Synthetic Textures for 3D Urban Models in Pedestrian Navigation

Volker Coors

Stuttgart University of Applied Sciences, Schellingstr. 24, 70174 Stuttgart
Germany, volker.coors@hft-stuttgart.de

Abstract

Since years the market of mobile navigation systems is growing enormously. Within this paper the goals and first results of the joint project “Mobile Navigation with 3D City Models” (MoNa3D; http://www.mona3d.de) are introduced. The project consortium consists of the University of Applied Science Stuttgart (coordinator), the University of Applied Science Mainz, the University of Bonn and the four companies Navigon, Teleatlas, GTA Geoinformatik, and Heidelberg Mobil. The aim of the project is to develop and evaluate the support of landmarks and 3D visualization in mobile navigation systems for pedestrian navigation as well as the “last mile” car navigation for instance from a parking place to the final destination. The two main goals of the project are to provide a cognitive semantic route description by using landmarks and secondly to make use of synthetic building facade textures along with a corresponding compression system for an efficient storage, transfer, and rendering of 3D urban models. This paper focuses on the synthetic texturing and compression.

Keywords: mobile navigation, synthetic textures, 3D urban models

Introduction

There is a wide variety of techniques to present directions and support on mobile devices ranging from spoken instructions to 3D visualization. In order to produce a coherent and cohesive presentation it is necessary to adapt a presentation to the available technical and cognitive resources of the presentation environment. Coors et al. (2005) evaluated several means of route instructions to a mobile user. 3D visualization seems to be well suited where time and technical resources are not an issue, and where the available position information is imprecise: a 3D model of the area allows the user to search her environment visually for specific features, and then to align herself accordingly, thereby compensating for the imprecision. 3D urban models as integrated part of a navigation system will support pedestrian navigation as well as the “last mile” car navigation from a parking place to the final destination. Especially using public transport and finding the best exit at a larger tube station for instance is a typical three-dimensional problem that is hard
to solve with traditional 2D navigation support. The two main companies for navigation data, TeleAtlas and Navteq currently offer 3D landmarks of almost all big European cities. For some cities even large 3D urban models are available (Vande Velde 2005). In Germany, more and more municipalities themselves build up 3D urban models. However, only a few research prototypes actually support mobile navigation with real 3D urban models on mobile devices such as smartphones (Coors & Schilling 2003, Nurminen 2006).

The aim of the project “Mobile Navigation with 3D City Models” (MoNa3D) is to develop and evaluate such a mobile navigation system. The two main goals of the project are:

- To provide a cognitive semantic route description by using landmarks in 3D, which allow context dependent personalized navigation support
- To develop an approach to create suitable building textures using image processing methods and synthetic textures along with a corresponding compression system for an efficient storage, transfer, and rendering of 3D urban models.

The focus of this paper is on the second goal of the Mona3D Project: synthetic texturing and compression of 3D urban models.

**Synthetic Textures**

In photogrammetry, a 3D building model is usually created by geometry and textures. Digital pictures of actual buildings are modified and mapped onto the building geometry. With this method very high detailed and visually photo-realistic building models can be created. However, it is a huge amount of work to create high quality models. In addition, the resulting textures are memory intensive and could not be used on mobile devices due to its memory limitations.

The aim of synthetic textures in this field is to build inexpensive and small building textures but still represent the building as close to reality as needed for navigation purposes. The goal of texture synthesis is to generate new textures that look similar to a given sample texture. Texture synthesis can be roughly subdivided in two main categories: procedural texture synthesis (Ebert et al. 1998), and texture synthesis by example (Sabha, M.et al 2007).

Procedural texture synthesis is basically an algorithm that generates the needed facade texture. Certain pattern like stone and brick walls can be analyzed and synthesized. Parish and Müller (2001) extended this approach by so-called layered grids, a semi-automatic technique based on layering and functional composition to create procedural models of entire building facades.
Texture synthesis by example generates a novel texture that is similar to the given sample. Texture synthesis by analysis usually characterizes a sample texture by a statistic. The new texture is synthesised such that the texture statistics are maintained. Pixel- and Patch-based texture synthesis create new textures by selecting and copying a single pixel (or a whole patch) from the sample texture, based on already synthesised pixels in the novel texture. While texture synthesis by example is usually used to generate texture for specific materials such as bricks and marble, and natural phenomena such as clouds, Sabha et al. (2007) have demonstrated an approach to generate entire building facades by using exact neighborhood matching.

Tele Atlas together with GTA Geoinformatik produce textured 3D urban models for navigation purposes based on a texture library approach (Vande Velde 2005). Each building facade is split into components such as window and door types per storey. The used facade textures are taken from a generic library. Prominent buildings that are used as landmarks are modeled in detail by classical photogrammetry methods.

In the Mona3D project a procedural texture synthesis approach combined with patch-based texture synthesis is used. A facade texture is generated based on libraries of typical window, door and material images. Under the assumption that the building geometry is given as a boundary representation based on polygons, each wall of a building is represented by at least one polygon. For each of these polygons one pulse function for each dimension controls the process of texture creation. For a rectangular polygon these pulse functions $p_x$ and $p_y$ are fairly easy to define:

A pulse function $p: [0,1] \to \{0,1\}$ is used to place features as windows and doors on a given background texture. If the product of the two pulse functions $p_x$ and $p_y$ is 1 the given feature image is inserted, otherwise the background image is used for texturing. To be more flexible, layers of feature textures can be used, for example one layer for windows and a second layer for doors. Each layer should make use of different pulse functions. An example is given in figure 1.

![Figure 1: pulse function to generate a window pattern on a facade](image-url)
The pulse functions can be mapped to any geometry using the local surface coordinate system of this geometry. On GPUs the pulse functions can easily implemented directly in the Pixel shader. The only input for generating the textures are the above mentioned pulse functions and a texture library of typical facade element and background. No individual image texture will be used which saves an enormous amount of image data.

To simplify the definition of the pulse functions a user interface has been developed. For each layer of the synthetic texture, several parameters can be specified to generate the resulting facade texture. For example, the position of window rows and columns and the grouping of windows can be defined. A similar model is given to define the door(s). Currently, a set of input parameters to specify the pulse functions is evaluated.

Figure 2: from left to right: User Interface to specify the pulse function for window layer, the facade generated by this pulse function with an additional layer to position a door; similar with another position of the door; variation of the window layer pulse function.

Compressed 3D urban models

The introduction and application of 3D models in distributed environments and mobile clients has driven the development of efficient 3D compression schemes. While general purpose compression tools such as Win-zip or Gzip achieve average compression results on 3D data, specialized algorithms that are not based on textual representation but take into account the 3D model structure of the content propose a deflation of up to 95% (Eppinger and Coors, 2005).
The de facto standard for 3D urban model data interchange, CityGML, makes use of polygonal boundary representation of terrain data as well as building data. This is also the case for visualization formats such as VRML and the *.m3g format for mobile 3D graphics (JSR 184). Each polygonal boundary representation can always be transformed into a triangle mesh without any loss of information. A triangle mesh is usually represented by its geometry information, the vertex location of the mesh, and by its connectivity information, which describes the incidence relation between each triangle of the mesh’s surface and its boundary vertices. Usually, additional attributes such as color and texture information may be part of the description. However, in the proposed approach of synthetic textures, only a few parameters are stored per polygon or triangle patch. Based on these attributes, synthetic textures are generated during rendering on the mobile device.

Geometry information of the model may in many cases be reduced by lossy compression. Basically, such techniques reduce the precision of the vertices to a level that is sufficient for the corresponding application using a combination of:

- Quantization to map the set of (floating point) coordinates to a less accurate set of values, thereby eliminating (unnecessary) precision overhead. The resulting coordinate may be represented as integer values after quantization.
- Prediction to encode small correction vectors (residues) instead of the absolute vertex coordinates
- Entropy coding such as Huffman or Arithmetic coding to minimize the storage of the residues with variable length bit codes.

Depending on the acceptable loss of precision, which can be adjusted through quantization, this method may deflate geometry information down to approximately $1/7^2$ of its original size (Goodman and O’Rourke, 2004).

While geometry information may be subject to some loss without forfeiting crucial information of the model, connectivity information has to be strictly conserved. Several techniques have been proposed for the lossless compression of the connectivity of triangle meshes (Rossignac 1999, Alliez & Desbrun 2001, Coors & Rossignac 2004). 

**EdgeBreaker** (Rossignac 1999) is known to be one of the simplest and most effective single rate compression algorithms. It traverses the mesh surface in a spiral triangle spanning tree order and generates the so called CLERS string, in which each symbol corresponds to a triangle of the mesh. The CLERS string contains all the information required by the decompression algorithm to restore the connectivity of the mesh by attaching new triangles to previously reconstructed ones. The original method guarantees 4 bit per vertex (denoted b/v for simplicity). This guaranteed upper bound was later improved to 3.60 b/v (Gumhold, 2000).

Instead of encoding the CLERS string directly, Delphi compression (Coors & Rossignac, 2004) tries to predict each CLERS symbol based on the geometry and connectivity of previously visited triangles. Just like Edgebreaker, Delphi compression traverses the mesh starting by a given seed triangle. At each step to the next triangle during this traversal, the compression algorithm encodes the identification of the tip vertex $v_{tip}$ of the new triangle. This is accomplished by guessing an estimate location $\hat{g}(v_{tip})$ using a prediction scheme, for instance parallelogram prediction. If the predicted vertex is
sufficiently close to an already compressed vertex \(v_x\) of the mesh than \(v_x\) is predicted to be \(v_{tip}\). Depending on the location of vertex \(v_x\) the next triangle is predicted as L, E, R, or S. Otherwise \(v_{tip}\) is predicted as new vertex and the corresponding triangle as C. If the guess is correct, a single confirmation bit is written to the so called Apollo sequence, which may be seen as the predicted CLERS string. If the prediction is incorrect, additional information are added to the Apollo sequence to rectify the predicted symbol. Experiments with standard computer graphic benchmark models have shown that Delphi compresses the connectivity of a triangle mesh down to 0.4 bits per vertex, depending on the number of correct predictions, of course.

The compression scheme was tested on portions of a 3D city model of Stuttgart (LoD 2, no textures). Two different parts of the model have been chosen to represent the experiments: A small model of Stuttgart downtown with 111 buildings (11474 triangles) as shown in Figure 3 and a larger residential area with 6771 buildings (280306 triangles). Table 1 shows the mesh compression results.

Figure 3: 3D-Model of the city of Stuttgart, © Stadtmessungsamt Stuttgart.
### Conclusion

In this paper, two main technical challenges towards a pedestrian navigation system using 3D urban models, synthetic texturing and 3D mesh compression have been discussed and solutions have been proposed. Synthetic textures enable the effective and low-cost production of 3D urban models for navigation purposes. In addition, it also solves the problem of storing a huge amount of image data by using texture library elements. Per building facade just a few parameters have to be stored to generate a suited texture which leads to a very compact representation of the texture. The building geometry can be stored and transmitted very effectively by using 3D mesh compression techniques such as the proposed Delphi compression scheme. Beside the compact representation of the building geometry the mesh compression has a second advantage compared to standard text-based compression such as WinZip or Gzip. Using text-based compression, the compressed file has to be decompressed into the original text file and in a second step this usually large file is parsed to build the 3D urban model. In contrast, using mesh compression, the 3D urban model is built directly from the compressed file saving runtime for parsing the uncompressed file.

Currently, the proposed components are integrated into a prototype navigation system based on a service based infrastructure including a Web3D Service that grants access to the 3D urban models and a routing service based on the OpenLS interface. The client is implemented in Java ME using the M3G API in addition for 3D rendering on smartphones.

### Acknowledgements

This work has been supported by the Germany Ministry of Education and Research (BMBF) within the project MoNa3D. Special thank to all project partners and co-workers for their valuable support and input.
References


