Dynamic Mapping of Raster-Data for 3D Geovirtual Environments

Matthias Trapp, Jürgen Döllner
Hasso-Plattner-Institute, University of Potsdam, Germany
{matthias.trapp, juergen.doellner}@hpi.uni-potsdam.de

Abstract

Interactive 3D geovirtual environments (GeoVE), such as 3D virtual city and landscape models, are important tools to communicate geo-spatial information. Usually, this includes static polygonal data (e.g., digital terrain model) and raster data (e.g., aerial images) which are composed from multiple data sources during a complex, only partial automatic pre-processing step. When dealing with highly dynamic geo-referenced raster data, such as the propagation of fires or floods, this pre-processing step hinders the direct application of 3D GeoVE for decision support systems. To compensate for this limitation, this paper presents a concept for dynamically mapping multiple layers of raster for interactive GeoVE. The implementation of our rendering technique is based on the concept of projective texture mapping and can be implemented efficiently using consumer graphics hardware. Further, this paper demonstrates the flexibility of our technique using a number of typical application examples.

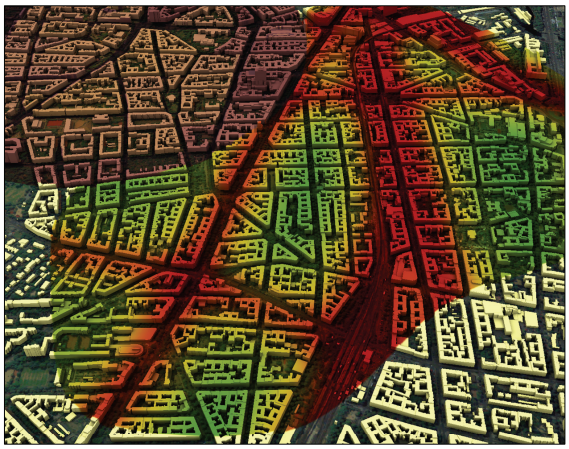

Figure 1. Exemplary hierarchical and dynamic mapping of a traffic-frequency and building category dataset onto a 3D virtual city model.

1. Introduction

Today, the availability and acceptance of interactive 3D geovirtual environments, especially of 3D virtual city and landscapes models, as a tool for information visualization have increased. The possible (semi)automatic derivation of virtual representations of the real world with a certain degree of up-to-dateness, at a sufficient quality and at reasonable costs, provides the basis of applications beyond marketing purposes: as tools for scientific visualization of geo-referenced information. This includes 3D GeoVE as scenery for visualizing geo-referenced thematic data.

High-order visualizations [2], e.g., coloring according to specific statistical data or cue-based focus+context visualization [4], typically rely on assigning information encoded in raster data [15] to the geometry of a 3D GeoVE. This assignment can be implemented using the concept of *texturing* [1], which requires a *texture coordinate mapping* to associate geometry and texture data. This data is acquired from different data sources, e.g., computer aided design (CAD), geoinformation systems (GIS), and building information modeling (BIM) models. For these models, the texture mapping is *static*, i.e., the necessary texture coordinates are computed in a pre-processing step. Consequently, the design and appearance are inherent properties of these models and difficult to change and edit at run-time. Thus, static mapping is only suitable for the visualization of non-dynamic data, such as land-use or long-term statistical information.

To enable the application of 3D GeoVE for decision support systems, e.g., to facilitate the fighting of floods or forest fires, or for the visualization of dynamic simulation results, a *dynamic texture mapping process* is required. For example, modern approaches [10] use the concept of general-purpose computation on graphics processing units (GPGPU) to compute physical simulations of gases or fluids in real-time. These techniques use textures as main data structure.

Projective texture mapping [12] can perform *dynamic texture coordinate generation*, but existing implementations

[6] are limited with respect to the number of mappings that can be applied in a single rendering pass. This limitation can be compensated for by using multiple rendering passes, but this decreases the performance especially for models with high geometrical complexity. Further, the available concepts of texture mapping do not meet the requirements, in terms of flexibility and scalability, necessary for modern information visualization.

Based on the assumption that the 3D geometry of a virtual city or landscape model can be partially approximated by a 3D reference plane, our approach performs the dynamic mapping process by projecting the raster-data into the virtual scene. In combination with a framework for color transfer functions, our approach can be used to render most surface-related phenomena (Figure 1). To summarize, this paper presents the following contributions to the reader:

1. It presents an extendable concept of mapping raster-based input data onto 3D GeoVE based on projective texture mapping and user-specified color transfer functions.

2. It provides a prototypical, fully hardware accelerated implementation that enables the rendering of a high number of projective mappings within a single rendering pass.

3. It demonstrates the capabilities of our framework by different application examples and provides the respective performance evaluations.

The remainder of this paper is organized as follows. Section 2 reviews related work concerning our topic. Section 3 introduces the basic concept of our approach. Section 4 gives implementation details. Section 5 presents our results using application examples. Section 6 discusses the performance and limitations of our approach and gives ideas for future work. Section 7 concludes the paper.

2. Related work

This section reviews related work on *dynamic texture coordinate generation* (DTCG) and style transfer functions (STF). The concept of *projective texture mapping* (PTM) [12] is fundamental for a number of advanced rendering techniques, such as per-pixel spotlight rendering or shadow mapping. It is supported by a wide range of rendering application programming interfaces (API) and hardware. The DTCG is performed by applying a projection matrix to each vertex or fragment of a 3D scene. In [5], it is used for *view-dependent texture mapping* (VDTM) of oblique aerial images onto 3D geometry. Due to the design of hardware APIs, existing implementations of PTM [6] are limited with respect to the number of projective textures, i.e., the number of simultaneously available texture units (varying between 8 and 32). Further, PTM suffers from back-projections and artifacts caused by mismatches in sampling frequencies when perspective projection matrices are used. In existing geovisualization frameworks [8], PTM is used to apply rasterized polygonal data, e.g., street networks or land-coverage information onto digital terrain models. This approach avoids z-fighting artifacts between the digital terrain model and the polygonal data that are caused by the different tessellation levels of the respective geometry.

Texture bombing [7] is another approach for combining textures at a random basis. This rendering technique places small images at irregular intervals on larger images and is applied within a single rendering pass. During the texture mapping process, the texture-coordinate space of an input primitive is divided into cells. Then, detail images are randomly placed within the selected cells which results in a collage-like output texture. The process of cell-determination is costly, since it requires a high number of texture sampling operations to apply detail images across cell borders without artifacts.

Textures are able to store data that represent different information. During the process of information visualization, color or *style transfer functions* (STF) are often used to map such data to specific color spaces [13] or to highlight salient features of a virtual scene. In [3], style transfer functions for volume rendering are presented. The approach uses texture data for representing transfer functions and shader programs for their evaluation at runtime.

3. Concept of projective mappings

Together with a concept of color transfer functions, specific to 3D GeoVE, this paper presents a combination of projective texture mapping and texture bombing to overcome the limitations of both approaches. Our concept is based on the assumption that the geometry of a geovirtual model can be approximated by a reference plane $R = (O_R, N_R)$, represented by the origin $O_R \in \mathbb{W} = \mathbb{R}^4$ in homogeneous world-coordinates and the plane normal vector $N_R \in \mathbb{D} = [0;1]^4 \subset \mathbb{R}^4$. Given this, our concept is designed of three main components:

1. **Data and Color Layers:** The data layers represent the raster-data that is mapped onto the geometry of a 3D GeoVE. The set of all available 2D raster-data layers L_i is denoted as $\mathcal{L} = (L_0, \ldots, L_m)$. The content of these layers can further be mapped to final colors using 1D color look-up tables $\mathcal{C} = (C_0, \ldots, C_k)$.

2. **Texture Coordinate Generation:** This component computes the texture coordinates for each scene point using projection matrices that are derived from a parameterization described in Section 3.2. The step as-

sumes that an application already performed the georeferencing for each parametrization, and thus, provides world-space coordinates for rendering.

3. **Color Transfer Functions:** Given the texture coordinates and the sampled layer values, this component enables coloring, masking, blending, and the composition into an output value for each mapping using instances of the transfer function described in Section 3.3.

3.1. Projective texture mapping

The sampling coordinates $S \in \mathbb{D}$ for fetching the layer data for an input point $P \in \mathbb{W}$ are computed by $S = \mathbf{T} \cdot P$. The homogeneous projection matrix $\mathbf{T}$ is defined as follows:

$$\mathbf{T} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{M}_T \cdot \mathbf{M}_P \cdot \mathbf{M}_O \cdot \mathbf{M}_V^{-1}$$

$\mathbf{M}_V^{-1}$ represents the inverse viewing transformation that maps a point from eye-space into world space coordinates. The projection mapping is computed using the *projector orientation matrix* $\mathbf{M}_O$ and the orthographic *projector matrix* $\mathbf{M}_P$, which direction-of-projection (DOP) is usually N_R. $\mathbf{M}_T$ represents possible scaling, rotation, and translation operations applied before the coordinates are mapped into the texture space $\mathbb{D}$.

3.2. Mapping parameterization

To enable a compact parameterization of PTM, we propose the structure of a *projective mapping* (PM) as follows:

$$PM = (O, U, V, \mathbf{M}_T, L_i, C_j, CM)$$

The parameter $O \in \mathbb{W}$ denotes the center of the rectangular mapping. To support an anisotropic orientation, the vectors $U, V \in \mathbb{W}$ define the bounds on the reference plane and form an orthonormal base used for the projector orientation $\mathbf{M}_O$. Associated with each mapping is a data layer $L_i \in \mathcal{L}$, a color look-up table $C_j \in \mathcal{C}$, and an additional transformation matrix $\mathbf{M}_T$. CM represents the parameter set for the color transfer function described in Section 3.3.

We denote a set of n projective mappings as $\mathcal{PM} = \{PM_i | i = 0, \ldots, n-1\}$. The explicit order is necessary for the correct compositing of the transfer-function results during the evaluation process of each PM (Section 4.2). Given the above parameterization, instances of $\mathbf{M}_O$ and $\mathbf{M}_P$ are derived as follows:

$$\mathbf{M}_O = \begin{bmatrix} J_x & J_y & J_z & -O_x \\ K_x & K_y & K_z & -O_y \\ L_x & L_y & L_z & -O_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{ll} J & = |U| \\ K & = |V| \\ L & = J \times K \end{array}$$

$$\mathbf{M}_P = \begin{bmatrix} h & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{lll} h & = & \frac{\|U\|}{2} \\ v & = & \frac{\|V\|}{2} \end{array}$$

3.3. Color transfer functions

This section focuses on mapping the scalars sampled from L_i into an output color value. To enable a general approach, the user needs to define the following functions for a color mapping CM:

$$CM = (f_{BS}, f_{BD}, f_{CS}, f_{CD}, \mathbf{M}_C, f_\alpha)$$

For a projective mapping PM, the source color $C_S = f_{CS}(S)$ can be fetched from standard static texture channels (diffuse color, light maps, etc.), from the associated data layer L_i, or derived from the C_j using the sampled value of L_i. The results can then be adjusted using a color matrix $\mathbf{M}_C$, which does not affect the alpha channel. Thus, $f_\alpha(S)$ is used to adapt the respective alpha channel. The opacity scalar values can be fetched from a mask layer, derived from a gray-scale color ramp, or simply be uniform. The output color C_O for a PM is computed via:

$$C_O = f_B(C_{BS}, C_S, C_{BD}, C_D)$$

f_B is the standard color blending operation [1], based on the blend functions for source color $C_{BS} = f_{BS}(C_S, C_D)$ and destination color $C_{BD} = f_{BD}(C_S, C_D)$. Thus, standard combination modes, such as color replacement, multiplication, or blending, are supported (Section 4.2).

4. Implementation

Our prototypical implementation uses OpenGL [11] with the support of shader programs [9]. The implementation of our concept is capable of rendering a high number of projective mappings within a single rendering pass. This is useful to achieve a high rendering performance for large-scale 3D GeoVE. Therefore, it is necessary to encode the data layers $\mathcal{L}$, color look-up tables $\mathcal{C}$, and the parameterization of $\mathcal{PM}$ into suitable GPU data structures.

4.1. GPU data-structures

To enable an efficient and scalable implementation, we use *2D texture arrays* to represent $\mathcal{L}$ and $\mathcal{C}$. This enables the usage of more textures than the number of simultaneously accessible texture units and texture coordinates slots. A 2D texture array is similar to a 3D texture without the bi-linear interpolation between and with direct access to the z-slices. It further facilitates sharing of the layer data between different projective mappings.

To enable scalability of the implementation with respect to the number of simultaneously active PMs, we use *texture buffers* to store the settings of $\mathcal{PM}$. Our implementation transfers the scalar and vector parameters of each active PM (see Section 4.3) into a matrix form. The projection matrices $\mathbf{T}_i$ are constant for each rendering frame and, therefore, are pre-computed on the central processing unit (CPU) as described in Section 3.2. Together with the color matrices $\mathbf{M}_{C_i}$, they are stored successively within a single texture buffer that is accessed via indexing during shader execution. The transfer step is only performed if the configuration is changed at runtime.

4.2. Per-fragment mapping evaluation

```
1  evaluate(P, PM, L, C) {
2  C_O ← sceneColor()
3  i ← ||PM||
4  while(i > 0) {
5   PM ← PM_i ∈ PM
6   T ← T_i ∈ PM
7   S = T · P
8   if(testAABB2D(S)) {
9     L  ← L_i ∈ L
10    C  ← C_i ∈ C
11    M_C ← M_Ci ∈ CM_i
12    C_S = f_at(S, f_CS(S, L, C))
13    C_D ← f_CS(C_O, L, C)
14    C_BS = f_BS(C_S, C_D)
15    C_BD = f_BD(C_S, C_D)
16    C_O = f_B(C_BS, C_S, C_BD, C_D)
17  } i = i - 1
18 }setFragmentColor(C_O) }
```

Figure 2. Evaluation of projective mappings.

The evaluation of the projective mappings is performed in a fragment shader program that is activated and configured before rendering the complete scene geometry. The pseudo-code in Figure 2 shows the evaluation process for all active $PM_i \in \mathcal{PM}$. For each PM, the parameters are fetched from the texture buffer (Line 5-6, 9-11), then the texture coordinates for the data layer are computed (Line 7), the color mapping and blending-mode functions are applied (Line 12-15), and finally the color blending is performed (Line 16). Given the explicit order over $\mathcal{PM}$, the evaluation iterates over all mapping starting with the last (Line 3-17). At the end of the evaluation, the fragment output color is set accordingly (Line 18).

4.3. Rendering optimizations

Per-fragment looping and sampling are costly operations, especially if performed for a high number of projective mappings. To increase rendering speed, we apply two CPU based culling techniques to each mapping PM, before it is encoded into the GPU data structure. First, we use view-frustum culling to disable the mapping, if its respective bounding area does not intersect the view frustum of the virtual camera. Following to this test, we apply a thresholding based on a simple screen-space error metric. The object-space area of a PM is mapped to a screen space error as follows:

$$\rho = \frac{w}{\phi} \cdot \frac{\|U\| \cdot \|V\|}{\|O - COP\|}$$

The vector COP represents the virtual cameras center-of-projection. The variable $w \in \mathbb{N}_+$ denotes the horizontal screen resolution, while $\phi \in [0; 360[\subset \mathbb{R}$ is the current horizontal field-of-view. If the resulting value $\rho_i \in \mathbb{R}$ is smaller than a user defined threshold $t \in \mathbb{R}$, the PM will be disabled. For a scene within a normalized volume ($[-1; 1]^3$), $t \approx 0.02$ is appropriate to balance the trade-off between rendering performance and possible popping artifacts.

During the evaluation of the mapping, we further test (`testAABB2D` in Figure 2, Line 8) if the computed sampling coordinates S are within the valid range $\mathbb{D}$. If this test fails, the sampling of the data layer L_i is avoided.

5. Results

Our implementation enables a hierarchy-based combination of different mappings (Figure 1) and can be applied to render different standard visualizations (Figure 3). In this section, we briefly discuss four application examples and their respective parameterization.

Raster-Data Visualization: The main area of application is the mapping of dynamic spatial-temporal data, e.g., crime data, noise data and environmental pollution data (Figure 3.A), transmitter coverage, or fire-, explosion-, and flood zones. It can further be used to visualize building heights, iso-lines, and the positions of mobile objects. By changing the color transfer functions of CM accordingly, it is possible to the communicate the overlapping of scalar values (Figure 3.F).

Object Highlighting: A variant of the value mapping is used to highlight specific features or landmarks of a virtual city or landscape model. This can be performed for a number of points-of-interests or for routes to facilitate navigation and orientation within 3D GeoVE. Figure 3.B shows a visualization that applies different PMs to highlight a route and its immediate surrounding by reducing brightness and saturation of the contextual scene using the color matrix M_C.

Given a route described by a number of way points $\mathcal{W} = \{W_0, \ldots, W_n\}$, a point radius $r \in \mathbb{R}_+$, and an edge height $h \in \mathbb{R}_+$, we derive n projective mappings $PM_i, i = 1, \ldots, n$ for the way points with the following setting: $O_i = W_i, U_i = A \cdot s$, and $V_i = B \cdot s$. Here, $A, B \in \mathbb{D}$ are orthogonal normal vectors in the reference plane R. Further, we derive $n-1$ mappings $PM_j, j = n, \ldots, 2n-1$ for each edge between the way points $W_S = W_{j-n}$ and $W_E = W_{j-1-n}$. The respective setting is computed as follows: $O_j = (W_S + W_E)/2, A_j = \|W_E - W_S\| \cdot h$, and $B_j = N_P \times A_j \cdot |W_E - W_S|$.

Annotations: Since our approach enables the rendering of a high number of projective mappings, it can be used to annotate 3D GeoVE with cartographic symbols (Figure 3.E)

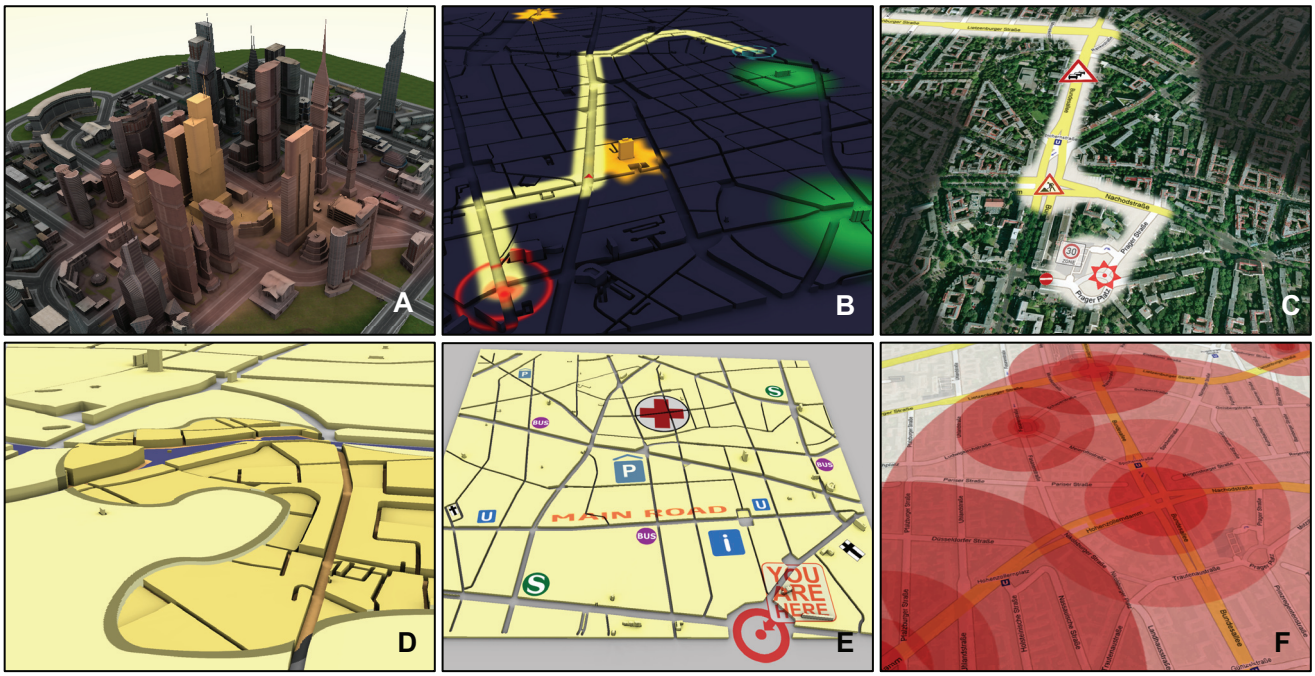

Figure 3. Exemplary visualization created using the presented approach. A: Fictive data projected onto a virtual city model; B: Object-highlighting of a route and landmarks; C: Selective compositing of different terrain textures using masks; D: Combination of different geometries using multi-pass rendering; E: Annotation of generalized virtual 3D city model with multiple symbols; F: Rendering of overlapping, non-uniform transmitter ranges.

to facilitate the creation of 3D digital maps. The advantages of projecting symbols onto the city model geometry are the emphasis of the symbol-object relationship, the reduction of interpretation ambiguities, and the removal of label-scene occlusions. This delivers good visual results for generalized versions of virtual 3D city models [14], viewed from a bird's-eye or a plan perspective. Due to our dynamic concept, the icons can be exchanged, scaled, rotated, or faded view-dependently to ensure an optimal readability of such a 3D digital map. It further enables view-dependent switching between different levels-of-abstraction (LOA) without adapting the projective mappings.

Focus+Context Visualization: Another application of PM is lens-based or cue-based focus+context visualization. Projective mappings can be used for masking and blending different texture layers (Figure 3.C). The particular data layer masks can be derived from thematic data or drawn directly by the user onto the digital terrain. In such use-cases, our approach has a number of advantages. It enables smooth transitions between data layers defined by 2D masks, which can have arbitrary shapes. The DTCG enables the interactive movement of a lens, while the implicit hierarchy supports the usage of overlapping lenses.

Our approach can further be used to combine different city model geometries (Figure 3.D) as a simplified version of the approach presented in [14]. Using multiple rendering passes, this can be achieved by projecting binary masks and enable alpha testing to discard the respective fragments.

6. Discussion & future work

6.1. Performance evaluation

Our test platform is an NVIDIA GeForce 8800 GTS with 640 MB video memory and Athlon 64 X2 Dual Core 4200+ with 2.21 GHz and 2 GB of main memory at a viewport resolution of 1600×1200 pixels. The test application does not utilize the second CPU core. Table 1 provides the performance measurements of the examples that can be rendered within a single rendering pass. The performance evaluation showed that our implementation is fill-limited and depends mainly on the geometrical complexity of the model and the number of PM used. This means, the performance decreases, while the number of fragments affected by a projective mapping increases. This is usually the case if the user is near the reference plane or a high number of projective mappings inside the current view frustum have to be evaluated. Here, the implementation of the color transfer

Table 1. Performance evaluation for the scenes depicted in this paper (in Frames-Per-Second).

Figure	#Vertex	#PM	$\mathbf{PM}_{off}$	$\mathbf{PM}_{on}$
1	1,040,503	3	10.41	9.87
3.A	41,032	4	52.88	51.20
3.B	61,756	27	156.10	27.94
3.C	6	4	1063.83	236.97
3.E	37,404	16	206.21	45.70
3.F	6	6	1075.27	110.54

functions is the main computational bottleneck.

6.2. Problems & limitations

In principle, our concept suffers from the same problems and limitation of texture mapping itself. The quality of the rendering output depends on the texture resolution of the data layer, the sampling strategies, and the numerical precision of projective texture mapping. Since we are using orthographic projector matrices with a DOP along N_R, our implementation does not suffer from artifacts caused by a mismatch in sampling frequencies. Further, the current implementation requires a uniform texture resolution and color channel configuration and does not support the usage of texture atlases. However, the main limitation represents the restriction of our concept to 3D GeoVE that can be approximated by a planar surface.

6.3. Challenges for future work

For future work, we are heading toward an implementation that supports out-of-core rendering for the data layers. The concept needs also to be extended to support texture atlases and 3D texture for data as well as color layers. Further, the concept and implementation can be extended to provide a level-of-detail mechanism for the data layers. Furthermore, the usage of adaptive, view-dependent projector matrices and the integration of procedural textures indicate promising features.

7. Conclusions

This paper presents an extendable concept for dynamically mapping 2D raster-data to 3D geovirtual environments. It is based on projective texture mapping for dynamic texture coordinate generation and provides a fully hardware accelerated implementation that enables the interactive rendering for a high number of projective mappings. This approach enables the implementation of dynamic visualization of arbitrary thematic data using 3D geovirtual environments as scenery for an effective information communication.

Acknowledgments

This work has been funded by the German Federal Ministry of Education and Research (BMBF) as part of the InnoProfile research group "3D Geoinformation"' (www.3dgi.de). We are thankful to Tassilo Glander and Haik Lorenz for providing test data sets.

References

[1] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 3rd edition edition, 2008.

[2] S. Björk, L. E. Holmquist, and J. Redström. A Framework for Focus+Context Visualization. In *Proc. of INFOVIS '99*, page 53, 1999.

[3] S. Bruckner and M. E. Gröller. Style Transfer Functions for Illustrative Volume Rendering. *Computer Graphics Forum*, 26(3):715–724, Sept. 2007.

[4] A. Cockburn, A. Karlson, and B. B. Bederson. A Review of Focus and Context Interfaces. Technical report, 2006.

[5] P. Debevec, Y. Yu, and G. Boshokov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. Technical report, Berkeley, CA, USA, 1998.

[6] C. Everitt. Projective Texture Mapping. Technical report, NVIDIA Coorporation, April 2001.

[7] R. S. Glanville. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, chapter Texture Bombing. Addison-Wesley, 2004.

[8] O. Kersting and J. Döllner. Interactive Visualization of 3D Vector Data in GIS. In *Proc. of the ACM GIS*, pages 107–112, 2002.

[9] J. Kessenich. *The OpenGL Shading Language Language Version: 1.20 Document Revision: 8*, September 2006.

[10] Y. Liu, X. Liu, and E. Wu. Real-Time 3D Fluid Simulation on GPU with Complex Obstacles. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 247–256, Washington, DC, USA, 2004. IEEE Computer Society.

[11] NVIDIA. *NVIDIA OpenGL Extension Specifications for the GeForce 8 Series Architecture (G8x)*, November 2006.

[12] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast Shadows and Lighting Effects Using Texture Mapping. *SIGGRAPH*, 26(2):249–252, 1992.

[13] C. Tominski, G. Fuchs, and H. Schumann. Task-Driven Color Coding. In *IV*, pages 373–380, 2008.

[14] M. Trapp, T. Glander, H. Buchholz, and J. Döllner. 3D Generalization Lenses for Interactive Focus + Context Visualization of Virtual City Models. In *Proc. of IEEE IV*, pages 356–361, July 2008.

[15] J. Wood, S. Kirschenbauer, J. Döllner, A. Lopes, and L. Bodum. *Exploring Geovisualization*, chapter Using 3D in Geovisualization, pages 295–312. Elsevier, 2005. ISBN 0-08-044531-4.