Texture Management for high-quality City Walk-throughs

Gerd HESINA & Stefan MAIERHOFER & Robert F. TOBLER

VRVis Research Center, Donau-City Str. 1/3, Vienna, {hesina|maierhofer|tobler}@vrvis.at

1 INTRODUCTION

For a city viewer application to be a useful tool in supporting municipal planning and various other urban applications, high quality texture maps, for both terrain and building facades are essential. We present a texture management system which makes it possible to view high resolution textures on a state-of-the-art consumer PC, with a resolution of up to approximately 1 cm per pixel for all the buildings in a city walkthrough. The system is based on estimating the average viewing distance for all facades and the terrain in the field of view, and continuously loading the necessary high-resolution textures for optimal display. The effect is to simulate a high-resolution texturing of the complete city corresponding to a total of multiple gigabytes of texture data. By providing such high texture quality, important detail information is available for both the city planner and the tourist: street labels on houses can be read, traffic signs are easily discernible, and ground markings can be read (provided the data is available). The system has been successfully demonstrated using the data provided by the City of Graz.

Figure 1: Screenshot of a Walk-through of the city of Graz

2 PROBLEM STATEMENT

State of the art City Models that include image data for facades and terrains, contain huge amounts of textures that cannot be loaded into main memory or onto the graphics card at once. Thus in standard viewing applications it is not possible to view the model with all the modelled data. For a number of applications in the areas of municipal planning and urban management, it would be highly valuable to be able to view all the data at once. In order to facilitate such a viewer application, the available geometry and textures have to be organized in such a way, that the part that is visible at any time can be loaded on demand. Previous work on this topic is either specialized to terrains [ULRICH] or very general without taking into account the special properties of city geometry [BOOL, CIGNONI et al.].

3 SPLITTING OF THE MODEL

The available data of a city model for viewing contains two major classes of data: geometry and textures. In the general case, the available data in both these classes is larger than the available main-memory. In order to be able to load parts of the geometry and textures, the data has to be split into manageable chunks. The strategies for splitting the data are different for the two classes of data: geometry has to be handled differently than the texture data.

3.1 Splitting of the Geometry

In order to be able to load the visible geometry on demand, the geometry has to be split into chunks in such a way, that the viewer application can easily calculate which chunks are close to the viewing point, or will be close to the viewing point in the near future, so that they can be loaded into main memory. Here an arbitrary hierarchy of splits such as a k-d Tree or a BSP-Tree would be possible, however in order to keep the management algorithms as simple as possible, a quad-tree based structure was chosen: the geometry is split in half in both dimensions at each level, giving 4 rectangular sub-chunks at each level. At the finest split level this gives rise to a rectangular grid of geometry chunks.

The typical geometry in a city contains terrain data and building data. Splitting the terrain data geometrically is necessary, since the complete terrain object is as large as the complete city. However it is in general not necessary to split the data of each and every building, it is only necessary to place the geometry of each building inside the chunk that contains the center point of the building. This avoids costly and unnecessary splits of geometry.

The terrain data of a city is often associated with a texture map containing an orthophoto of the terrain. This is often generated from aerial photographs and available as a single large texture image. In order to be able to load this texture in manageable chunks, the texture data has to be split into small textures that correspond to the chunks of geometry that have been created by splitting the geometry data.

3.2 Splitting of the Textures

In order to be optimally suited for rendering on current rendering hardware, it is necessary to split the terrain texture maps into tiles that have dimensions that are powers of two. A large terrain will then be textured using a grid of texture tiles. Optimally these texture tiles are completely aligned with the grid of geometry chunks. Although it is normally useful to resample the data in order to meet the power-of-two requirement, resampling on a rotated grid is complicated and error-prone. Therefore we decided to align the geometry splits with the texture splits, simplifying the underlying management routines.

Texture seams

A naive splitting algorithm for the textures, would just cut the textures into power-of-two tiles and calculate the necessary texture-coordinates for each associated geometry chunk. However due to the linear interpolation of texture maps that is used in current hardware in order to increase the rendering quality, this can lead to visible texture seams at the border lines of the texture tiles [TURNER, B] (see figure 2a).

Figure 2: a) Texture split with seams on the left. b) Corrected textures on the right

In order to avoid these artefacts it is necessary to generate the texture tiles with one pixel overlap, and adjust the texture coordinates accordingly (see figure 2b). Only by integrating this improved splitting method, high quality textured terrains are feasible.

3.2.1 Sequence of Texture Resolutions

The resampled textures are made available at a sequence of resolutions, so that they can be rendered at different output quality, depending on the distance to the view-point:

- Textures which appear close to the view-point are rendered at the maximal available resolution in order to achieve the highest possible rendering quality.
- Textures which appear very far from the view-point are rendered at a low resolution, so that they need not be kept in memory at their highest quality level.

4 DATA MANAGEMENT

Based on the split geometry and textures, the second important step for interactive city visualization is the management of loading this geometry and the associated textures into memory on demand. In order to accomplish that, it is necessary to maintain a cache for the geometry and the textures in main memory.

4.1 Caching of Geometry

Currently the city models that we have available are of limited geometric complexity: the model of the city of Graz has about 150 000 Triangles, including both buildings and terrains. Thus the geometry of such a model can be completely loaded onto the on-board memory of current graphics hardware. Thus it is currently not necessary to implement and use a full-featured geometry caching system. Of course this will be changing in the near future, and therefore we are developing a geometry caching system that interacts with the texture caching system described in the following sections.

4.2 Caching of Textures

The texture cache in main memory is responsible for holding each texture tile at an appropriate resolution in memory. Thus the following functionality is implemented:

- Loading of texture tile at a specified resolution.
- Recording of timestamps whenever a texture is used (i.e. viewed) at a specific resolution.
- Releasing of texture at a specified resolution based on usage: e.g. a least-recently-used (LRU) scheme, that eliminates textures that haven't been used for a while.

4.3 Two-level cache hierarchy

The texture cache in main-memory is however only one step to a complete texture management: all textures that need to be rendered also need to be loaded into the on-board memory of the graphics card. Thus a second cache has to be maintained, that works on the same principles as the main-memory cache, but manages the textures that reside on the graphics card at any point in time.

4.4 Implementation Issues

For both caches, it is important to schedule the loading operations in such a way, that they do not block rendering, as this would result in stalls that impede interactive viewing. For this reason,

4.5 Texture cache policies

The basic functionality of the two-level texture cache only provides the mechanism for getting high resolution textures into memory when they are needed. Based on these mechanisms, an adequate policy has to be set, that determines when to load a specific texture, and at what resolution to load this texture. Such a policy has to take into account a number of parameters, so that it can be fine-tuned to a specific application:

- Distance of the textured object to the view-point: far away objects can be shown with very low resolution textures, whereas close-by objects need to be shown with high resolution textures.
- Speed of the view-point: when the viewer moves through the city, it is prohibitively expensive to load the textures for all close-by objects at high resolution, only to release them moments later when the viewer has passed the objects.
- Size of the texture caches: the policy has to be designed in such a way, that it optimally uses the available memory.

Experimenting with a few policies that take the above points into account, we arrived at the following cache policies:

- Low resolution textures for the complete geometry is loaded on startup. The resolution is chosen in such a way, that all low-resolution textures will fit in memory.
- High resolution textures are loaded for objects that are visible and whose distance is closer than a predefined threshold. Determining of the actual objects can be performed in various different ways, such as stochastic raycasting.
- For a city walk-through application that allows to switch into a flying mode, the height above ground can be used as a simplified estimate for the closest distance of objects: above a certain height it is not necessary to load any high resolution textures (see figure 3).
- The movement speed of the observer: above a certain threshold speed, no textures are loaded into memory.

The described policy only uses two resolutions, a low texture resolution that is chosen so that all textures fit in memory (with enough free space for high resolution textures) , and a high texture resolution, that is chosen in such a way that enough space for all high resolution textures for visible close-by objects is available.

Of course more sophisticated cache policies are possible, and we are continuously improving our existing policies, nevertheless the described policies lead to a fairly good user experience already and can be thought of as a basis for city walk-throughs.

Figure 3: Screenshot of a Fly-over of the city of Graz

5 CONCLUSION AND FUTURE WORK

We demonstrated how intelligent texture management can cope with the huge amounts of texture data that is available in city models with facade and terrain images. This texture management makes it feasible to interactively view state-of-the art city models on consumer PCs.

However it is foreseeable, that in the near future large geometric models will be available for certain buildings in a city. We are currently developing improved algorithms to manage this increased amount of geometry while maintaining interactive frame rates in our viewer application.

6 ACKNOWLEDGEMENTS

This work has been done at the VRVis research center, Vienna, Austria (http://www.vrvis.at), which is partly funded by the Austrian government research program Kplus. Thanks to the City of Graz for the use of their city model in this research project.

7 REFERENCES

BOOL, C.: View Independent Progressive Meshes (VIPM), June 2000. http://www.cbloom.com/3d

CIGNONI, P., MONTANI, C., SCOPIGNO, R.: A comparison of mesh simplification algorithms, Computers and Graphics, Vol 22, 1, pp. 37-54, Feb. 1998.

TURNER, B.: Texture Seams. http://www.fractalscape.org/TextureSeams

ULRICH, T.: Rendering Massive Terrains using Chunked Level of Detail Control. SIGGRAPH '02 Course Notes, Course 35.