INTEGRATED MULTIRESOLUTION MODELING OF TERRAIN GEOMETRY AND TERRAIN TEXTURE DATA

Jürgen Döllner
University of Muenster

Abstract

The increasing capabilities of today's 3D graphics hardware raise the question how visualization software can take actually advantage of these developments. In this paper, an approach for integrating multiresolution representations of terrain geometry and terrain texture is presented which provides high image quality and high rendering speed focussing on PC graphics hardware. A terrain is modeled by a regular grid which can be partially refined by local TINs in order to represent morphologically complex terrain parts. The multiresolution model for texture data of a terrain and the multiresolution model for its geometry data are closely related: The rendering algorithm selects geometry terrain patches based on a criterion which takes into account geometry and texture approximation errors. Multiple texture hierarchies, which may represent different thematic information layers, can be bound to a terrain model and rendered using multitexturing. For example, the terrain shading can be provided by a pre-calculated shading texture which permits the implementation of different shading schemes and improves visual quality compared to a geometry-based shading.

Introduction

The increasing capabilities of today's 3D graphics hardware raise the question how visualization software, for example terrain visualization systems, can actually take advantage of these developments in order to provide higher image quality and rendering speed. Texture memory and texture paging band width are increasing, and multitexturing gets available even on low-cost hardware.

In this paper, an approach for integrating multiresolution representations of terrain geometry and terrain texture is presented. It provides high image quality and high rendering speed based on a new data structure and texture-based rendering. The data structure represents terrain models; they play a central role in many kinds of virtual environments as fundamental tools to present and communicate spatial information. Various hierarchical data structures have been developed or applied for representing terrains, e.g., hierarchical TINs (DeFloriani et al., 1998), R-trees (Kofler et al., 1998), restricted quadtree triangulations (Pajarola, 1998), and progressive meshes (Hoppe, 1998). Each multiresolution modeling scheme supports a specific type of input data such as arbitrarily distributed data points for triangulated irregular networks (TINs) or regularly distributed data points for grids. In general, real-world terrain data sets are composed of data of different types. For example, a cartographic terrain model can include grid data describing the digital elevation model (DEM) and additional TINs describing structures at a finer resolution such as topographically complex or interesting terrain parts (e.g., streets as illustrated in Figure 1).

Another important category of terrain data are texture data. Frequently, textures are used to visualize thematic information such as temperature values or land use information. For real-world terrain models, large, high-resolution textures need to be processed which do not fit into graphics texture memory or even into main memory. Therefore, multiresolution modeling is necessary for texture data as well.

Since geometry data and texture data are tightly coupled, we can expect that they are best handled by an integrated multiresolution model.

Presented at the IMAGE 2000 Conference
Scottsdale, Arizona 10-14 July 2000.

Related Work

A large number of multiresolution modeling techniques for arbitrary polygonal meshes and, in particular, for digital terrains have been developed in the past. Hierarchical triangulations based on TINs have been applied to generate multiresolution models which can be used by level of detail algorithms (De Floriani et al., 1998; Xia et al., 1997). Regular grids have been used for multiresolution modeling (Falby et al., 1993) and for real-time, continuous LOD rendering (Duchaineau et al., 1997; Lindstrom et al., 1997; Pajarola, 1998; Hoppe, 1998). Chen (1999) discussed a method for combining LOD techniques with image-based modeling and rendering techniques to take advantage of the frame-to-frame coherence in screen-space.

As a common characteristic, each LOD technique operates either on grids or TINs, i.e., they do not permit the integration of geometry data of different topological type into one multiresolution model. In our approach a hybrid terrain model is used because this allows for combining different data sources (e.g., Douglas, 1986; Fowler and Little, 1979) and to maintain the original semantics without having to standardize or convert data into a uniform representation.

Most LOD techniques concentrate on an efficient internal geometric representation in order to achieve a high rendering performance. However, in many applications the visual quality with respect to topographic terrain features or thematic terrain data is as important as rendering performance. Recent developments (e.g., Hoppe, 1999; Xia, 1997) take into account the visual quality by considering surface normals, but do not provide an explicit control of the terrain shading as proposed in this paper.

Furthermore, most LOD techniques are limited with respect to the management of large-scale texture data: no analogous LOD mechanism is provided for terrain-related texture data. Lindstrom (1995) proposed a method which handles a single large-scale texture related to a LOD terrain geometry. However, an efficient treatment of multiple textures, which are applied to a single terrain using different techniques such as multitexturing, is required for the interactive exploration and manipulation of that data (e.g., terrain visualization used for landscape analysis and planing).

Hybrid Terrain Models

In many applications, digital elevation models are represented by regular grids. While their memory requirements and implementation complexity are low, they do not approximate well terrain details. TIN-based approaches are adaptive to those details, but involve high memory requirements and preprocessing overhead. Handling the details as precisely as possible is important because we perceive a terrain model basically by the terrain shading and terrain silhouette, and both depend on geometric details.

Characteristics of Hybrid Terrain Models

We constitute a *hybrid terrain model* by a regular grid, called the *reference grid*, and a collection of TINs which are associated to grid cells and refine the terrain representation within the cell domain (see Figure 1). For such *refined grid cells*, the following properties must hold:

- The domain of the TIN is equal to the domain of the grid cell.
- The height values at the corners of the grid cell are equal to the height values of the corresponding TIN vertices at the corners.
- If the TIN has a vertex at the border of the grid cell (except the corners), than the neighbor grid cell must have a TIN and that TIN must have a vertex with the same height at that position, too.

These properties ensure that TINs refining a grid cell adapt themselves continuously to the terrain represented by neighbor cells and neighbor TINs.

Grid cells are refined where a complex morphology has to be represented, for example, in the case of riverbeds, streets, or ridges (Buziek and Kruse, 1992). This hybrid terrain model, roughly speaking, combines the advantages of both grids and TINs: it leads to a memory-efficient and morphologically precise terrain representation. To generate hybrid terrain models, high-resolution regular grids can be preprocessed by analyzing their morphology and constructing microstructures.

In our experience, about 15% of the reference grid cells of real-world terrain data sets exhibit morphologically complex parts. Compared to a pure TIN representation, the remaining 85% of the terrain domain can be represented by grids without loss of visual quality.

Generic Multiresolution Data Structure

This section defines a multiresolution model for geometric data, the *approximation tree*, which is generic with respect to the type of terrain data as required by hybrid terrain models. Let $P = \{p_1, \ldots, p_n\}$ be a set of data points in the xy-plane. Let $a_{\min}$ and $a_{\max}$ be the extremal points of the axis-parallel bounding box of *P*. Let $G = (P, h_G)$ be a terrain model defined by the set of points *P* and an elevation function h_G. The domain of *G* is defined as:

$$D(G) := \{(x,y) \in \Re^2 \mid x(a_{\min}) \le x \le x(a_{\max}), y(a_{\min}) \le y \le y(a_{\max})\}$$

The elevation function $h_G : D(G) \to \Re$ calculates elevation values for points of $D(G)$ by interpolating the values for data points of *P*.

An *approximation tree* $A_{s,d}(G)$ for a terrain model *G* is represented by a tree; its nodes are called *geometry patches*. Each geometry patch *N* represents a rectangular region $D(N) \subseteq D(G)$ and approximates the terrain surface in that region by an approximating terrain surface $G_N = (P_N, h_{G_N})$. The set P_N consists of at most *s* data points: $|P_N| \le s$. Furthermore, the four corner points of $D(N)$ must be contained in P_N. The way the node calculates the data points P_N depends on the *approximation strategy* adopted by the node. For example, a grid-based node will select evenly spaced points from a grid data set, whereas a TIN-based node will select points from an arbitrary data set based on an error criterion.

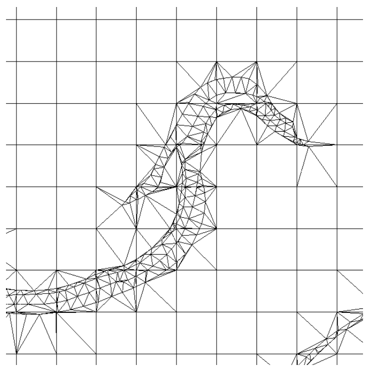
Figure 1: Part of a hybrid terrain model.

The *geometric approximation error* ε_N of a geometry patch N is defined by the maximal vertical distance between the terrain models G_N and G:

$$\varepsilon_N := \max_{p \in D(N)} \left| h_{G_N}(p) - h_G(p) \right| .$$

Each terrain patch N can have at most d child nodes. The child nodes are constructed as follows: If the geometric approximation error ε_N exceeds a certain threshold $\varepsilon \geq 0$, the domain $D(N)$ is subdivided into a set of at most d rectangular and disjoint subdomains $D(N_i)$ such that

$$\bigcup D(N_i) = D(N) .$$

For each subdomain $D(N_i)$, a child node N_i of N is constructed which approximates the terrain surface in that subdomain. The domain of the root node of the approximation tree $A_{s,d}(G)$ is $D(G)$ covering the whole domain of the terrain G. The definitions so far generalize several hierarchical terrain representations, for example, quad trees and two-dimensional binary trees.

Approximation Strategies

Grid-based and TIN-based geometry patches require different approximation strategies. The concrete strategy is part of a node's behavior, i.e., the node class defines the algorithm for subdividing the terrain domain and the strategy for selecting appropriate data points. For hybrid terrain models, the following strategies have been implemented:

- *Grid approximation:* This strategy is based on selecting points of a regular grid by specifying the equidistant rows and columns the node actually uses. The reference grid is stored only once because it can be multiply referenced by nodes.
- *TIN approximation*: This strategy inserts those points into an initially empty TIN, which cause the largest approximation errors. The insertion process is stopped if the approximation error becomes zero or if the TIN reaches the maximum number s of points (e.g. van Kreveld, 1997).

Both strategies are static: they represent a geometry patch at one concrete level of detail. It would be possible to use dynamic approximation strategies such as the restricted quadtree triangulations (Pajarola, 1998) or the progressive meshes (Hoppe, 1998) within the same approximation tree by implementing specialized node classes.

Multiresolution Models for Textures

Multiresolution modeling for texture data in the context of multiresolution terrain models is motivated by the following observations:

- Visualization applications are likely to have texture data up to several hundreds of megabyte, which is, for example, typical in Cartography. However, graphics hardware will always impose constraints on the size of textures. Common OpenGL hardware, for example, limits the texture size by 2048x2048 pixels and requires a texture size of a power of 2.
- An explicit modeling of texture pyramids introduces more freedom in the selection and calculation of levels of detail. For each level the textures can be calculated with different aspects to permit anisotrophic texturing. In addition, textures can be computed on demand.
- The selection of a level-of-detail texture can be correlated with the geometric approximation error, i.e., together with a texture approximation error we can define a criterion for the visual quality of terrain visualization.

Multiresolution Data Structure for Terrain Textures

Let T be a geo-referenced 2D texture used for a terrain model G. The texture pyramid $\widehat{T}$ of a terrain texture T consists of a sequence of textures T_i with decreasing resolution. Conceptually, each texture is created by scaling down the predecessor texture by a factor of ½. The first texture of the sequence is the original terrain texture, the last texture fulfills the constraints imposed by the rendering system and graphics hardware. All textures in a texture pyramid have the same domain as the original texture.

In analogy to the approximation tree for geometric data, we can now define a similar tree for multiresolution textures, the *approximation tree for terrain textures* $A_{s,d}(G,T)$. A tree $A_{s,d}(G,T)$ consists of nodes M, called *texture patches*. A texture patch has the following properties:

- A texture patch M is associated with exactly one geometry patch $N_M \in A_{s,d}(G)$.
- The domain of M covers the domain of N_M: $D(M) \supseteq D(N_M)$.
- M references a part S_M of an image of the texture pyramid $\widehat{T}$, whereby S_M is the image part with the highest resolution in $\widehat{T}$ which completely covers the domain $D(N_M)$ and fulfills the constraints of the rendering system.
- If S_M is not an image part of the first texture in the texture pyramid $\widehat{T}$ (i.e., it is not part of the original terrain texture T) and the geometry patch N_M has child nodes, than the texture patch M has child nodes, too.
- If the texture patch M has child nodes, than the number of child nodes and their corresponding domains are equal to the child nodes and their domains in the associated geometry patch N_M.

The domain of a texture patch M covers at least the domain of its associated geometry patch N_M. This restriction has been imposed because texture switches (like all other context changes) are costly. The texture resolution of *M* is considered optimal for the geometry patch N_M. The geometry patch can also be rendered with any parent texture patch V_M of the texture patch *M* because its domain covers the domain of *M*, too. In such a case, the texture resolution is probably non-optimal for the geometry patch. But if the geometry patch is far away from the viewer this reduction of resolution is not visible and can speed up rendering since less texture data have to be processed from the rendering system. We take care of this improvement by calculating the visual texture error γ_M in the algorithm presented in next section.

The texture hierarchy and graphics implementation requires rectangular textures. For that reason, rectangular domains for geometry patches have been chosen.

Efficient Handling of Texture Data

In general, it is unlikely for real-world data sets that texture data fits into main memory (e.g., cartographic high-resolution textures). We need to handle such large texture data efficiently. The following mechanisms are deployed:

- *File-memory mapping,* which refers to the ability of operating systems to blend files into the process address space, permits to access file data by pointers and therefore by memory-related functions (e.g., memcpy). Thus, the handling of file data does not differ to that of memory-resident data. This way texture files can be bound to an OpenGL texture object directly. In contrast to the virtual memory mechanism, file-memory mapping does not burden with the swap space, avoiding costly writing operations.
- *Texture subimages* permit to reuse large textures: Different and possibly overlapping image parts can refer to a source texture without duplicating its contents. For example, texture objects in OpenGL can select parts of an image by specifying row length, row offset and column offset.

Rendering Algorithm

The rendering algorithm traverses an approximation tree recursively, calculates visual approximation errors, and selects geometry patches and texture patches based on that quality criterion.

Visual Approximation Errors

Let $A_{s,d}(G)$ and $A_{s,d}(G,T)$ be a geometry approximation tree and an associated texture approximation tree. Let $N \in A_{s,d}(G)$ be a geometry patch with an approximating terrain surface G_N, and let $M \in A_{s,d}(G,T)$ be a texture patch, whereby $D(M) \supseteq D(N)$. Let B_N be the 3D axis-parallel bounding box of G_N.

Visual approximation errors can be defined if the bounding box B_N intersects the current view volume.

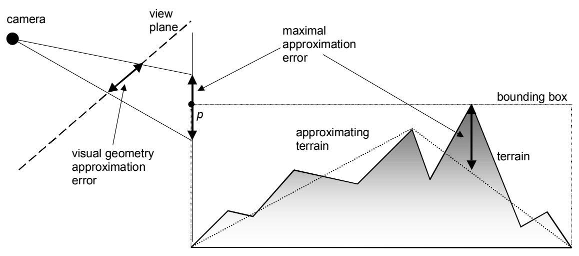
camera
view plane
maximal approximation error
bounding box
p
approximating terrain
terrain
visual geometry approximation error

Figure 2. Visual geometry approximation error and its calculation.

If the bounding box intersects, then the point *p* of the bounding box closest to the camera and inside the view volume can be determined. The visual approximation errors are calculated as follows:

- *Visual geometry approximation error:* A line segment parallel to the z-axis (the direction of elevation) of length of the approximation error ε_N is constructed which is centered at *p*. Projecting that segment onto the view plane, the visual geometry approximation error α_N is the length of the projected line segment measured in pixels (see Figure 2).
- *Visual texture approximation error:* Determine the width *w* and height *h* (in terrain coordinates) of a texel of the texture patch *M*. Construct a rectangle with the extension *w* and *h* parallel to the *xy*-plane centered at *p*. Projecting that rectangle onto the view plane, the visual texture approximation error γ_M is the area of the projected rectangle measured in pixels. γ_M describes how much the area of the projected rectangle differs from a single pixel.

The visual approximation errors α_N and γ_M are a measure for the visual quality of a geometry patch *N* and a texture patch *M*. If we render the approximating terrain surface G_N, it is ensured that each of the pixels of G_N differs by at most α_N pixels compared to the original terrain *G*. In analogy, for the texels of the approximating terrain texture *M*, we can guarantee that the texels are sufficiently dense.

Recursive Rendering of Approximation Trees

A hybrid terrain model *G* together with a terrain texture *T* which are represented by the approximation trees $A_{s,d}(G)$ and $A_{s,d}(G,T)$ is rendered recursively, starting with the root pair (N_0, M_0). $N_0 \in A_{s,d}(G)$ denotes the root geometry patch and $M_0 \in A_{s,d}(G,T)$ the root texture patch. For a patch pair (N, M), the algorithm works as follows:

1. If the bounding box B_N of the current geometry patch does not intersect the current view volume, the recursive rendering stops at this point (hierarchical view-volume culling).

2. Otherwise, the visual approximation errors α_N and γ_M are calculated. If α_N is larger than a user-defined geometric threshold or if γ_M is larger than a user-defined texture threshold, then the rendering calls itself recursively for the child patch pairs (N_i,M_i). If M has no child texture patches M_i, M is used instead.
3. If both errors are ok, assign to M the parent texture patch V_M of M until the visual texture approximation error γ_{V_M} exceeds the user-defined texture threshold (a texture patch can always use – by definition – the texture of its parent). The resolution of the parent's texture is lower but this way we guarantee that the resolution comes close to the user-defined texture threshold.
4. Render the approximating terrain surface G_N together with the texture determined in the previous step, and finish the recursion.

The thresholds for both errors allow the user to prioritize either interactivity or visual quality. Low thresholds lead to higher visual quality, higher thresholds accelerate rendering. Step 3 of the algorithm ensures that the selected texture patch matches the resolution needed for projecting it onto the screen without a visual distortion, but with as less texture data to be processed as possible.

Gaps between two adjacent geometry patches are possible. The gaps are at most as big as the user-defined geometry approximation error (given in screen pixels). To close the gaps, walls are inserted between geometry patches of different level-of-detail. The walls have the same texture coordinates as the edge between the two geometry patches, so they get colored the same way and are visually not recognizable for reasonable geometry approximation errors (commonly smaller than 4 pixels).

Implementation Issues for OpenGL

The geometry of a terrain patch can be stored in a display list because the contents of a geometry patch are static. Display lists accelerate rendering drastically because vertex information is already kept on the graphics hardware side and stored in its internal format. Furthermore, display lists are essential for applications which should run remotely over a network. In this case, geometry data has to be transferred only once. To speed up further the rendering process, texture switches can be minimized by grouping geometry patches along the texture patches they use. In analogy to display lists, textures can be stored by texture objects.

Only a part of the texture pyramid is actually required to render a single frame. Therefore, texture patches load their texture data on demand, spanning a separate thread. While the texture data is being loaded, the texture data of the parent texture patch can be used. In this case, texture resolution is not optimal, but interactivity is ensured because the application is not blocked and can use at least a reasonable approximation of the demanded texture.

Shading by Topographic Textures

In many terrain visualization systems the visual quality depends directly on the geometric resolution of the approximating terrain due to the underlying Gouraud shading. The vertex normals of a triangle determine the shading of the whole triangle, leading to shading artifacts if triangles become large or thin. For a level-of-detail terrain model this implies that the lower the resolution, the more topographic details get lost, and that shading changes if the LOD-dependent geometry changes. In our approach, the terrain shading relies on textures.

Topographic Textures

A *topographic texture* is pre-calculated and applied as regular terrain texture to the terrain model, reintroducing topographic details that might be removed during the geometric simplification process (see Figure 3). A similar approach has been made in the context of appearance-preserving simplification strategies (e.g., Cignoni et al., 1998; Cohen et al., 1998).

Terrain models shaded by topographic textures have the following properties:

1. The visual quality of a terrain model depends on the quality of the topographic textures because

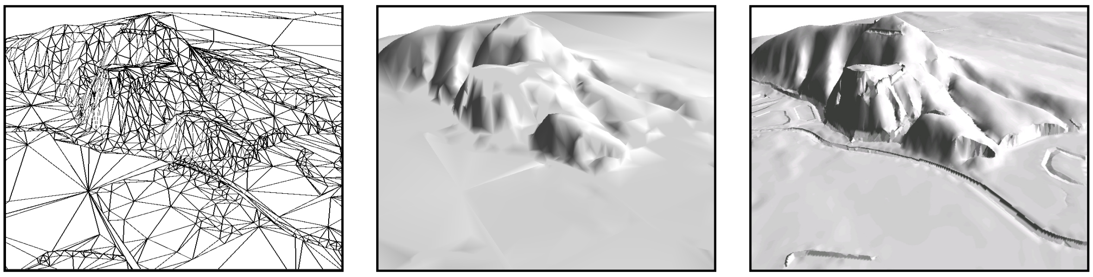
Figure 3. Different views of a terrain model. Wire-frame (left), Gouraud-shaded (center), shaded by topographic texture (right).

they encode visually the terrain's morphology.

2. The geometric complexity needed to achieve high-quality images of terrain models is considerably less compared to non-texture shading because topographic textures are applied pixel-precisely.
3. The visual effects of LOD changes can be minimized by topographic textures because shading via topographic textures "hides" discontinuities in the geometric representation during a change in the LOD.

A topographic texture consists of luminance values and depends on the high-resolution terrain model, the terrain surface properties, the lighting conditions, and on design rules. In cartography, for example, design rules have been developed which facilitate the perception of morphologically important terrain parts such as peaks, pits, valleys and ridges. Note that the ambient and diffuse light, which are the most important ones for terrain shading, does not depend on the camera settings which justifies the pre-calculation.

Generating Topographic Textures

Topographic textures can be calculated efficiently with OpenGL. A high-resolution version of the terrain model is rendered into a hidden frame buffer (e.g., back buffer or offscreen canvas) using Gouraud shading and an orthogonal projection. Then, the contents of the frame buffer are used as topographic texture. If light conditions or terrain properties change, the topographic texture is re-computed. In general, the recalculation is not time-critical because it does not occur during user interaction.
We are not limited to the lighting and shading models provided by the underlying 3D rendering system. Figure 4 shows an automatically generated topographic texture which takes into account self-shadowing of a terrain model. The ambient color of a triangle is modified in response to the result of the shadow test.

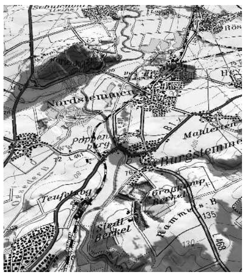

Figure 4. Topographic texture with self-shadowing, combined with a cartographic texture.

Implementation

Since the frame buffer size is limited, we can use a tiling process to generate large-scale topographic textures (e.g., 4096x4096 or even larger) which is used as input for the texture tree. Topographic texture can be stored compactly as 2D luminance texture.
The rendering of a terrain model is accelerated using topographic textures because lighting calculations can be turned off, saving expensive per-vertex calculations. Low-cost 3D graphics hardware, in particular, benefits from topographic textures because their geometry and lighting capabilities are low compared to their texturing capabilities.

Multitexturing for Terrain Visualization

In most applications of terrain visualization, the terrain models are used as 3D maps for visualizing topographic and thematic data. Thematic texturing is responsible for mapping application-relevant information onto the terrain surface. In many cases, two or more thematic textures are represented together on a terrain surface (e.g., topographic texture, road map, and land use information).

Applications of Multitexturing

We can extend an approximation tree by additional texture trees, i.e., a geometry patch can be associated with texture patches of different texture trees. The applications of multitexturing are distinguished into two categories:

- *Combining information layers.* Two or more textures are merged visually in order to combine the corresponding themes. The kind of merging is specified by a 2D image operation such as modulating, adding etc.
- *Specifying visibility of information layers.* A texture can define the region and degree of visibility for other textures. A visibility texture is specified by transparency values.

It is important that each texture remains independent. In general, the textures of several information layers cannot be merged into one final 2D texture due to their different domains and resolutions. In addition, it is not be feasible to merge large-scale textures with several hundreds of megabytes in real-time.
Furthermore, multitexturing facilitates the implementation of dynamic textures and texture-based animations. One or more information layers can recalculate their texture data without affecting the textures of other layers. For example, a luminance texture implementing a highlight lens can be modified in response to user interaction in real-time.

Example: Thematic Lenses

In Figure 5, a luminance texture is used to implement a highlight lens which is combined with a cartographic texture. The luminance texture defines low

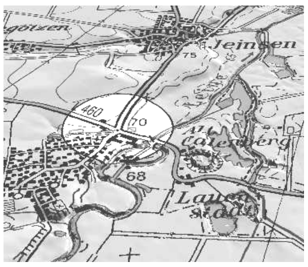

Figure 5. Multiple texture layers: a luminance texture layer, modeling a high-light lens, is superimposed on a cartographic texture layer.

luminance outside a circular region around the focus point, and a high luminance inside that region. The lens is used to control the visual focus in the terrain model and can be interactively moved across the terrain. The luminance texture needs not to be merged again with the high-resolution cartographic texture.

Example: Thematic Animation

The multitexturing approach facilitates the animation of thematic information. Assume we have a sequence of texture "key frames". During the animation, we linearly interpolate between two consecutive texture key frames. The interpolation is done by multitexturing with appropriate weights assigned to both key frame textures. No intermediate texture has to be created which permits to animate even high-resolution texture sequences.

Experimental Results

All screen shots presented in this paper have been taken from the *LandExplorer* (MAM/VRS, 2000), a prototype implementing the concepts described in (Buziek and Döllner, 1999). The time measurements (Figure 6) were performed on a standard PC equipped with a 350 MHz Pentium II processor, 128 MB RAM, Riva TNT graphics card with 16 MB graphics memory, and running Windows NT 4.0 (SP6). The window was 640 x 480 pixels large in true-color. We used a terrain data set consisting of about 500.000 triangles (a 640x320 reference grid plus additional fine-structures introducing about 130.000 triangles). The original data set needs more than 5 seconds to render without level-of-detail techniques. The size of the topographic texture is 2500 x 5000 pixels (13 MB, gray-scale) and the thematic texture is 3200 x 6400 pixels (62 MB, RGB) large. The method using Gouraud shading is the slowest method, even rendering with two textures is almost always faster.

Conclusions

The integrated multiresolution model for terrain geometry and terrain textures facilitates the implementation of interactive terrain visualization software due to its flexible terrain geometry representation and its concepts for multiple texture layers. Its implementation can take advantage of recent improvements of graphics hardware, in particular the texture capabilities can be efficiently used.
The core part, the hybrid terrain model, is well suited for real-world terrain data sets because it allows us to integrate geometry data of different topological type in a uniform multiresolution model that maintains the semantics of the original data sets and supports customized approximation strategies. In particular, the

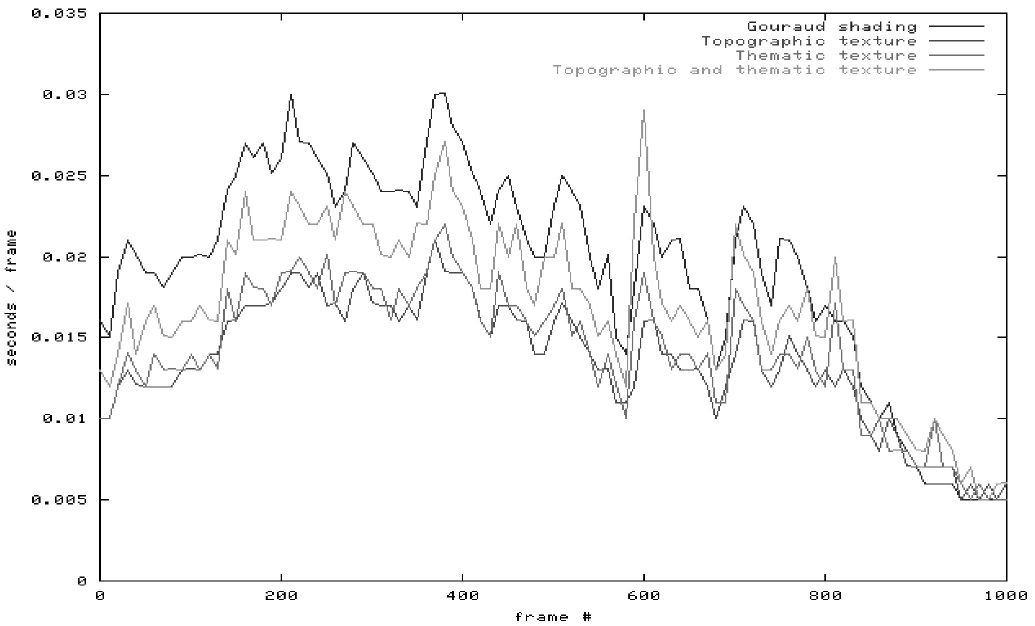

Figure 6. Time measurements for different rendering configurations of hybrid terrain models.

visual quality of terrain visualizations profits from microstructures because they have a great impact on the perception of a terrain model. We plan to automate the extraction of microstructures from high-resolution grids.

Multitexturing and the management of multiple texture hierarchies turned out to be essential for implementing advanced visualization techniques. For example, interactive visual tools can be implemented based on luminance and alpha texture layers. Independent texture layers are required to visualize time-varying and dynamic thematic information. As a direct application of multiple texture layers, shading can be implemented by topographic textures improving the visual quality dramatically because they outwit human perception by providing detailed shading information for a coarse, dynamic geometry. In addition, topographic texturing allows for using non-standard shading algorithms and takes fully advantage of accelerated texture mapping found in today's 3D graphics hardware.

In the future, terrain visualization could benefit from 3D graphics hardware which offers sophisticated multitexturing and permits to program the texturing stage in the rendering pipeline.

References

K. Baumann, J. Döllner, K. Hinrichs, O. Kersting (1999). A Hybrid, Hierarchical Data Structure for Real-Time Terrain Visualization. Proceedings Computer Graphics International '99, 85-92.

G. Buziek, J. Döllner (1999). Concept and Implementation of an Interactive, Cartographic Virtual Reality System. Proceedings of the International Cartographic Conference ICC '99, Ottawa, 637-648.

G. Buziek, I. Kruse (1992). The DTM-System TASH in an Interactive Environment. EARSeL Advances in Remove Sensing, 1(3):129-134.

B. Chen, J. Swan, E. Kuo, A. Kaufman (1999). LOD-Sprite Technique for Accelerated Terrain Rendering. Proceedings IEEE Visualization '99.

P. Cignoni, C. Montani, C. Rocchini, R. Scopigno (1998). A general method for preserving attribute values on simplified meshes. Proceedings Visualization '98, 59-66.

J. Cohen, M. Olano, D. Manocha (1998). Appearance-Preserving Simplification. Proceedings of SIGGRAPH '98, 115-122.

L. De Floriani, P. Magillo, E. Puppo (1998). Efficient Implementation of Multi-Triangulations. Proceedings IEEE Visualization '98.

D. Douglas (1986). Experiments to locate ridges and channels to create a new type of Digital Elevation Model. Cartographica, 23(4):29-61.

M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, M. Mineev-Weinstein (1997). ROAMing Terrain: Real-time Optimally Adapting Meshes, Proceedings Visualization '97, 81-88.

J. Falby, M. Zyda, D. Pratt, R. Mackey (1993). NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation. Computers & Graphics, 17(1):65-69.

R. Fowler, J. Little. Automatic extraction of irregular network digital terrain models (1979). Computer Graphics (SIGGRAPH '79 Proceedings), 13(2):199-207.

H. Hoppe (1998). Smooth View-Dependent Level-of-Detail Control an its Application to Terrain Rendering. Proceedings Visualization '98, 35-42.

H. Hoppe (1999). New quadric metric for simplifying meshes with appearance attributes. Proceedings IEEE Visualization `99, 59-66.

M. Kofler, M. Gervautz, M. Gruber (1998). The Styria Flyover - LOD management for huge textured terrain models. Proceedings Computer Graphics International 1998, 444-454.

MAM/VRS (2000). LandExplorer. WWW-Site: http://www.mamvrs.de/geovisualiz.htm

P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, G. Turner (1996). Real-Time, Continuous Level of Detail Rendering of Height Fields. Proceedings SIGGRAPH '96, 109-118.

P. Lindstrom, D. Koller, L. Hodges, W. Ribarsky, N. Faust, G. Turner (1995). Level-of-detail Management for Real-Time Rendering of Phototextured Terrain. GVU Technical Report 95-06.

R. Pajarola (1998). Large Scale Terrain Visualization using the Restricted Quadtree Triangulation. Proceedings IEEE Visualization '98, 19-26 & 515.

M. van Kreveld (1997). Digital Elevation Models and TIN Algorithms. In: M. van Kreveld, J. Nievergelt, T. Roos, and P. Widmayer (Eds.), Algorithmic Foundations of Geographic Information Systems, Lecture Notes Computer Science, Springer-Verlag, 1340:37-78.

J. Xia, J. El-Sana, A. Varshney (1997). Adaptive Real-Time Level-of-detail-based Rendering for Polygonal Models. IEEE Transactions on Visualization and Computer Graphics '97, 3(2):171-183.

Appendix: Image Plates

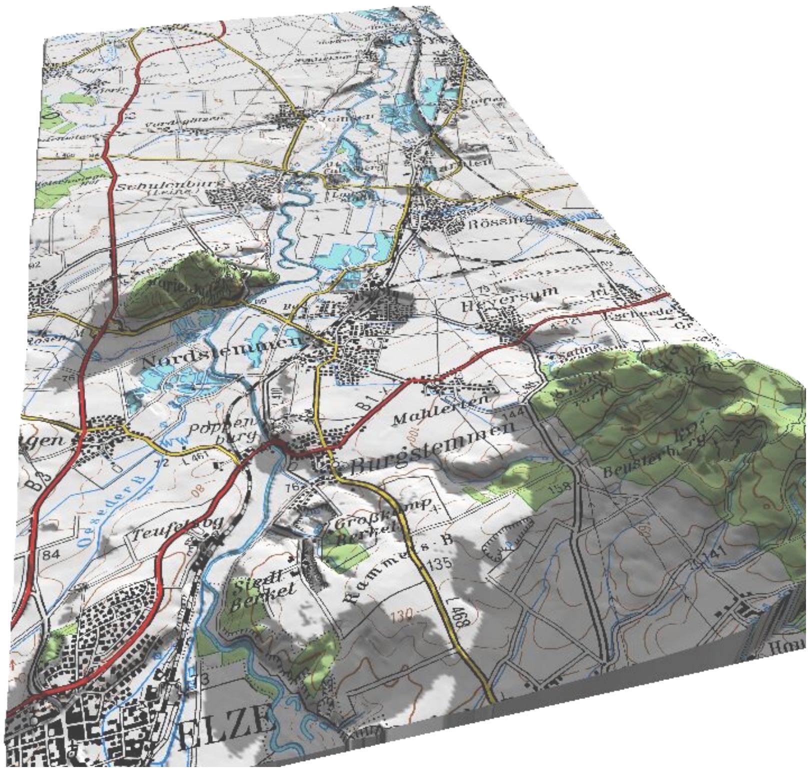

Figure 4: Topographic texture with self-shadowing, combined with a cartographic texture.

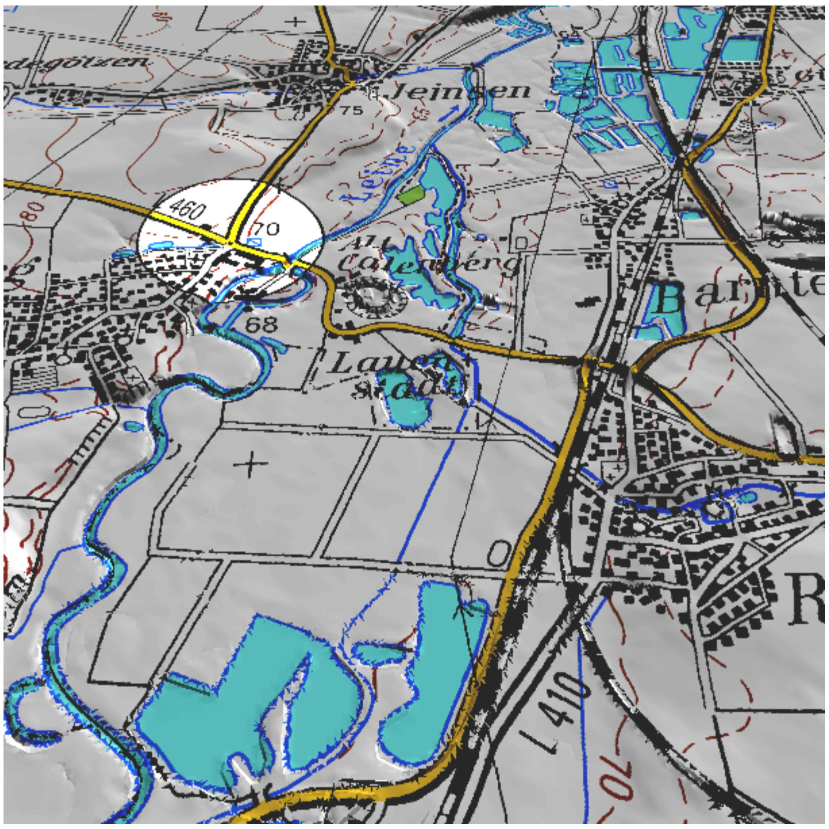

Figure 5: Multiple texture layers: a luminance texture layer, modeling a highlight lens, is superimposed on a cartographic texture layer.