*Fifth International Conference on Information Visualisation - IV 2001*, London, UK, 2001

# A Scaleable Approach to Visualization of Large Virtual Cities

Jiri Zara[1], Pavel Chromy[2], Jiri Cizek[2], Kamil Ghais[2], Michal Holub[2], Stanislav Mikes[2], Jakub Rajnoch[2]
[1]*Department of Computer Science and Engineering, Czech Technical University in Prague*
*zara@fel.cvut.cz*
[2]*Faculty of Mathematics and Physics, Charles University in Prague*
*{pchr5077, jciz6107, kgha6147, mhol5152, smik6381, jraj7493}@barbora.ms.mff.cuni.cz*

## Abstract

*Visualization of large urban complexes on the web is highly demanding task both from the networking and computational point of view. The whole three-dimensional model of a city is mostly too big to be downloaded in a reasonable time and to be stored in a memory of commonly used computers. This paper presents a scaleable approach for subdividing an urban model into smaller parts. Dynamic loading and unloading data allows theoretically unlimited extension of a model with constant network and computation load. Further requirements specific to presentation of virtual cities are also discussed. Theoretical considerations are followed by a practical implementation utilizing VRML language for modeling historical city of Prague.*

## 1. Introduction

Contemporary technologies (fast CPU, 3D accelerated graphic cards, and fast network) allow interactive visualization of quite detailed parts of a virtual city with very authentic 3D models combined with photographically obtained textures. Several cities can be currently found on the web including well-known towns like Paris [1], Glasgow [2], New Orleans [3], Sydney [4] or Toronto [5]. Two main technologies are frequently used for modeling and visualization – QuickTime VR [6] and VRML [7]. While QuickTime approach is based on photographs only, the VRML uses both 3D models and images. In this paper we focus on utilization of VRML due to its higher interactive functionality.

The common problem arising in almost all examples mentioned above is the amount of data that must be sent through the web and rendered in one. The size of such city models ranges from hundreds of kilobytes up to tens of megabytes per city district. Most users are unable to download this data in interactive time due to the network connection speed. Moreover, such large data cannot be rendered in a real time even when powerful graphic accelerators are used. Therefore the solution is to send to the user only a small amount of data describing his/her vicinity. Because of memory limits, no longer used data should be simultaneously removed from the memory as a user walks through the virtual town. Similarly, new data should be downloaded and prepared in advance. These demands require well-prepared and structured city models, designed for this specific purpose. Most of currently implemented virtual cities use either no data hierarchy or they divide the city into smaller blocks connected via hyperlinks and presented by parts. While the first approach is inefficient the second causes undesirable pauses/jumps when walking from one virtual district to another one. This paper proposes a universal data structures and methods suitable for smooth and continuous walk-through arbitrary virtual city.

The paper presents in Section 2 typical requirements for virtual cities presented on the web. Section 3 introduces a solution for scaleable structured models and Section 4 describes one concrete implementation of the proposed approach.

## 2. Characteristics of Virtual Cities

Three-dimensional interactive models of real cities have several specific features different from other spatial virtual scenes. They are characterized by large number of virtual houses covered by texture images of big sizes. Hardware support for texture mapping is often required when the city is presented to users. Number of textures can reach hundreds or even thousands of images. The modeled area covers several squared kilometers and includes from tens to hundreds of houses. On the other hand, virtual models of houses can have simple shapes because users mostly walk on the streets and do not enter any houses. This fact leads to significant simplification – houses are typically represented by their front faces (frontages) only. Neither roofs or floors are to be specified. The only exception is the bird view of a city when all roofs are visible. This case should be suppressed since it requires rendering of the whole town. This is

**Figure 1.** Example taken from 2D map of Dublin shows that a given square area determines a lot of visible objects in various directions and distances. The small circle marks an area currently examined by a user. The big circle containing all potentially visible objects has a radius about 300 m. Visible objects represent less than 10 % of data covered by the big circle. (Courtesy of Jiri Bittner, CTU Prague)

usually impossible for large cities thus the visitor should be delighted by a number of other nice urban features rather than the bird view.

The main purpose when presenting virtual cities is to give a complex information to users. Virtual visitors can be attracted to become real visitors/tourists later. The 3D model is only a subset of offered information. The full and rich model should contain the following components:

- 2D map synchronized with user's movement in 3D virtual space
- Additional textual information, description of sights
- Direct choosing of target places from a list or a map
- Interactivity (animations, hyperlinks)
- Virtual bus tour (animated viewpoint)
- Search function combined with automatic navigation from current to target place
- Help subsystem, hints, advices

All those information are usually stored in a database. The idea to store also a structured 3D model in a database is thus straightforward. Some applications already use a database [8], but mostly for searching purposes and not for the task of spatial arrangement of virtual houses and other models. The following section discusses problems and solutions for such a task.

## 3. Scaleable urban model

When thinking about the structure for the virtual city an idea of regular space subdivision seems to be inviting and suitable. A regular grid has good properties from the perspective of finding neighboring city parts for a given region. When a user examines a certain cell of a grid, four or eight neighboring cells can be read in advance and prepared for the visualization. Similarly, when a user leaves currently visited cell, a set of probably no longer examined city areas can be easy determined and unloaded.

Unfortunately, the problem of advanced data preparation cannot be solved on the base of topology only. A user can see a number of virtual houses that are very far from a given area. This is demonstrated on the Figure 1 showing a user examining a square in the city center. While house frontages around the square hide many houses in the closed vicinity, long and straight streets allow seeing objects placed hundreds of meters from a user. The conclusion is that the city structure should take both *topology* and *visibility* issues into a consideration. A presentation system should download models which are both *near* the user and *visible* from his/her current position.

### 3.1. Topology

The grid or other regular structure does not fit into arbitrary city configuration. Only a few real towns like New York has a strict rectangular structure while other cities with historical centers delight users with serpentine alleys and variously shaped plazas. A topology of a virtual city has to correspond with the reality thus non-uniform but flexible data structures are required. Of

course the request for fast determination of neighbors for a given area has a big importance.

We propose a system of *blocks* corresponding with streets, plazas and other open spaces. One block is typically assigned to a square while several successive blocks represent one long street. This approach is similar to space subdivision used for QuickTime VR presentations [6] where the whole space is covered by set of cylindrical panoramas. Here we extend this method to universal description of 3D virtual space.

The block is depicted on Figure 2. The polygonal layout is basically two-dimensional. The border consists of *edges* holding house frontages and *gates* allowing walking from one block to the neighboring one. Special *closing gates* serve as obstacles when a user wants to visit spaces that are not modeled yet. Standalone 3D models representing statues, fountains, tram stops, trees and other solitaires can be placed into a polygonal block.

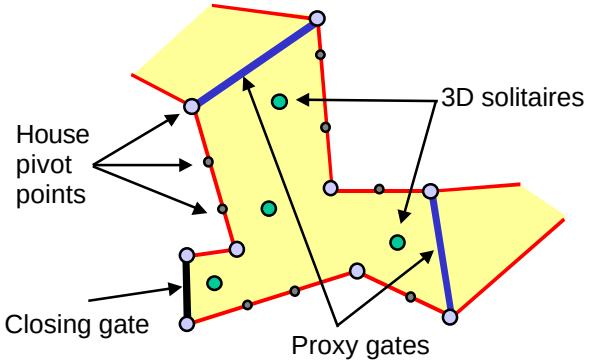


**Figure 2.** Each block consists of area surrounded by house frontages, optional 3D objects and several gates connecting neighboring blocks. The two-sided proxy gate holds information on objects that are to be loaded/unloaded when a user crosses the gate.

A standard *gate* is invisible and users can cross it. The main purpose for introducing a gate is that it holds an information about neighboring block. Each time a user crosses a gate a request for loading/unloading data is produced. This approach suppresses the need for continuous evaluation of user's position in the whole city, because the only important moment is the time when a user walks through a gate. That is why we call gates *proxy gates*. Note that proxy gate is actually two-sided. It should detect whether the user enters into a block or leaves it. Leaving a block leads to releasing certain data, entering causes downloading new models. A two-sided proxy gate can be implemented either by two separate sensors or by one proximity area combined with evaluation routine checking avatar's movement.

It is useful to read data belonging not only to one block but to have a possibility to specify the amount of models to be downloaded in advance. Based on the actual available computer memory a visualization system can request the database holding the city structure for

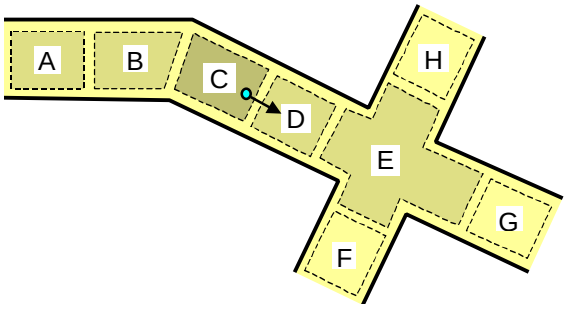


**Figure 3.** Based on the topology only, the specified distance 2 (blocks) ensures, that blocks A, B, D, and E are already in a memory when a user examines the block C. When crossing the gate between C and D, blocks F, G, and H should be downloaded while A can be released.

sending data from blocks in specific distance from currently examined block. This can be achieved by breadth-first search algorithm known from graph theory.

A simple set operation is needed to determine data for unloading – the difference between newly specified set of blocks and a set of blocks already presented in a memory. Note that these two sets should contain a common subset of blocks ensuring that a user can walk through blocks without undesirable waiting for a data. Figure 3 depicts the situation when the user leaves block C and enters block D. A common subset consists of blocks B, C, D, and E for that case.

An ordinary edge in the block structure holds data for several adjacent house frontages. A closing gate can be also considered as an edge with one face. Usually this face is covered by photography showing street or other free space that is not included into a 3D model.

### 3.2. Visibility

A potentially visible area for a given user's position contains typically quite different blocks than those found on the base of topology only as shown on Figure 1. Thus the visibility issue has to be included into a city structure. Each block contains a list of blocks that could be seen as a user moves within this block. Data belonging to potentially visible blocks should be read as soon as the block is evaluated by topology search algorithm to be a candidate for advanced downloading. Note that lists of visible blocks can overlap for neighboring blocks thus

they have to be maintained using set operations similarly like in the previous case of topology.

The problem of finding the potentially visible set of polygons for a given spatial arrangement has been intensively investigated in past decades. The literature from the field of computational geometry and computer graphics [9] introduces a lot of methods that differ in the speed (from off-line to real time processing), utilization of additional data structures (mostly based on trees), computations performed in various coordinate systems (dual space, 2D space only) and other features. Here we do not suggest using any specific method. Relatively slow algorithms can be used since the evaluation is performed in time of creating a city structure. This preprocessing saves resources for later real-time visualization.

Note that the visibility computations should be repeated each time when a database containing the city structure is changed. Newly added houses can be visible from many blocks. The example is Eiffel Tower in Paris. While the topology evaluation is 2D problem only, a visibility should be really computed in 3D space.

### 3.3. Memory management

When designing data structures for scaleable virtual cities, the VRML language has been considered as a tool for the final presentation. The memory management in this language is unfortunately not directly accessible and the visualization system cannot control the memory allocation/de-allocation. VRML browsers are often part of WWW browsers. They depend on the host memory management algorithm based mostly on late garbage collection. No explicit command in VRML allows for memory control.

The future extension of the VRML language will bring certain improvement. The GeoVRML specification [10] introduces *GeoLOD* and *GeoInline* nodes allowing requesting a browser for releasing data from a memory. Although the basic approach to geographic data expects a use of orthogonal 2D grid for the description of a virtual world, a proposed structure for virtual cities could also benefit from the new VRML nodes. Currently the implementation of visualization system has to trust to external languages like JavaScript and Java and their memory management subsystems. Routines written in those languages are able to communicate with the database on the server, to transfer 3D data from it, and to control relevant VRML scene structure.

### 3.4. Urban Level of Detail

The principle known as “Level of detail” is frequently used in virtual reality systems. It allows achieving a higher speed of rendering by simplifying a model representation when an object appears far from a user. While this simplification is often based on decreasing a number of polygons per object, virtual houses require different approach. High number of polygons does not cause a problem here since only a few faces represent each house. The most demanding task is to map and render big texture images. Our observation shows that levels of details should carefully work with textures. We have designed the following “urban” levels of detail that save both memory and rendering time markedly.

We propose a four-step level of detail for each virtual house. Those levels are shown on Figure 4 from the simplest to the most complex one. A distance from the user together with graphic performance of client computer can be taken into a consideration when switching between levels. While the simplest representation definitely does not look wonderfully, the second one gives surprisingly realistic results with small network and computation demands. The idea is based on downloading a texture for a window that is the most noticeable part of a house. This image has always a small size (typically 64 x 64 pixels) and it is fast transferred from a server to the client. A window is then repeatedly placed on the front face thus saving a lot of texture memory. One house can hold several windows with various shapes. The visible difference between level b) and c) on Figure 4 varies from case to case, but the memory savings are evident in all cases.

a) Colored front face b) Repeated small texture(s) c) Overall texture d) Textured 3D model

**Figure 4.** Newly proposed levels of details for virtual houses. The simplest representation consists of one polygon, two intermediate levels use textures and the best model is fully three-dimensional.

## 4. Virtual Old Prague

The principles described above have been utilized for the model of historical part of Prague [11]. A student team has implemented a database holding the proposed city structure as well as a number of additional informations (viewpoints, web addresses of shops and other objects located in a city etc.). The database (MySQL) runs on the web server. PHP scripts process requests from clients and prepare new data for them. Virtual Prague is modeled using VRML. Client Java applet synchronizes attached 2D map with user's walk through the town using External Authoring Interface [12]. A program written in JavaScript loads all necessary 3D data and releases unused models from the memory.

All modules of the system are shown on Figure 5. Two utilities have been developed. The first (House Editor) allows semi-automatic construction of "urban" level of detail. It works in connection with another program for drawing a front face of a house when a texture image is given. The second utility (City Editor) maintains geographical data in the database. It allows creation and editing urban blocks with proxy gates and other information. The visibility is not solved automatically yet but it is written to the database explicitly by the author/designer.

The user interface consists of several parts as shown on Figure 6. A user can either directly choose a viewpoint from a list or to interactively pick it from 2D map. Simultaneously updated icon on the map symbolizes user's position and orientation in the city. The 3D window offers real time walk through the town. Sensitive parts (billboards, tram stops etc.) are connected to hyperlinks that are targeted to a new standalone window. A textual and pictorial information is presented on the bottom right side accordingly to user's movement from one block to another one. Several control icons on the top allows to set a view angle, a height of a user, a walking speed, to display a process of loading data from the server, and to change various system settings like required quality of houses with levels of detail, computer performance, and network parameters.

## 5. Conclusion

The scaleable data structures describing virtual cities have been introduced in this paper. A sequence for levels of detail specific to urban modeling has been designed. Although originally targeted to VRML, the proposed approach can be directly used in arbitrary visualization tool working with 3D models.

Theoretical work has been followed by a practical implementation of visualization system for historical city

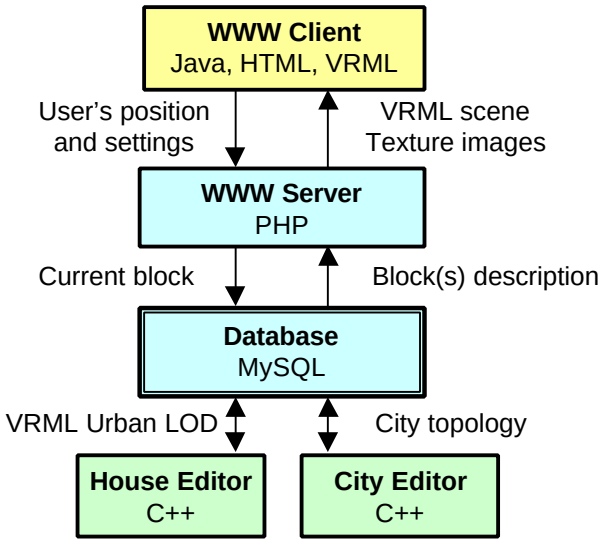


**Figure 5.** Modules used for the model of the Virtual Old Prague. The database stores basic geometrical information. VRML models and texture images are located on the web server file system.

of Prague. Currently about 60 houses were modeled and placed into a database. Even though this number is far from real requirements to complex presentation of the whole city, the measurements performed on those data have demonstrated the accomplishment of intended goals. The virtual walk through the town is smooth thanks to reading 3D data in advance. Careful loading and processing textures in levels of details saves graphics performance thus the Virtual Old Prague can be presented on ordinary computers with acceptable speed of rendering.

It should be stressed that the visualization system utilizing several technologies and components (HTML, VRML, Java applets, and JavaScript routines) is not as steady as we wish. The bottleneck of the system seems to be a VRML browser. Its stability when manipulating with scene structure is low. Two browsers were tested – CosmoPlayer and Cortona. The final visualization program had to be adapted in such a way that it uses different techniques for different VRML browsers. We hope that VRML browsers will be implemented much more robustly in the future thus the dependencies on certain browser could be removed.

## 6. Acknowledgment

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under research program No. Y04/98: 212300014 (Research in the area of information technologies and communications).


**Figure 6.** Main window of Virtual Old Prague visualization system contains several components. Virtual houses are modeled in four levels of details. All of them can be found in VRML window in this figure.

## 7. References


[1] *Virtual Paris* (based on blaxxun technology) http://www.2nd-world.fr/

[2] *Virtual Glasgow* (VRML model) http://iris.abacus.strath.ac.uk/glasgow/

[3] *Virtual New Orleans* (VRML model) http://www.planet9.com/earth/neworleans/

[4] *Virtual Sydney* (VRML model) http://www.planet9.com/earth/sydney/

[5] *Virtual Toronto* (VRML model) http://www.intoronto.com/

[6] *QuickTime VR* (Apple web site) http://www.apple.com/quicktime/qtvr/

[7] The Virtual Reality Modeling Language. *International Standard* ISO/IEC 14772-1:1997, http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm

[8] A. Heinonen, S. Pulkkinen, and I. Rakkolainen: "An Information Database for VRML Cities". In *Proceedings of IV 2000 - Information Visualization 2000*, London, Great Britain. IEEE Computer Society, ISBN 0-7695-0743-3, pages 469-473.

[9] F. Durand: "A multidisciplinary survey of visibility". *ACM SIGGRAPH Course notes.* Visibility, Problems, Techniques, and Applications, July 2000.

[10] M. Reddy, L. Iverson, and Y. G. Leclerc: "Under the Hood of GeoVRML 1.0". In *Proceedings of The Fifth Web3D/VRML Symposium*. Monterey, California. February 21-24, 2000 http://www.ai.sri.com/geovrml/

[11] *Virtual Old Prague.* Student Project. http://www.ms.mff.cuni.cz/vsp/

[12] The Virtual Reality Modeling Language - External Authoring Interface. *Draft* ISO/IEC 14772-2, http://www.vrml.org/WorkingGroups/vrml-eai/Specification/