com/earth/neworleans/.

[14] Virtual Old Prague Project. www.cgg.cvut.cz/vsp/.

[15] Virtual Paris . www.2nd-world.fr/.

[16] Virtual Sydney. www.planet9.com/earth/sydney/.

[17] Virtual Toronto. www.intoronto.com/.

[18] J. Zara. Concise tour to the virtual old prague. In *Proceedings of the EUROGRAPHICS 2002, Short Presentations*, pages 191–198. Eurographics Association,, 2002.

[19] J. Zara, P. Chromy, J. Cizek, K. Ghais, M. Holub, M. Mikes, and J. Rajnoch. A Scalable Approach To Visualization of Large Virtual Cities. In *Proceedings of the Fifth International Conference on Information Visualization*, pages 639–644. IEEE Computer Society, 2001.