Non-Photorealistic Rendering of Complex 3D Models on Mobile Devices

D. Hekmatzada, J. Meseth, R. Klein

Institute of Computer Science II, Computer Graphics Group

Römerstrasse 164, 53117 Bonn, Germany
{hekmatza,meseth,rk}@cs.uni-bonn.de

1. Abstract

Upon today, mobile geographic information systems (GISs) for handheld devices were limited to display of 2D graphics since interactive visualization of complex, textured 3D meshes appears to be impossible due to lack of specialized hardware and memory constraints.
In this paper, we present an approach to render large, triangular meshes on mobile devices with non-photorealistic rendering techniques, thereby achieving visually appealing results at interactive rates. An important feature of our system is the use of progressive level of detail (LOD) representations for the transmitted models, which enables coarse renderings of the model to be loaded very quickly even at limited bandwidth. For the transmission we designed a special protocol assuming a reliable, wireless or wired communication channel.
We report results on the implementation of several different non-photorealistic rendering techniques like silhouette-drawing and feature-edge drawing, which all produce images containing the most important details of the underlying environments. We show that complex 3D GIS models can be rendered at close to real-time frame rates. Our approach is capable of providing enhanced navigation, perfectly suited for the human since the views shown on the screen of the handheld device match the user's view of the environment. It enables the use of 3D town-models even on present-day, low-cost mobile devices.

2. Introduction

The pervasiveness of mobile devices has increased steadily during the last years and according to Gartner Dataquest, in a few years, more mobile devices will provide access to the Internet than desktop computers. Combined with mobile communication, they will open up various new opportunities like ubiquitous Internet access, enhanced navigation and mobile GIS.
The increase of available devices is associated with the steady improvement of the device's technology, providing increasing computational power, enabling more and more sophisticated applications with increasingly complex user interfaces. Especially enhanced navigation and mobile GIS raise demands for real-time 3D graphics even on mobile devices, which unfortunately lack specialized hardware support for these tasks. Non-photorealistic rendering (NPR) methods are extremely suitable for these devices, since they save computational power and convey visual information to the user in a more efficient way than photorealistic methods – especially on the small displays of existing PDAs, which forbid comprehensible display of highly detailed models and additional (e.g. textual) information at the same time.
In this work, we describe a method for rendering complex models on standard handheld devices using NPR methods. We report results for different rendering styles and combinations of these, describe our transmission method and system architecture and finally show, that close to real-time rendering of complex GIS environments is possible on existing, commonly used devices.
The rest of the paper is structured as follows: In section 3 we describe previous work related to this paper, section 4 reports details on our NPR rendering primitives. Section 5 provides details on our implementation, section 6 reports results of our method and finally section 7 concludes and describes opportunities for future work.

3 Previous Work

3.1 Client-Server Rendering

The idea of client-server rendering was first introduced in the area of Virtual Reality to reduce the amount of transmitted data over channels with limited bandwidth. FUNKHOUSER 1995 employs a client-server system to reduce the amount of messages to be interchanged between multiple participants in VR scenes by computing visibility per participant on the server. MANN 1997 uses a powerful graphics-workstation server to incrementally send large collections of textures to a PC based client by transmitting required data on demand, i.e. the client interpolates rendered images from available data while the server renders a difference image between the client's and the original view every couple frames and sends missing texture data to the client. HESINA 1998 selectively downloads parts of the full model that are contained in a circular area of interest of the client. SCHNEIDER 1999 developed an adaptive framework that selects the LOD of 3D objects to transmit based on the available bandwidth, the client's computational power and its graphics capabilities. TELER 2001 employs a client-server model to transmit only parts of a huge model to the client at an appropriate LOD, based on the client's position, view direction, the available transmission bandwidth and the expected improvement in image quality that is achieved by transmitting an object.

In contrast to previous approaches, our system includes a client with very limited computational power and memory and no specialized graphics hardware. By employing NPR methods, we reduce the amount of transmitted data and minimize the computations to be performed by the client. In addition, our method ensures, that the client always needs to store just those parts of the complete model, which are relevant for rendering the current frame.

3.2 Non Photorealistic Rendering

During the last years, researchers realized that in many cases NPR methods are more suitable for conveying information to the user than photorealistic methods, resulting in a growing interest in this area. NPR methods generate pictures that look hand-drawn, whereas individual methods range from painterly rendering (HERTZMANN 2000 a) to abstract, art-like illustrations like "pen-and-ink" (WINKENBACH 1994, HERTZMANN 2000 b), hatching (PRAUN 2001) or sketching (MARKOSIAN 1997). They try to convey specific information to the viewer by employing visual effects like omitting irrelevant parts of the model and accentuating relevant ones. Experiments prove that NPR methods are more successful at achieving this goal than photo-realistic rendering methods. Important examples from the real world are technical illustrations that consist of simple line-drawings. Additional advantages are the reduced computation and rendering costs, since NPR methods concentrate on the important parts of the objects, only.

Our work was strongly influenced by Markosian et al. (MARKOSIAN 1997), who describe a method to generate line-drawings in real-time. They efficiently detect silhouette edges employing a probabilistic detection algorithm and they determine their visibility with a hidden-line-removal algorithm (see APPEL 1967). This algorithm is described in detail in the next section.

3.3 Progressive Meshes

Highly detailed geometric models are commonly used in various computer graphics applications like terrain rendering, virtual environments, automotive prototyping, simulation and medical applications. Their huge sizes impose problems for rendering, transmission and storage, which were and still are tackled by recent research results. One group of methods for managing polygonal models are mesh simplification methods, which reduce the number of geometric primitives in the model while preserving the overall shape and appearance. An overview of existing methods can be found in HECKBERT 1997, more recent results include HOPPE 1997, GARLAND 1998, KLEIN 1998 a, KLEIN 1998 b, LINDSTROM 2000 and KLEIN 2001. Our work uses the Progressive Mesh representation, which was first introduced by HOPPE 1996, since it allows:

- efficient storage,
- progressive transmission and
- view-dependent selection of continuous LOD

for triangular meshes. Further details can be found in his paper.

4. NPR rendering primitives

Silhouettes and contour-lines convey important information concerning the shape and the appearance of an object and thus are essential in any line-drawing. For smooth surfaces, the silhouette can be defined as those points, whose surface normal is perpendicular to the view direction. For polygonal meshes, the definition can be specialized as follows: all edges adjacent to a front-facing and a back-facing face are part of the silhouette. Several algorithms were developed that detect those edges, either in image space or in object-space (several methods for both approaches are discussed in HERTZMANN 1999 and GOOCH 2001).

4.1 Probabilistic Silhouette Search

Our algorithm is based on MARKOSIAN 1997, who first took advantage of the fact, that - on the one hand - only a tiny fraction of all existing edges are part of the silhouette, but that - on the other hand - silhouette edges are grouped in long chains. The method works by testing some fraction of the edges of the model. Whenever an edge is found to be part of the silhouette, the neighboring edges are tested recursively for being part of the silhouette as well.

To speed up the algorithm and to improve its accuracy, two facts can be utilized. First, the likelihood of being a silhouette edge is proportional to the dihedral angle between the neighboring faces of an edge. Thus, sorting the edges of the model based on this angle, the search can be improved. Second, silhouettes feature frame-to-frame coherence, meaning that edges are likely to be part of the silhouette in the current frame if they were part of the silhouette in the previous frame. Thus, storing the silhouette edges per frame and retesting them in the following frame can improve the algorithm. Having extracted as many silhouette edges per frame as possible, the algorithm first determines their visibility by e.g. applying the modified Appel's algorithm (described in subsection 4.4) and then drawing all visible silhouette edge segments.

The method we implemented trades accuracy for speed: the more time we spend on the search, the more silhouette edges are found. Since visibly important, long silhouettes are very likely to be found, the algorithm leads to nice results, especially if the model consists of smooth surfaces – which is the case in most terrains, at least if seen from some distance.

4.2 Crease and Border Edges

In addition to the silhouette edges, we identify crease and border edges, since they represent key features of a model and thus provide important information. An edge is called a crease edge, if the dihedral angle exceeds a predefined threshold (our results suggest to choose 60°) and a border edge if it is adjacent to a single face only. For non-deformable meshes, crease and border edges can be determined as a preprocessing step, for progressive meshes, they have to be recalculated after each edge-collapse or vertex-split operation. Fortunately, the operators cause local changes to the model, only. As a result, the recalculations are limited to the neighborhood (the 1-ring) of the modified edge. We compute crease and border edges by first applying a brute-force detection algorithm to the base mesh that consists of few edges only, and then iteratively recompute these feature edges whenever an LOD modifying operator is applied to the mesh.

4.3 Progressive Meshes

We decided to support as well the possibility to transmit entire models to the client as Progressive Meshes, since triangular renderings are interesting by both themselves and in combination with NPR renderings. They provide two interesting and helpful features:

- They enable progressive transmission by first submitting a base mesh consisting of few vertices and faces only, and then subsequently sending refinement operations. Thus, a first, coarse representation of the object can be displayed very quickly even at limited bandwidth and even if the whole model consists of huge amounts of data. In a system, where the client frequently request models from the server on demand (e.g. for navigation in 3D town models where the client frequently requests data based on his position in town), this representation permits to render real-looking images of the models at any time.
- They provide adaptive LOD, which allows to smoothly balance the complexity of the model against the required frame rate.

Figure 1 provides an example of a progressively transmitted, triangular mesh.

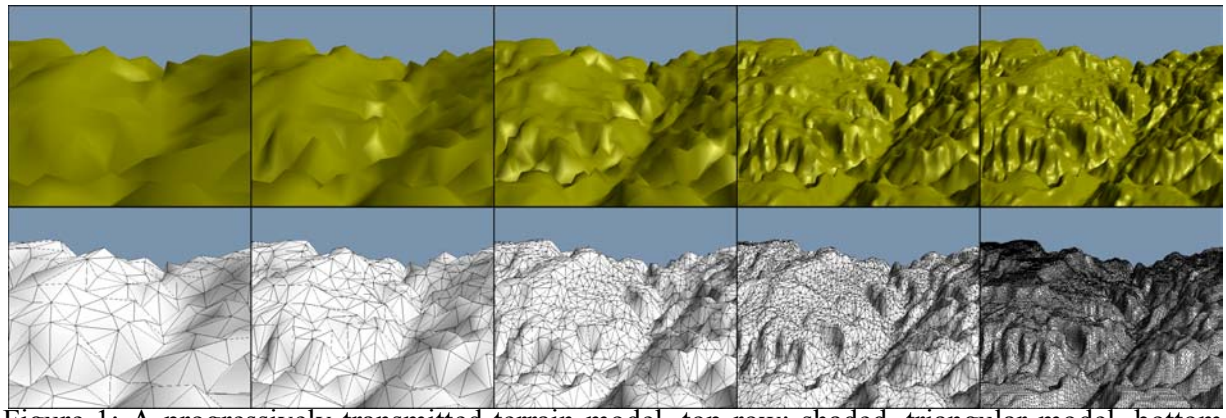
Figure 1: A progressively transmitted terrain model, top row: shaded, triangular model, bottom row: triangles in current LOD, from left to right: the base mesh with 400 faces, refined meshes with 1000, 4000 and 16000 faces, and the final mesh with 130050 faces

4.4 Visibility Determination

In order to determine the visibility of rendering primitives, we implemented two different algorithms. First we implemented the Z-Buffer algorithm (see e.g. WOO 1999), a very simple image space algorithm, which is available in hardware on standard graphics boards for desktop computers. Since this method is not well suited for line-drawings, we additionally implemented Appel's hidden line removal algorithm (APPEL 1967), which determines the quantitative invisibility (QI) for every point on the edges. The QI describes, how many front-facing triangles are situated in between the viewer and the point, thus edge segments with QI zero are visible. Following an edge, the QI can only change if the edge vanishes behind some silhouette edge or appears from behind such an edge.

5. Implementation Details

We implemented a client-server rendering system consisting of a standard PC (server) and a standard PDA (client) without any specialized graphics acceleration hardware, employing Bluetooth interfaces for fast, wireless transmission of rendering primitives (lines or triangles).

Our transmission protocol assumes a reliable channel between the client (the PDA) and the model-server and can utilize either wired (e.g. Ethernet) or wireless (e.g. infra-red or Bluetooth) communication. We implemented and tested several rendering methods, which can be combined with each other. If the models are rendered as lines, the server computes the visible silhouette edge segments (and if desired the visible crease and border edge segments) based on the client's position and view direction. It then selects a subset of them based on the available bandwidth, the total number of visible edge parts and their individual lengths and sends the selected ones to the client for each frame that is to be rendered. If rendering Progressive Meshes, the meshes are incrementally transmitted to the client, who provides either the option to manually select the desired LOD or to specify a minimal frame rate, in which case the appropriate LOD is selected by the system automatically by monitoring the rendering speed of the current frame and adjusting the LOD accordingly.

Since existing PDAs lack floating-point units (FPUs) due to reduced production costs and power savings, we decided to utilize fixed-point arithmetic. The loss of accuracy when turning from floating-point to fixed-point numbers is not visible in the rendered models. By employing this technique, we achieved a speed-up of about factor six.

Another problem we had to face was the lack of an existing graphics library for PocketPCs, which provides very fast drawings of lines, even though many implementations exist (all of them are optimized to render triangles). We used the Extremely Fast Line Algorithm (LIN 2001) for drawing lines, which is substantially faster than the implementations for drawing lines that exist in available graphics libraries for the PocketPC. To render triangular meshes, we additionally implemented a scan-line rasterization algorithm for polygons (see e.g. FOLEY 1997).

6. Results

Our implementation runs on a system consisting of a 1.8GHz PC with Pentium 4 processor and a GeForce 3 graphics cards and a Compaq iPaq 3870 PDA with 206MHz Strong Arm Processor, which lacks both a FPU and specialized graphics hardware. The client's display has a resolution of 320 × 240 pixels. The left part of figure 2 shows the view of a landscape-model, which is rendered with our line-drawing algorithm. Additional information was added to the image by rendering the names of interesting sites. The image contains about 1700 silhouette and crease edges and renders at about 8 frames per second. The right part of figure 2 shows a town-model, drawn with about 300 silhouette and crease edges. The whole scene renders at about 20 frames per second. In both pictures, silhouettes are drawn as black and crease edges as gray lines.

Figure 2: left: line-drawing of a landscape-model enhanced with textual information, right: line-drawing of a city model

7. Summary and Future Work

In this work, we described our implementation of a non-photorealistic, client-server rendering system achieving close to real-time frame rates even for complex models. We showed, how to overcome the limits of current mobile devices that lack an FPU as well as specialized graphics hardware. We showed that NPR methods are well suited within a client-server system, since they allow rendering huge models that don't fit into the PDA's memory and since they convey important information to the user more successfully than photorealistic rendering methods, especially if relatively small displays are employed.

In the future, we plan to improve our method by implementing a data compression scheme to increase the transmission speed of models even more. In addition, the technical feasibility and suitability of different rendering styles will be evaluated on PDAs and especially for the new generations of processors, which provide significantly increased computational power.

A different, but very promising attempt we will investigate are point-based rendering techniques, which might be highly suitable due to the simplicity of their rendering primitives.

8. References

Appel, A., 1967: The notion of quantitative invisibility and the machine rendering of solids, In Proceedings ACM Natl. Mtg., p. 387, Thompson Books.

Foley, J., van Dam, A., Feiner, S., Hughes, J., 1997: Computer Graphics, Principles and Practice, Second Edition in C. Addison Wesley, pp.91-99

Funkhouser, T., 1995: RING: A client-server system for multi-user virtual environments. In: 1995 Symposium on Interactive 3D Graphics, pp. 85-92, ACM Siggraph

Garland, M., Heckbert, P., 1998: Simplifying Surfaces with Color and Texture using Quadric Error Metrics. In: Proceedings of IEEE Visualization '98

Gooch, B., Gooch, A., 2001: Non-Photorealistic Rendering. A K Peters Ltd., 1st edition, pp. 117-146

Heckbert, P., Garland, M., 1997: Survey of Polygonal Surface Simplification Algorithms, Technical Report, CS Dept., Carnegie Mellon University, to appear

Hertzmann, A., 1999: Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines. In: Non-Photorealistic Rendering, ACM Siggraph 99 Course Notes

Hertzmann, A., Zorin, D., 2000 (a): Illustrating Smooth Surfaces. In: ACM Siggraph 2000, Computer Graphics Proceedings, pp. 517-526, ACM Press

Hertzmann, A., Perlin, K., 2000 (b): Painterly Rendering for Video and Interaction. First International Symposium on Non-Photorealistic Animation and Rendering (2000), pp. 7-12

Hesina, G., Schmalstieg, D., 1998: A network architecture for remote rendering. In: Proceedings of Second International Workshop on Distributed Interactive Simulation and Real-Time Applications, pp. 88-91

Hoppe, H., 1996: Progressive Meshes. In: ACM Siggraph 1996, Computer Graphics Proceedings, pp. 99-108, ACM Press

Hoppe, H., 1997: View-Dependent Refinement of Progressive Meshes. In: ACM Siggraph 1997, Computer Graphics Proceedings, pp. 189-198, ACM Press

Klein, R., 1998 (a): Multiresolution representations for surfaces meshes based on the vertex decimation method. In: Computers and Graphics, Vol. 22, No. 1, pp. 13-26

Klein, R., Cohen-Or, D., Hüttner, T., 1998 (b): Incremental view-dependent multiresolution triangulation of terrain. In: The Journal of Visualization and Computer Animation, Vol. 9, pp. 129-143

Klein, R., Schilling, A., 2001: Efficient Multiresolution Models. In: Festschrift zum 60. Geburtstag von Wolfgang Straßer, 2001, pp. 109-130

Lin, P., 2001: Extremely Fast Line Algorithm. http://www.edepot.com/algorithm.html

Lindstrom, P., Turk, G., 2000: Image-Driven Simplification. In: ACM Transactions on Graphics, Vol. 19 No. 3, July 2000, pp. 204-241

Mann, Y., Cohen-Or, D., 1997: Selective Pixel Transmission for Navigating in Remote Virtual Environments. In: Computer Graphics Forum, Vol. 16, No.3, pp. 201-206

Markosian, L., Kowalski, M., Trychin, S., Bourdev, L., Goldstein, D., Hughes, J., 1997: Real-Time Non-Photorealistic Rendering. In: ACM Siggraph 1997, Computer Graphics Proceedings, pp. 415-420

Praun, E., Hoppe, H., Webb, M., Finkelstein, A., 2001: Real-Time Hatching. In: ACM Siggraph 2001, Computer Graphics Proceedings, ACM Press, pp. 579-584

Schneider, B., Martin, I., 1999: An adaptive framework for 3D graphics over networks. In: Computers and Graphics, Volume 23, No. 6, pp. 867-874

Teler, E., Lischinski, D., 2001: Streaming of Complex 3D Scenes for Remote Walkthroughs. In: Computer Graphics Forum, Vol. 20, No.3, pp. 17-25

Winkenbach, G., Salesin, D., 1994: Computer-Generated Pen-and-Ink Illustration. ACM Siggraph 1994, Computer Graphics Proceedings, ACM Press, pp.91-100

Woo, M., Neider, J., Davis, T., Shreiner, D., 1999: OpenGL Programming Guide, Third Edition, Addison Wesley, p.176