Service-Based Decomposition of 3D Geovisualization Systems

Jürgen Döllner and Dieter Hildebrandt

Abstract—In this contribution, we outline and discuss a service-oriented architecture for systems that visualize geovirtual 3D models (e.g., virtual 3D city models and landscape models) on lightweight clients. The approach is motivated by the growing demand for scalable, client-server based geovisualization systems and increasingly complex, massive 3D models. For this purpose, we have developed a 3D server responsible for 3D rendering of G-buffer cube maps, and a corresponding light-weight 3D client that visualizes these G-buffer cube maps, controls their streaming, and handles 3D interaction. The approach provides a solution to the bandwidth-limited streaming of complex, massive geovirtual 3D models because it decouples streaming data complexity from model complexity. We briefly discuss a collection of service building blocks for stylizing and rendering effects, which can be composed within a data flow graph to implement application-specific 3D geovisualization techniques. The approach facilitates development, deployment, and scaling of 3D geovisualization systems within service-oriented infrastructures.

Index Terms—Geovisualization, virtual worlds, virtual environments, virtual 3D city models, service-oriented architecture.

1 Introduction

Geovisualization forms part of a growing number of mobile and web-based applications. For their service-oriented implementation and deployment, scalable and high-performance 3D rendering services are needed, in particular, if these services operate within a cloud computing environment. For 2D map visualization, web map services [4] have been successfully evolved, whereas web view services [7,8] for 3D geovirtual worlds have not reached the same level of maturity. 3D web view services, however, are crucial elements for the incorporation of 3D geovisualization into complex web-based workflows and services.

While common geovisualization software architectures assume a client that is responsible for real-time 3D rendering ("thick client") and receiving streamed model data, this approach is faced with a number of limitations in the case of *massive geovirtual 3D worlds* such as virtual 3D city models and landscape models due to limited bandwidth and limited client 3D rendering capabilities. Consequently, service-based and service-oriented system architectures that cope with these limitations are essential.

In our approach, we model the 3D geovisualization pipeline [6] by a service-oriented architecture. Its key elements include:

- Server-side, off-screen 3D rendering service;
- Generation of multi-layered, six-sided panorama images, so called *G-buffer cube maps*, based on the G-buffers [10];
- Post-processing services operating on G-buffer cube maps that provide advanced rendering and stylization techniques;
- Streaming of G-buffer cube maps to thin 3D clients, which reconstruct approximated 3D views.

The G-buffer cube maps allow us (1) to achieve high performance 3D viewing on the thin client due to decoupling streaming data complexity from model data complexity and (2) to model advanced rendering techniques by a data flow graph whose nodes are G-buffer cube maps or general image operators. This way, a large number of rendering and stylization effects, e.g. cartographic stylization or illustrative rendering, can be transparently modeled and efficiently implemented in a service-oriented way.

2 G-Buffer Cube Maps

The server-side, off-screen 3D rendering generates G-buffer cube maps (Fig. 1). A G-buffer cube map is a virtual six-sided texture set that captures the view from the cube's center in all six directions. G-buffer cube maps represent an omnidirectional panoramic view of the scene not only in a discretized but also in a homogeneous way, abstracting from the original geometry (e.g., polygon, freeform surface, point cloud, etc.) of a visible object. The G-buffer can contain several layers, including:

- RGB layer: A color texture that captures the visual appearance of the geovirtual 3D model.
- Depth layer: A one-channel float texture that captures the corresponding depth buffer for a given view.
- Normal layer: A vector-based texture that captures the surface normals.
- World coordinates layer: For each pixel, it contains the world coordinates of the visualized object fragment.
- Object-ID layer: A one-channel integer texture that captures an object identifier.
- Object-Type layer: A one-channel integer texture that captures the object class identifier.

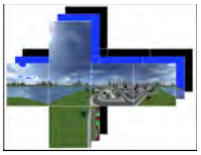


Fig. 1. Concept of G-buffer cube maps.

The G-buffer cube maps play a central role to decouple model complexity from streaming data complexity: The G-buffer cube maps have a constant size, which depends on the resolution requested by the 3D client, but does not depend on the number and size of geometry primitives of the geovirtual 3D models and on the number and size of textures used to define their visual appearance. The order of magnitude for, e.g., a typical virtual 3D city model commonly ranges between 10^7 and 10^9 geometries (or primitives), e.g., including a high-resolution DTM (e.g., 0.1m) and textured LOD1/LOD2/LOD3 building and site models. The complexity of the computer graphics model (e.g., scene graph and LOD data structures) typically increases that complexity by one order of magnitude leading to 10^8 to 10^{10} due to LOD and spatial data structures. On the contrary, the complexity of the G-buffer cube map is determined by the six sides of the cube map, number and bit depth of layers per side (typically 2-5) and the image resolution per side (e.g., 1,024x1,024 pixel) leading to a complexity of 10^7 - 10^8 pixel. Our approach uses a thin-client approach (Fig. 2) keeping the model on the server side, which is essential, for example, in the case of massive LOD-3 city models like the Roman Cologne model (Fig. 3) with more than 500M triangles. Dedicated hardware can be used for the server that permits in-memory city model representation and uses n-core CPUs.

- *Jürgen Döllner, Hasso-Plattner-Institut, University of Potsdam, Germany, E-Mail: doellner@hpi.uni-potsdam.de.*
- *Dieter Hildebrandt, Hasso-Plattner-Institut, University of Potsdam, Germany, E-Mail: dieter.hildebrandt@hpi.uni-potsdam.de.*

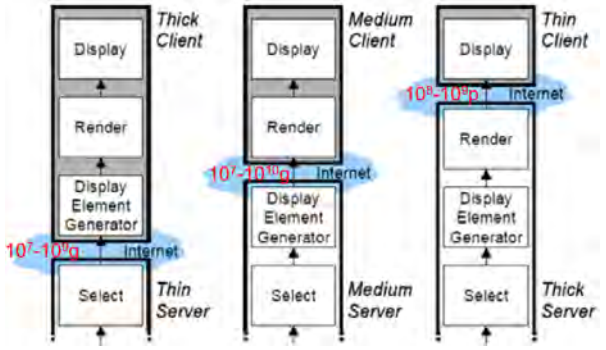


Fig. 2. Thick, medium, and thin 3D client architectures with orders of magnitude of geometries (g) and image data (p).

Fig. 3. The Roman Cologne model, a geographically small but geometrically/graphical massive virtual 3D city model.

3 Using G-Buffer Cube Maps

The G-buffer cube maps can be used similar to panoramic images to reconstruct an approximated view of a geovirtual 3D model. The client renders the six-sided textured cube from a current camera configuration (Fig. 4).

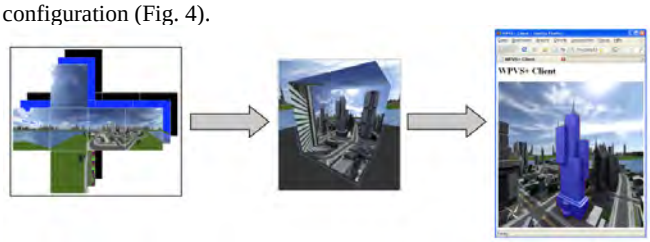


Fig. 4. Scene reconstruction based on G-buffer cube maps.

If the camera changes, the client re-calculates the specification of the next G-buffer cube map and sends the request to the server. The client can support fundamental navigation operations such as zooming, panning, and 3D camera movements. Until it receives an up-to-date G-buffer cube map, it can operate on the current version, potentially introducing artifacts commonly visible as blurred image areas due to insufficient resolution.

The client can also implement object-sensitive operations such as object selection or interactive spatial measurements if it takes into account the various information layers (e.g., object-id layer or 3D world coordinate layer).

The implementation of a typically 3D client application requires moderate 3D graphics capabilities such as provided by OpenGL ES. However, it does not require advanced shader programming because it is not responsible for generating the source image data, i.e., it can rely on G-buffer contents as provided by the 3D server. The separation between image synthesis on server side (a controlled hardware/software environment) and image use on client side (a potentially diversified, changing environment) simplifies software development drastically. In addition, clients can avoid energy-critical operations such as shader-intensive multi-pass rendering.

4 Service-Based Advanced Rendering Techniques

The service-based approach based on G-buffer cube maps can be extended on the server-side by a data flow graph (e.g., [1,3]), whose nodes represent G-buffer or general image operations [5] related to, e.g., color, depth, normal, and object-id information and whose directed edges represent the data flow. This way, advanced rendering and stylization effects can be efficiently implemented and transparently composed.

We have identified a first collection of general-purpose, reusable subservices that enable the composition of web view services for geovirtual 3D models operating on G-buffer cube maps. The collection includes:

- 3D image synthesis that generates the base G-buffer;
- Shadow mapping synthesis that generates shadow maps [2];
- Ambient occlusion synthesis that generates approximated ambient light intensities [2];
- Nonphotorealistic image processing that emphasizes edges [9];
- Depth-of-field effect to infiltrate an artificial object-based focus area in the view [2];
- Projective texture effect to superimposed projective textures on the given view;
- Highlighting effect that emphasizes the silhouette of selected objects;
- Generic G-buffer and image converter, blending, and convolution operators.

An example of a complex geovisualization pipeline based on these subservices is given in Fig. 5, which shows the related data flow graph. This service generates perspective stylized 3D views on virtual 3D city models, including advanced rendering effects such as global illumination, shadows, and non-photorealistic rendering. The RGB layer as the most important appearance capturing input and the RGB layer of the final result are depicted in Fig. 6.

5 Conclusions

The server-side, service-oriented rendering of geovirtual 3D models based on G-buffer cube maps enables lightweight client applications to allow users to access high quality, image-based visual representations of these models regardless of their complexity because geodata remains on the server. Since the resolution of the G-buffer cube maps is fixed, only a set of multi-layer images needs to be transferred. The lightweight client requires only a small amount of 3D rendering features needed to model and interactively render the panorama-like visual representation; it can additionally implement various object-based interaction techniques since object-based information and general geometry information is encoded in the G-buffer cube maps.

The approach furthermore provides a framework for developing G-buffer based advanced rendering and stylization effects, decoupled from model complexity due to the discretized G-buffer. These effects can be independently developed, maintained, deployed, and scaled; they also represent a collection of building blocks to compose complex geovisualization systems based on a data flow graph.

References

[1] Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA. Adobe Pixel Bender Developer's Guide, 2010.

[2] Tomas Akenine-Möller, Eric Haines, and Natty Hoffman. Real-Time Rendering. A.K. Peters, Ltd., Natick, MA, USA, 3rd Ed., 2008.

[3] Apple Inc., 1 Infinite Loop, Cupertino, CA 95014. Apple Quartz Composer User Guide, July 2007.

[4] Jeff de la Beaujardiere, editor. OpenGIS Web Map Server Implementation Specification, Version 1.3.0. Open Geospatial Consortium Inc., March 2006.

[5] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing. Prentice Hall International, Upper Saddle River, N.J., 3rd edition, 2008.

[6] R.B. Haber and D. A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In B. Shriver, G. M. Nielson, and L. Rosenblum, editors, Visualization in Scientific Computing, pages 74–93, Los Alamitos, 1990. IEEE Computer Society Press.

[7] Benjamin Hagedorn, Dieter Hildebrandt, and Jürgen Döllner. Towards Advanced and Interactive Web Perspective View Services. In Developments in 3D Geo-Information Sciences, Lecture Notes in Geoinformation and Cartography, Berlin, Heidelberg, November 2009. Springer.

[8] Benjamin Hagedorn, Dieter Hildebrandt, and Jürgen Döllner. Web View Service Discussion Paper, Version 0.6.0. Open Geospatial Consortium Inc., February 2010.

[9] Jan Eric Kyprianidis and Jürgen Döllner. Image Abstraction by Structure Adaptive Filtering. In Ik Soo Lim and Wen Tang, editors, EG UK Theory and Practice of Computer Graphics, pages 51–58. Eurographics Association, 2008.

[10] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. SIGGRAPH Computer Graphics, 24(4):197–206, 1990.

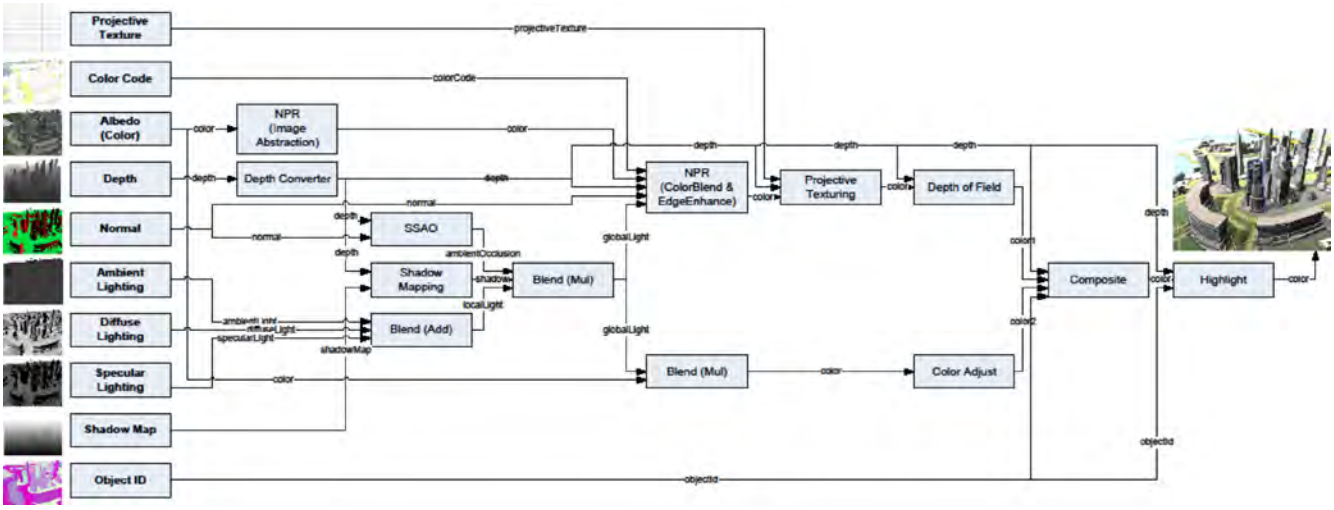
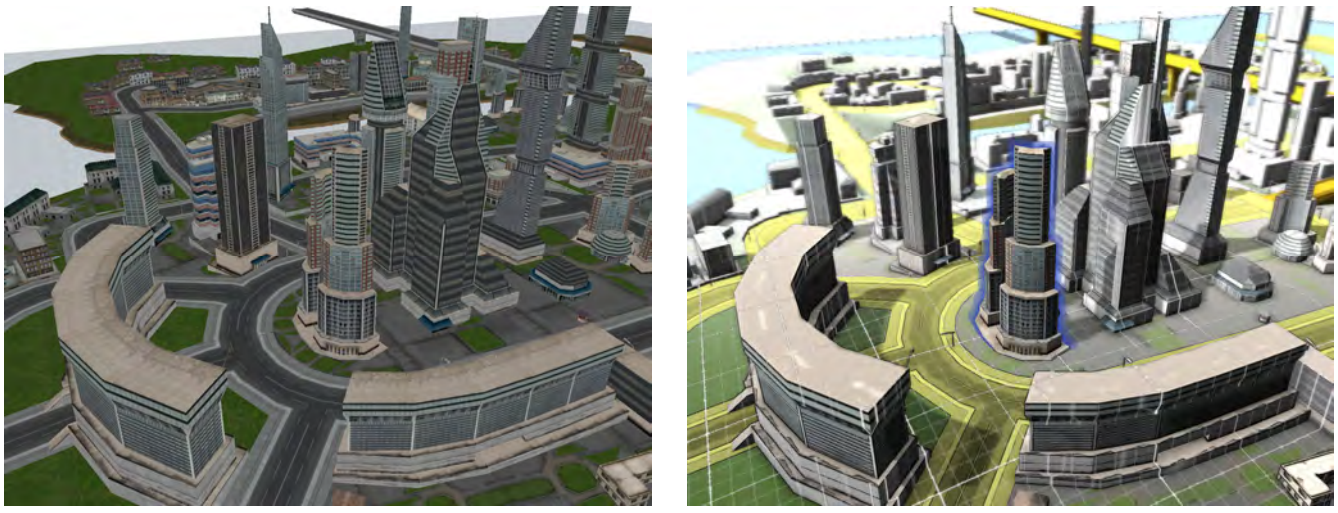


Fig. 5. Data flow graph for a stylized web 3D view services for virtual 3D city models.



(a) (b)

Fig. 6. Exemplary input and output of the data flow graph depicted in Fig. 5: (a) The RGB layer as the most important appearance capturing input and (b) the RGB layer of the final result.