Secure and Fast Urban Visualization

Gerd HESINA and Robert F. TOBLER

(Dipl.Ing. Dr. Gerd Hesina, Dipl.-Ing. Dr. Robert F. Tobler, VRVis Research Center,
Donau-City Str. 1/3, Vienna, Austria, {hesina|tobler}@vrvis.at)

1 ABSTRACT

Due to the rising interest in urban visualization, a number of issues have emerged that need to be solved for practical application. The most prominent of these issues are: visualization performance and security. Visualization speed concerns the usability of urban visualization. In order to make urban visualization practical, it is necessary to completely hide all latency of loading data. Here we will demonstrate techniques like multi-threading, different rendering methods and level-of-detail that our platform uses to maintain interactive frame-rates in the face of the large amounts of data that are common in urban visualization environments. The second issue, securing the geometry and image information contained in an urban model is of eminent importance for the owner of the data. We will point out a number of techniques that have been incorporated in our viewing platform in order to maintain a high level of data security: data compression and encryption, obfuscation of access keys, compression and encryption of executables. While perfect security is an unattainable goal, the multi-staged approach of security in our system significantly raises the cost of illegally obtaining access to the geometry or image data. By demonstrating solutions for both issues our framework serves as a viable platform for urban visualization.

2 INTRODUCTION

During the last few years, the advent of various methods for acquiring models of the various buildings and facades of a city, the interest in visualizing these models in an interactive way has dramatically risen. Up to now most of these visualization solutions are standard applications that have been built for specialized uses in research and business settings. However these models can serve a variety of purposes if they are made available to the public via the internet. This has been demonstrated by the Virtual Old Prague project. However there are a number of issues that arise when a complete model of a city should be made available for walk-through applications via the internet. The main problems are *transmission performance*, *visualization performance* and *data security*.

3 VISUALIZING A CITY

In order to display a city over the internet the standard setup is to use a standard client server setup. The server consists of a standard web-server which is connected to a database containing all the geometry and texture data. This server serves requests for geometry data and textures that are issued by a client based on the movement on the user through the virtual city. In order to limit the amount of geometry that has to be transferred to the client at one time, the geometry of the city is split up into reasonably sized chunks that can easily be addressed by their absolute location. The client can then issue request for the immediately visible chunks and request additional chunks as the user moves through the streets of the city.

3.1 The Server

The standard setup for the server is a web-server and an associated database or an additional dedicated database server. In order to speed up geometry request to the database server, it will probably be necessary to create a dedicated database for the walkthrough application, that only contains the relevant data. Thus the original data, which might be based on a scanning process, and can lead to a huge amount of data, will have to be preprocessed to create simple models for walkthroughs, possibly in multiple levels of detail.

3.2 Geometry Transmission

In order to transmit geometry data and associated textures to the client a geometry format needs to be chosen. As we built our solution on the Virtual Prague project, we use VRML as the format of choice. Due to its fairly complete featureset it serves as the ideal basis for such a project. The VRML standard alrady contains provisions for multiple levels of detail, and thus it is only necessary to set up a scheme that allows to transmit the data as needed.

3.3 Using an Embeded Client

The solution pioneered by the Virtual Old Prague project, is to use a standard plugin for Internet Explorer as a client. This embedded client uses Internet Explorer's Java for implementing functionality that cannot be easily modelled using VRML features. Although this solution has the advantage of supplying an integrated user interface for the user, the embedded VRML viewer pays a significant performance penalty when compared to a standard geometry viewer.

3.4 Using a standalone Client

In order to avoid these performance problems, we implemented a standalone client, based on an own viewer. In order to simplify the transition to this standalone client, it uses the exact same transmission protocol than the embedded viewer. As our new viewer is not hampered by the interaction with a html-browser, it can achieve the maximal graphics performance available on the machine it is deployed. In addition to that, the user interface of this viewer can be tailored to the needs of a walkthrough-application without the interference of the web-browser.

4 IMPROVING TRANSMISSION PERFORMANCE

If every single request from any of the clients results in a database request on the geometry server, this server can get to be a bottleneck. Thus it is necessary to devise some way of avoiding database requests in the first place. The most promising approach in this direction is, to generate a single VRML file for each chunk of geometry that contains static links to its neighbouring chunks. With this optimization all the requests can be handled exclusively by the web-server, completely avoiding the database server. In addition to that, these static files can be cached by proxy servers, and thereby reducing the load on the web-server as well.

5 IMPROVING VISUALIZATION PERFORMANCE

5.1 Improving visualization Performance of the embedded client

If the embedded client is used, the only possibilities for improving visualization performance are based on minimizing the amount of geometry that needs to be displayed. This can by done by a number of methods, among these are:

- **Optimizing the amount of geometry per chunk**: as the geometry is diced into chunks that are downloaded one at a time, the amount of geometry in each of these chunks needs to optimized. They should be small enough to provide good download performance, and contain little enough geometry for the browser to easily display, but they have to be large enough to avoid flooding the server with too many requests.
- **Generating levels optimized levels of detail**: in order to improve the display performance, each chunk of geometry should be available at different levels of resolution, and only the chunk around the viewpoint should be displayed at the highest possible resolution. The other chunks can be displayed at lower levels of detail.
- **Discarding invisible geometry**: geometry that can never be seen like courts, back-sides of roofs aso. can be discarded when generating the database for the walkthrough application.

5.2 Improving visualization Performance of the standalone client

In principle all methods that improve the performance of the embedded client, also benefit the standalone client, only the specific parameters for the optimal choice of chunk size, levels of detail, aso. may be different. However in addition to that, there are some additional approaches that can be used in the standalone client:

- **Threaded downloads**: in order to avoid any delays of the visualization while data is being downloaded, all the network downloads can be performed in seperate threads, without blocking the visualization thread. Thus the user still experiences a responsive application while additional data is gathered for display.
- **Caching of geometry**: The standalone client can keep track of the memory utilization and use all available RAM as a cache for geometry and textures. Thus it can achieve a significantly improved performance if the same places in the city are revisited.

6 ENCRYPTION OF THE TRANSFERRED DATA

The solution of Prague fulfills the demands initially placed but fails to support encrypted data transfer. In order to meet all demands for encryption a concept to ensure encrypted data transfer has been developed. This concepts involves encryption of geometry data and encryption of textures. Since the concept uses a modular design, it is easily possible to integrate it into existing systems.

6.1 Geometry encryption

The geometry encryption we employ uses a combination of Java and VRML. If the client side issues a request for a model, this request results in the delivery of a special VRML file, which when loaded, starts a Java loader. This Java loader makes it possible to transmit all geometry in encrypted form, and decode it on the client side using Java. With the help of the external authoring interface the decoded geometry can be supplied to the Browser and displayed. Additionally security can be increased by using the https protocol to partly switch off caching of Internet Explorer. If the https protocol is not used it is possible to find the encrypted and packed files in the browser cache. Although this is not perfectly secure, breaking the encryption of these files requires a considerable additional effort, and thus should serve as a significant obstacle to obtaining the unencrypted data.

6.2 Image encryption

Encryption of images is done in a similar way. By transferring the control of image loading to a Java loader, an arbitrary decryption process can be started on the client side. This decryptor then supplies the decrypted and decoded image to the embedded VRML browser. The decrypted version of the image file is never stored on disk, and thus the process is reasonable secure. However, since the textures in typical city models represent significantly more data then the geometry, a large amount of data has to be decrypted. As this decryption and decoding is performed in Java, the speed of the client is considerably slowed down when textures are encrypted.

6.3 Securing the decoding process

When an embedded client is used, the complete decoding process needs to be performed in the Java Virtual Machine on the client side. In order to the secure this process, a number of obfuscation techniques have been employed, so that this process cannot easily be replicated. Although this is not a bulletprof protection against malicious attacks, obvious security holes have been plugged, and thus our system represents a reasonable secure urbon visualization solution.

6.4 Encryption in the standalone client

As the standalone client is programmed in a compiled language, it only suffers minimal performance penalties for the encryption process, even if large numbers of textures are used. Thus this client not only achieves the better user experience in terms of user interface and display performance, it also represents the solution with the better security. In this client additional encryption layers are possible, as the current level of encryption does not slow down the client in any noticeable manner.

7 CONCLUSION

We presented a solution for internet visualization of a city that is based on the Virtual Old Prague project, that improves both visualization speed and data security. Thus our solution can be used for communities to make their city available for walkthrough on the internet, without having to fear that the data, which has often been acquired at considerable costs, is available for other purposes.

8 ACKNOWLEDGEMENTS

This paper is based on a join project of the VRVis Research Center, Vienna, Austria (http://www.vrvis.at) which is partly funded by the Austrian government research funding program Kplus and the partner company Nolimits GmbH, Graz.

9 REFERENCES

BAUER, J., KLAUS, A., KARNER, K., ZACH, C., SCHINDLER, K.: MetropoGIS: A Feature based City Modeling System. Photogrammetric Computer Vision PVC02, ISPRS-Commission III, September 9-13, 2002, Graz.

BOUCHNER, P.: Data Acquisition for VIRTUAL OLD PRAGUE. Central Europen Semina on Computer Graphics (CESCG), Bratislava, Slovakia, April 2002.

BROLL, W.: An Internet Protocol for Virtual Environments. International Symposium on the Virtual Realisty Modeling Language (VRML), 1998.

BRUTZMAN, D.: The Virtual Reality Modeling Language and Java. Communications of the ACM vol. 41, no. 6. June 1998, pp. 57-64.

BRUTZMAN, D., ZYDA, M., WATSEN, K., MACEDONIA, M.: Virtual Reality Transfer Protocol (vrtp) Design Rationale, Workshops on Enabling Technology: Infrastructure for Collaborative Enterprises: Sharing a Distributed Virtual Reality, MIT, Cambridge Massachusetts, June 18-20, 1977.

HOLZER, J., KARNER, K., LORBER, G., ARTES, S.: Digitale Stadtmodelle als Plattform für intuitive Stadtplanung und Bürgerinformation. Proceedings CORP 2002, 7th International Symposion, February 27 to March 1, 2002, Vienna.

SLAVIK, P., BENES, B., BERKA, R., MARAK, I., SIMEK, P., SOCH, M., ZARA, J.: Virtual Old Prague, http://sgi.felk.cvut.cz/Prague/.