

# Interactive Rendering Techniques for Focus+Context Visualization of 3D Geovirtual Environments

Dissertation  
zur Erlangung des akademischen Grades  
"doctor rerum naturalium"  
(Dr. rer. nat.)  
in der Wissenschaftsdisziplin Informatik

eingereicht an der  
Mathematisch-Naturwissenschaftlichen Fakultät  
der Universität Potsdam

**Dipl.-Inform. Matthias Trapp**

Potsdam  
23. Januar 2013

This work is licensed under a Creative Commons License:  
Attribution – Noncommercial – NoDerivs 3.0 Germany  
To view a copy of this license visit  
<http://creativecommons.org/licenses/by-nc-nd/3.0/de/>

Published online at the  
Institutional Repository of the University of Potsdam:  
URL <http://opus.kobv.de/ubp/volltexte/2013/6682/>  
URN <urn:nbn:de:kobv:517-opus-66824>  
<http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-66824>

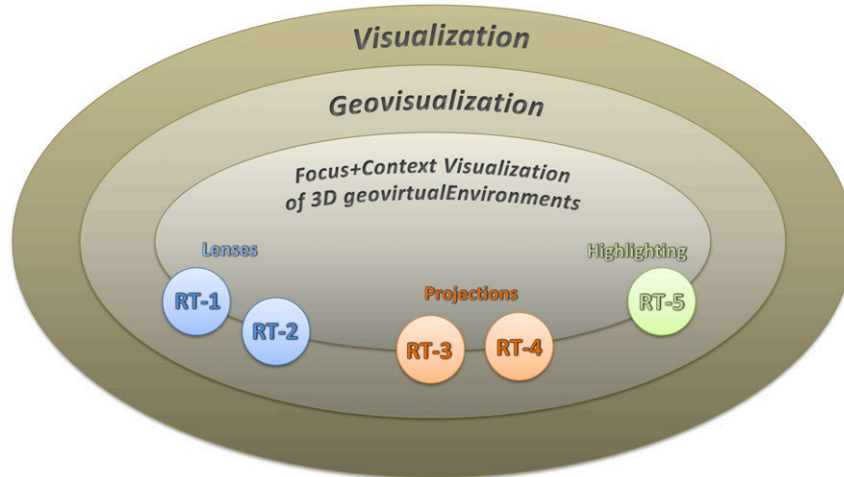


*To my family.*



## Abstract

This thesis introduces a collection of new real-time rendering techniques and applications for focus+context visualization of interactive 3D geovirtual environments such as virtual 3D city and landscape models. These environments are generally characterized by a large number of objects and are of high complexity with respect to geometry and textures. For these reasons, their interactive 3D rendering represents a major challenge. Their 3D depiction implies a number of weaknesses such as occlusions, cluttered image contents, and partial screen-space usage.



To overcome these limitations and, thus, to facilitate the effective communication of geo-information, principles of focus+context visualization can be used for the design of real-time 3D rendering techniques for 3D geovirtual environments (see Figure). In general, detailed views of a 3D geovirtual environment are combined seamlessly with abstracted views of the context within a single image. To perform the real-time image synthesis required for interactive visualization, dedicated parallel processors (GPUs) for rasterization of computer graphics primitives are used. For this purpose, the design and implementation of appropriate data structures and rendering pipelines are necessary. The contribution of this work comprises the following five real-time rendering methods:

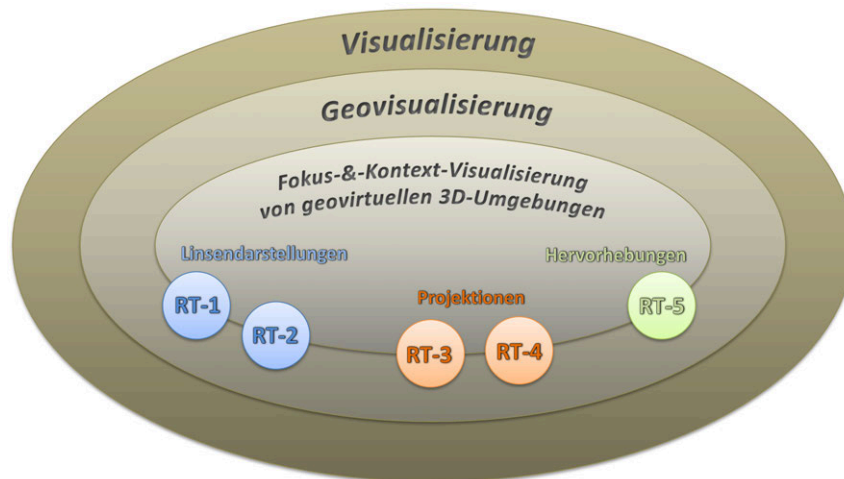
- RT-1** The rendering technique for *3D generalization lenses* enables the combination of different 3D city geometries (e.g., generalized versions of a 3D city model) in a single image in real time. The method is based on a generalized and fragment-precise clipping approach, which uses a compressible, raster-based data structure. It enables the combination of detailed views in the focus area with the representation of abstracted variants in the context area.
- RT-2** The rendering technique for the *interactive visualization of dynamic raster data* in 3D geovirtual environments facilitates the rendering of 2D surface lenses. It enables a flexible combination of different raster layers (e.g., aerial images or videos) using projective texturing for decoupling image and geometry data. Thus, various overlapping and nested 2D surface lenses of different contents can be visualized interactively.
- RT-3** The interactive rendering technique for *image-based deformation* of 3D geovirtual environments enables the real-time image synthesis of non-planar projections, such as cylindrical and spherical projections, as well as multi-focal 3D fisheye-lenses and the combination of planar and non-planar projections.
- RT-4** The rendering technique for *view-dependent multi-perspective views* of 3D geovirtual environments, based on the application of global deformations to the 3D scene geometry, can be used for synthesizing interactive panorama maps to combine detailed views close to the camera (focus) with abstract views in the background (context). This approach reduces occlusions, increase the usage the available screen space, and reduces the overload of image contents.

**RT-5** The object-based and image-based rendering techniques for *highlighting objects and focus areas* inside and outside the view frustum facilitate preattentive perception.

The concepts and implementations of interactive image synthesis for focus+context visualization and their selected applications enable a more effective communication of spatial information, and provide building blocks for design and development of new applications and systems in the field of 3D geovirtual environments.

## Zusammenfassung

Die Darstellung immer komplexerer raumbezogener Information durch Geovisualisierung stellt die existierenden Technologien und den Menschen ständig vor neue Herausforderungen. In dieser Arbeit werden fünf neue, echtzeitfähige Renderingverfahren und darauf basierende Anwendungen für die Fokus-&-Kontext-Visualisierung von interaktiven geovirtuellen 3D-Umgebungen – wie virtuelle 3D-Stadt- und Landschaftsmodelle – vorgestellt. Die große Menge verschiedener darzustellender raumbezogener Information in 3D-Umgebungen führt oft zu einer hohen Anzahl unterschiedlicher Objekte und somit zu einer hohen Geometrie- und Texturkomplexität. In der Folge verlieren 3D-Darstellungen durch Verdeckungen, überladene Bildinhalte und eine geringe Ausnutzung des zur Verfügung stehenden Bildraumes an Informationswert.



Um diese Beschränkungen zu kompensieren und somit die Kommunikation raumbezogener Information zu verbessern, kann das Prinzip der Fokus-&-Kontext-Visualisierung angewendet werden (siehe Abbildung). Hierbei wird die für den Nutzer wesentliche Information als detaillierte Ansicht im Fokus mit abstrahierter Kontextinformation nahtlos miteinander kombiniert. Um das für die interaktive Visualisierung notwendige Echtzeit-Rendering durchzuführen, können spezialisierte Parallelprozessoren für die Rasterisierung von computergraphischen Primitiven (GPUs) verwendet werden. Dazu ist die Konzeption und Implementierung von geeigneten Datenstrukturen und Rendering-Pipelines notwendig. Der Beitrag dieser Arbeit umfasst die folgenden fünf Renderingverfahren.

- RT-1** Das Renderingverfahren für interaktive *3D-Generalisierungslinsen*: Hierbei wird die Kombination unterschiedlicher 3D-Szenengeometrien, z. B. generalisierte Varianten eines 3D-Stadtmodells, in einem Bild ermöglicht. Das Verfahren basiert auf einem generalisierten Clipping-Ansatz, der es erlaubt, unter Verwendung einer komprimierbaren, rasterbasierten Datenstruktur beliebige Bereiche einer 3D-Szene freizustellen bzw. zu kappen. Somit lässt sich eine Kombination von detaillierten Ansichten im Fokusbereich mit der Darstellung einer abstrahierten Variante im Kontextbereich implementieren.
- RT-2** Das Renderingverfahren zur *Visualisierung von dynamischen Raster-Daten* in geovirtuellen 3D-Umgebungen zur Darstellung von 2D-Oberflächenlinsen: Die Verwendung von projektiven Texturen zur Entkoppelung von Bild- und Geometriedaten ermöglicht eine flexible Kombination verschiedener Rasterebenen (z.B. Luftbilder oder Videos). Somit können verschiedene überlappende sowie verschachtelte 2D-Oberflächenlinsen mit unterschiedlichen Dateninhalten interaktiv visualisiert werden.
- RT-3** Das Renderingverfahren zur *bildbasierten Deformation* von geovirtuellen 3D-Umgebungen: Neben der interaktiven Bildsynthese von nicht-planaren Projektionen, wie beispielsweise zylindrischen oder sphärischen Panoramen, lassen sich mit diesem Verfahren multifokale



3D-Fischaugen-Linsen erzeugen sowie planare und nicht-planare Projektionen miteinander kombinieren.

**RT-4** Das Renderingverfahren für die Generierung von *sichtabhängigen multiperspektivischen Ansichten* von geovirtuellen 3D-Umgebungen: Das Verfahren basiert auf globalen Deformationen der 3D-Szenengeometrie und kann zur Erstellung von interaktiven 3D-Panoramakarten verwendet werden, welche beispielsweise detaillierte Absichten nahe der virtuellen Kamera (Fokus) mit abstrakten Ansichten im Hintergrund (Kontext) kombinieren. Dieser Ansatz reduziert Verdeckungen, nutzt den zur Verfügung stehenden Bildraum in verbesserter Weise aus und reduziert das Überladen von Bildinhalten.

**RT-5** Objekt- und bildbasierte Renderingverfahren für die *Hervorhebung von Fokus-Objekten und Fokus-Bereichen* innerhalb und außerhalb des sichtbaren Bildausschnitts, um die präattentive Wahrnehmung eines Benutzers besser zu unterstützen.

Die in dieser Arbeit vorgestellten Konzepte, Entwürfe und Implementierungen von interaktiven Renderingverfahren zur Fokus-&-Kontext-Visualisierung sowie deren ausgewählte Anwendungen ermöglichen eine effektivere Kommunikation raumbezogener Information und repräsentieren softwaretechnische Bausteine für die Entwicklung neuer Anwendungen und Systeme im Bereich der geovirtuellen 3D-Umgebungen.

## Acknowledgments

This dissertation is the result of my research work at the Department of Computer Graphics Systems at the Hasso-Plattner-Institut (University of Potsdam). I am very grateful to my adviser Prof. Dr. Jürgen Döllner for granting me this opportunity. I would also like to thank Prof. Dr. Georg Gartner from the University of Vienna, Prof. Dr. Heidrun Schumann from the University of Rostock, and Prof. Dr. Holger Theisel from the University of Magdeburg for agreeing to review this thesis.

I thank my colleagues Haik Lorenz, Markus Jobst, and Sebastian Pasewaldt for the research collaboration on multi-perspective views and non-planar projections. Special thanks to Tassilo Glander for the long-standing collaboration on landmark scaling, 3D generalization lenses, and 3D iso-contours. I thank Amir Semmo for his outstanding support during various projects. Further, I thank Christine Lehmann, Lars Schneider, Norman Holz, and Christian Beesk for the collaborative research on object highlighting for 3D geovirtual environments. I'm thankful to Rafael Pokorski, Claus Daniel Herrmann, Michael Eichhorn, and all the others of the Colonia3D team who dedicate their time and spirit to this project and made it such a big success. Further, I am obliged to many of my colleagues at CGS: Jan-Eric Kyprianidis, Stefan Maass, Marc Nienhaus, Henrik Buchholz for fruitful discussions and guidance throughout the years.

I owe sincere and earnest thankfulness to all the anonymous reviewers for their suggestions to improve my work. Special thanks goes to Sabine Biewendt for their organizational and administration support over all the years. It is a great pleasure to thank everyone who supported me during writing this dissertation, who proofread my thesis and gave comments and ideas for improvement. This work has been partially funded by the German Federal Ministry of Education and Research (BMBF) as part of the InnoProfile research group "3D Geoinformation" ([www.3dgi.de](http://www.3dgi.de)) and "4D-nD GeoVis" ([www.4dndvis.de](http://www.4dndvis.de)).

I finally like to express my deepest gratitude and love to the most important people in my life: my family and friends; especially to Sabine Pommerening for her long-standing support.

Potsdam, Germany, January 23, 2013

*Matthias Trapp*

# Contents

<b>Frontpage</b>	<b>i</b>
<b>Dedication</b>	<b>ii</b>
<b>Copyright</b>	<b>iii</b>
<b>Abstract</b>	<b>iv</b>
<b>Zusammenfassung</b>	<b>vi</b>
<b>Acknowledgments</b>	<b>viii</b>
<b>Contents</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Geovisualization and 3D Geovirtual Environments . . . . .	1
1.2 Focus+Context Visualization of 3D Geovirtual Environments . . . . .	4
1.3 Advances in Hardware-Accelerated Real-time Rendering . . . . .	5
1.4 Problem Statement and Contributions . . . . .	8
<b>2 Overview of Focus+Context Visualization</b>	<b>13</b>
2.1 Lens-based Focus+Context Visualization . . . . .	13
2.2 Distortion-based Focus+Context Visualization . . . . .	16
2.3 Highlighting Techniques for Points-of-Interests . . . . .	20
<b>3 Focus+Context Visualization of 3D Geovirtual Environments</b>	<b>25</b>
3.1 Categorization of Rendering Techniques . . . . .	25
3.2 Preliminaries and Classification Criteria . . . . .	27
3.3 Focus Types for 3D Geovirtual Environments . . . . .	28
3.4 Representation of Focus Types . . . . .	29
3.5 Visual Variants for Focus and Context . . . . .	32
3.6 Separating Focus from Context . . . . .	33
3.7 Summary . . . . .	37
<b>4 3D Generalization Lenses</b>	<b>39</b>
4.1 3D Lenses and Level-of-Abstraction . . . . .	39
4.2 Generalization of Virtual 3D Landscape and City Models . . . . .	40
4.3 Generalized Clipping . . . . .	42
4.4 Concept of 3D Generalization Lenses . . . . .	50
4.5 Multi-pass Rendering . . . . .	51
4.6 Usage Scenarios and Discussion . . . . .	53
<b>5 Dynamic Mapping of Raster Data</b>	<b>55</b>
5.1 Decoupling Geometry and Texture Data . . . . .	55
5.2 A General Concept for Projective Mappings . . . . .	57

5.3	Real-time Rendering of Projective Mappings . . . . .	58
5.4	Application Examples . . . . .	61
5.5	Conclusions and Future Work . . . . .	63
<b>6</b>	<b>Rendering Technique for Image-space Deformations</b>	<b>65</b>
6.1	Non-planar Single-center Projections . . . . .	65
6.2	Real-time Rendering of Non-planar Projections . . . . .	66
6.3	Generalization of Single-center Projections . . . . .	68
6.4	Interactive Rendering Process . . . . .	71
6.5	Stereoscopic Rendering Non-planar Projections . . . . .	72
6.6	Extensions for Stereoscopic Rendering . . . . .	74
6.7	Comparison of Image-based and Geometry-based Approaches . . . . .	76
6.8	Results and Discussion . . . . .	78
<b>7</b>	<b>Multi-perspective Views for 3D Geovirtual Environments</b>	<b>79</b>
7.1	Multi-perspective Views . . . . .	79
7.2	Effective Presentation of 3D Geovirtual Environments . . . . .	81
7.3	View-dependent Multi-perspective Views . . . . .	83
7.4	Multi-scale Rendering . . . . .	86
7.5	Summary and Discussion . . . . .	89
<b>8</b>	<b>Highlighting Techniques for 3D Geovirtual Environments</b>	<b>91</b>
8.1	Applications and Challenges of Object Highlighting . . . . .	91
8.2	On-screen Highlighting Techniques . . . . .	92
8.3	Off-screen Highlighting Techniques . . . . .	105
8.4	Summary and Future Work . . . . .	110
<b>9</b>	<b>Case Studies and Applications</b>	<b>111</b>
9.1	Communication of Digital Cultural Heritage in Public Spaces . . . . .	112
9.2	Non-planar Projection Surfaces . . . . .	118
9.3	Multi-perspective Views for Navigation Systems . . . . .	119
<b>10</b>	<b>Conclusions and Future Research</b>	<b>123</b>
	<b>References</b>	<b>127</b>
	<b>Publication Overview</b>	<b>147</b>
	<b>Eidesstattliche Erklärung</b>	<b>151</b>

## Chapter 1

# Introduction

The expression "focus+context" describes the concept of visually discriminating interesting objects – the *focus* – from nearby related objects – the *context*. This chapter introduces the basics of 3D geovisualization using 3D geovirtual environments, such as virtual 3D city and landscape models. Next, it discusses potentials of rendering techniques for focus+context visualization of 3D geovirtual environments – the core topic of this thesis – and how these techniques facilitate design and construction of effective user interfaces for these environments. A brief overview of the state-of-the-art hardware-accelerated rendering pipeline is presented, on which the implementation of the rendering techniques rely on. The chapter closes with a problem statement and a list of contributions. This thesis is partially based on the author's scientific publications, which are listed in the publication overview.

## 1.1 Geovisualization and 3D Geovirtual Environments

The aim of visualization is to gain insights and understanding to data, which is invisible or of abstract nature, by forming a mental vision or picture, and thus make it visible to the mind or imagination [267]. It "attempts to display structural relationships and context that would difficult to detect by individual retrieval requests" [209]. This process can use computer generated images or collections of images, possibly ordered, using a computer representation of data as its primary source and a human as its primary target. It requires a suitable mapping from these computer representations to perceptual representations by choosing encoding techniques to maximize human understanding and communication.

**Geodata** Geodata (also spatial or geographic data) refers to and occupies geographic space, has an explicit spatial location according to a geo-referencing system, and consist of spatial and non-spatial elements. Spatial elements describe geometry (e.g., line, point, polygons) as well as topology, while non-spatial elements include descriptive data. Today, it is assumed that more than 80% of the data available can be interpreted as geodata.

Due to the fact that geographic space is continuous and infinite with respect to resolution, geodata is in its nature eminently fuzzy [226]. The need to define crisp boundaries for digital representations between spatial objects complicate rigorous and adequate representations of reality. Spatial data is also fuzzy in terms of interactions between spatial objects in the way these interactions subsist in time. Further, many sources of spatial data are managed and defined by legal regulations and laws (e.g., cadastral databases, borders, postal codes), i.e., their definitions are mandatory and cannot be easily changed according to the demands or capabilities of IT systems and technology. Geodata represents geographic features, objects, and phenomena: their state, geometry, and properties need to be reflected as they are, i.e., the way they modeled is set and defined by themselves. A *feature* represents the central modeling construct for geodata and denotes an abstraction from a real-world phenomena. It is described by geo-referenced geometry and additional feature attributes. Hereby, a *feature type* (e.g., polygon) is distinguished from a *feature instance* (e.g., polygon #23). A *feature attribute* has a name, data type, and value domain associated to it.

Geodata has the following spatial correlation that is important to visualization: "everything is related to everything else, but near things are more related than distant things" ("the 1<sup>st</sup> law of geography" [252]). This proximity relationship refers to closeness: objects or features that may have association because they are spatially near each other. There are three common proximity characteristics: (1) *connectivity* denotes features that connect, or at least touch; (2) *contiguity* denotes the degree of connectivity and (3) *adjacency* describes nearness, or the features that are close to each other. In general, most spatial processes are non-stationary and anisotropic. Geodata is spatially heterogeneous because of the unique nature of each geographic space. This and the lack of stability on the behavior of spatial relationships, i.e., how features relate to each other in space and over time (e.g., distance, distribution, density, and pattern), indicate that spatial data is non-stationarity.

**Geovisualization** Geovisualization, short for geographic visualization, represents a category of visualization and basically deals with understanding our geography [152] by visualizing spatial objects, relationships, processes, and phenomena that reflect the specific nature of spatial data. Among others, this involve topics about the human perception and interpretation process concerning geospatial data. It is based on the specific ways humans can perceive, cognitively process, and think about spatial information [173]. This includes culture and evolution of spatial information communication media throughout the past. To communicates geospatial information and to enable data exploration and decision-making processes, tools and techniques are required that support geospatial data analysis through the use of interactive visualization. Dykes et al. characterize geovisualization as "an emerging domain that draws upon disciplines such as computer science, human-computer interaction design, cognitive sciences, graphical statistics, data visualization, information visualization, geographic information science and cartography to discuss, develop and evaluate interactive cartography" [71].

Key issues in geovisualization are: (1) multivariate, space-time, and multi-scale geographic representation of spatial and spatio-temporal data. This includes data structures, level-of-detail concepts, and data management; (2) design, implementation, and integration of visual, statistical, and computational methods for the representation, analysis, knowledge construction, as well as sense making using both multivariate and heterogeneous geospatial information (geovisual analytics); (3) knowledge management and geo-collaboration to integrate, manage, and access data and knowledge generated by multiple organizations. This concerns computational and human issues that underling technologies have to address, e.g., the facilitation of group work and the development of semantic frameworks; (4) understanding of spatial cognition and human factors to develop methods and tools for interacting with geospatial information as well as (5) the development of methods and tools to integrate and use geospatial information for risk assessment and decision support.

With respect to 3D rendering systems, *3D geovisualization* is closely related to *virtual reality* (VR), where especially *virtual environments* (VE) deal with virtual representations of the real, physical world. Virtual reality is the range of techniques creating three-dimensional visualization [118] and providing three user characteristics: interactivity, spatial dimension, and real-time activity (action feedback is given without noticeable pause) [138]. All in all, 3D geovisualization is a highly multidisciplinary science, using methods and techniques from geographic information visualization, virtual reality, and human-computer interaction. In contrast to 2D geovisualization, the additional third dimension enables better transformations between model, data and reality. With respect to presentation, 3D models are natural and cognitively easier to interpret and are thereby suitable to communicate ideas and visions.

To summarize, 3D geovisualization offers a number of advantages: (1) the human vision is able to quickly explore and interpret a large amount of data that is geometrical represented in a 3D scene; (2) a user can be manipulated through hardware interfaces to induce the impression of immersion into a 3D scene, and thereby having a stronger sense of being in a physical world; (3) using 3D, a direct representation of height information can be used [72]; (4) temporal simulations of dynamics using 3D data can deliver more realistic and more exact results than in 2D.

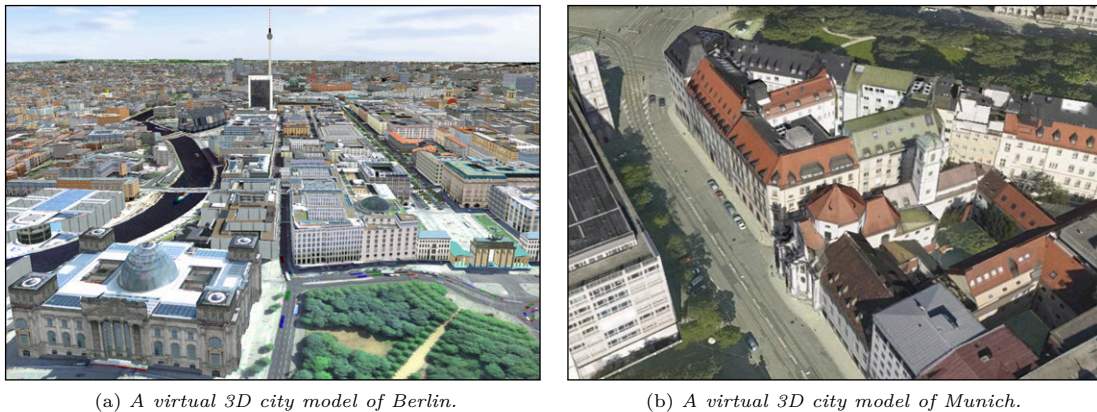


Figure 1.2: Examples of complex virtual 3D city models as instances of 3D geovirtual environments.

**3D Geovirtual Environments** In general, *virtual environments* can be defined as "interactive, virtual image displays enhanced by special processing and by nonvisual display modalities, such auditory and haptic, to convince users that they are immersed in a synthetic space" [77]. Based on 2D and 3D geodata as primary components and contents, a 3D GeoVE or *Geographic Virtual Environment* provides a uniform, seamless conceptual and technical framework to represent and visualize geodata by means of interactive, three-dimensional graphics, based on the virtual-world metaphor (Fig. 1.1). 3D GeoVEs accomplish the integration of geo-referenced thematic information by mapping it to corresponding objects or graphical variables [14] as primary components for their visualization. The main functionality comprises the visualization, presentation, exploration, analysis, use, and management of spatial objects and processes. Thus, a 3D GeoVE serves as effective user interface to static and dynamic spatial information and enables the construction and application of complex geoinformation spaces within a compact, well-defined framework. Using virtual environments as a medium for geovisualization provides more dynamic, more interactive, and more expressive geovisualization by taking advantage of – and focusing – on human sensory and cognitive capabilities. They apply state-of-the-art digital media and information systems technology as well as extend and specialize existing technology to reflect on specifics of geographic visualization and geographic data [172]. Thereby, they integrate findings from related fields, such as computer graphics, image vision, VR, augmented reality (AR), as well as information visualization, cartography, and data mining. One can distinguish between different types of GeoVE: (1) non-immersive 2D, 2.5D, 3D Desktop GeoVE and (2) immersive, 3D CAVE, Workbench/PowerWall GeoVE [171].

3D geovirtual environments (Fig. 1.2), as category of 3D virtual environments, represent essential Geovisualization tools and a general-purpose media to integrate, manage, visualize, and communicate complex geospatial information. They represent efficient tools in the field of geography or cartography, in particular if their visualization and knowledge can be transferred to the 3D visualization domain [136]. MacEachren et al. proposed fundamental characteristics of virtual environments that separate them from traditional, static representations and contribute to the sustainment of virtuality [172]. These are: (1) *immersion*, the sensation of "being in" the environment by involving more human senses; (2) *interactivity* expressed by the ability of a user to choose arbitrary viewpoints, interact with objects and manipulate the environment directly; (3) *information intensity* by adjusting information detail (e.g., by using level-of-detail representations) adaptively and by configuring of the degree of information intensity and themes; and (4) *intelligence* comprising the context sensitive behaviors of objects and their active assistance for navigation and interaction. This thesis focus on two specific variants of 3D geovirtual environments: virtual 3D city models and landscape models.

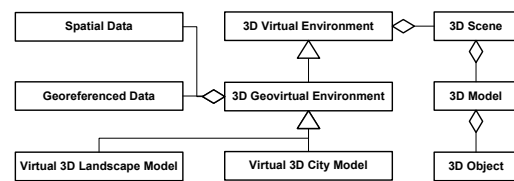


Figure 1.1: Overview of terms with respect to virtual environments and their representation.

*Virtual 3D landscape models* serve as frameworks for representing geographic and thematic aspects of landscapes. They are based on 3D geodata including high precision terrain and surface models (DTM, DSM), 3D building, site, and city models, 3D vegetation and water models, as well as photographic textures to model the appearance of these components. This way, virtual 3D landscape models achieve a high degree of realism as required by a number of applications in virtual reality, design, and architecture [GTD11]. *Virtual 3D city models and systems* are characterized by an underlying 3D terrain model, embedded 3D buildings models, 3D vegetation models, as well as complementary street and green space areas. Virtual 3D city models can be represented using the CityGML format [103], standardized by the Open Geospatial Consortium (OGC). CityGML is a common information model for the representation of 3D urban objects. It defines classes and relations for the most relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantical, and visual properties. Included are generalization hierarchies of thematic classes, aggregations, relations between objects, and spatial properties. This persistence of thematic information goes beyond the capabilities of standard graphic exchange formats and enables the application of virtual 3D city models in manifold domains.

For example, they are used in architecture [59, 60], urban or landscape planning and development [221], to visualize prospect changes or modifications. They serve as decision support and controlling systems for disaster management [65], urban security [270] and as integration platform for traffic simulation systems [279]. Further, these systems are applied in infrastructure management and planning, digital cultural heritage [TSP<sup>+</sup>10], as well as environmental monitoring and controlling [127]. Furthermore, they are used in city and tourism information systems [225, 100].

## 1.2 Focus+Context Visualization of 3D Geovirtual Environments

Due to perspective transformations of the input geometry and the limitations of current display devices, 3D GeoVEs exhibit a number of drawbacks: occlusion (especially in low or near ground perspectives) hinders the user to perceive potentially important information. Further, the low resolutions of computer screens in contrast to print media limits the amount of information displayable [136]; this is problem is reinforced due to perspective transformation, which decreases object sizes with increasing distance to the center-of-projection (COP) [85]. Furthermore, the complexity of orientation and navigation tasks in 3D often leads to "getting lost situations" [37].

In addition to that, one of the fundamental problem in visualization is *scale*, i.e., too many data sets are too large to visualize on a single or even a number of display screens. In general, this can be interpreted as a problem of *space and order encoding*. Space and order encoding transforms data from an information space to spatial representations (size and order) in a display space that preserves the informational characteristics of the dataset and facilitates human's visual perception and understanding of the data [267]. There are two challenges for efficient spatial encoding: (1) visualizing large information space (large maps, tables, documents, etc.) using a relatively small display screen (also called screen-real-estate problem) and (2) visualizing multi-dimensional data ( $n > 3$ ) in 2D or 3D space. Dürsteler describes that "the main problem of information visualization is the insufficient space, which restricts the user in showing detail and context contemporaneously, is called 'presentation problem'" [70]. The important problem of finding an effective and efficient spatial representation of the information is difficult and can be considered as the most important tasks in information visualization.

Despite the approaches of non-projective mappings between  $n$ D and 2D or 3D (e.g., parallel coordinates plots [137]), *Overview+Detail Visualization* is one approach to the challenges and drawbacks stated above. The basic idea is to show selected regions-of-interest (ROI) at high detail and preserve a global overview at reduced detail. This principle basically uses *different views, windows, or viewports* to display both, details and overview. The overview can be an inset panel, a panel beside the detail view, or another window in the case of a multi-window application. Tufte uses the terms "micro and macro readings" [255] to describe a similar concept for printed maps, diagrams, and other static information graphics. The user has the large structure in front of them at all times, while being able to peer into small details at will.



Similar to overview+detail visualization, *Focus+Context Visualization* (also synonym for *Detail-in-Context*) offers approaches for the first challenge by using spatial distortions, lenses, or overlays to efficiently use the available screen space. Focus+context systems enable a user to perceive detailed information linked to and embedded *within* a context. Focus+context visualization share the same basic idea as overview+detail, but with one key difference: overview and detail views are combined into a *single* image on a single viewport. Card et al. describe three premises of focus+context visualization: "First, the user needs both overview (context) and detail information (focus) simultaneously. Second, information needed in the overview may be different from that needed in detail. Third, these two types of information can be combined within a single (dynamic) display, much as in human vision" [41]. Hauser summarized the principle by having a "combination of both a general overview as well as a detailed depiction within one view of the data at the same point in time" [115]. Since this approach depicts how important information relates to the entire data structure, it potentially supports the human's natural vision system and reduces the cognitive load often induced by switching between multiple views [8].

Based on the definitions above, the following terminology for focus+context visualization is used throughout this document: **Focus**, plural foci or multi-focal, describes the referenced spaces, i.e., regions-of-interest or volumes-of-interests as well as features that are of particular interest to the user in a certain usage scenario. Section 3.3 describes different focus types, their representation and transformations for rendering techniques described in this thesis. **Context**, only singular, denotes the space and containing features in which the focus or foci is/are embedded in. Often the visual representation of the context varies from the one of the foci. Section 3.5 describes these representation variances in more detail. Finally, **Focus+Context** describes the category of visualization concepts that embeds foci and context within single images or views.

The seamless and smooth embedding of focus representations into a context requires specialized rendering techniques. Especially for 3D GeoVE, the handling of complex geodata, the integration of visualization concepts into the visualization pipeline (Chapter 3), the support and leverage of the hardware-accelerated rendering pipeline, and the integration into existing frameworks represent major challenges in designing and constructing systems and applications.

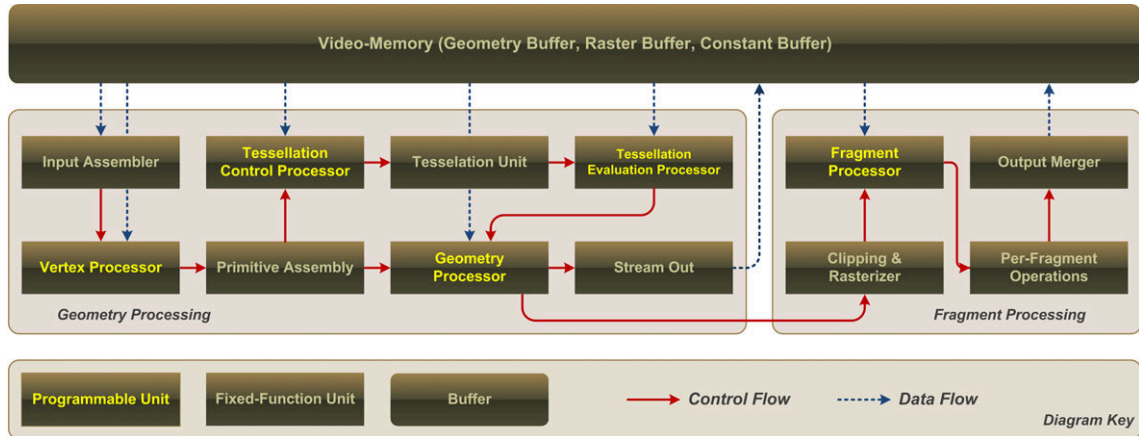
## 1.3 Advances in Hardware-Accelerated Real-time Rendering

*Computer graphics* is concerned with the visual representation of and interaction with real or virtual worlds. Its purpose is to render a *computer generated image* that is often called a *frame*. A sophisticated overview of the foundations of computer graphics can be found in [85]. In general, one can distinguish between two major rendering approaches: *ray-tracing* and *rasterization*. Real-time computer graphics or *on-line rendering systems* differs fundamental from traditional *off-line rendering systems* (usually non-real-time graphics systems). The latter typically rely on ray-tracing that is often a computational expensive operation, which can require minutes or hours for a single frame. *Rasterization* (or rasterisation) denotes the process of converting a scene described by polygonal shapes into a raster image (pixels or dots). This thesis focuses on real-time rendering techniques using rasterization-based graphics hardware (GPUs) [3].

### Real-Time Rendering and Interactive Graphics

If an event or function is processed instantaneously, it is denoted to occur in real-time. *Real-time computer graphics* is a discipline of computer graphics that focus on producing and analyzing images in "real-time": this means the images or frames are rendered "fast enough" that there is no noticeable delay experienced by the user. Frame-rate metrics for a real-time graphics system can be milliseconds (ms) or frames-per-second (FPS). In the context of IT systems, the term real-time is used by different references. This thesis considers a rendering technique as real-time if the rendering process is performed with at least 12 FPS, i.e., the time for displaying two consecutive frames does not exceed 100ms.

Another difference between real-time and non-real-time graphics is the interactivity enabled using real-time graphics. In Computer Science the term *interactive* relates to a program or software system that responds to user activity. In the case of interactive real-time computer graphics, usually



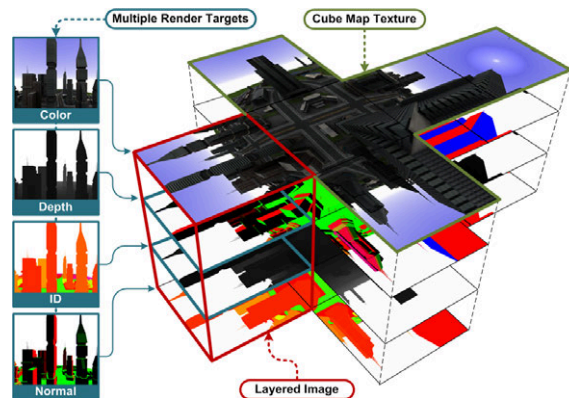
**Figure 1.3:** Simplified flow chart of a programmable rasterization-based rendering pipeline.

a user is in control of what is about to be drawn on the display screen, i.e., the user typically uses an input device to provide input and feedback to the system. In contrast to static media and static representations, 3D GeoVEs enable users to navigate by manipulating a virtual camera.

## Programmable Graphics Pipeline

Figure 1.3 shows a simplified version of a rasterization-based rendering pipeline with programmable units. It reflects a common subset of functionality that can be accessed by low-level application programming interfaces (API) such as OpenGL [230] or Direct3D [20]. It represents the foundations for the rendering techniques and algorithms presented in this thesis.

**Hardware-Accelerated Rasterization** *Rasterization* requires a virtual scene to be decomposed in a number of *rendering primitives* (e.g., triangles or quads). These primitives are transformed and projected from *object-space* coordinates into *screen-space* coordinates. A hardware *rasterizer* generates *fragments*, i.e., pixel values with additional attributes such as depth, normal, etc. that can be displayed on a screen. Prior to display, *per-fragment operations* such as depth, or stencil tests are performed on the fragment. Current hardware comprises multiple, programmable, single-instruction multiple data (SIMD) processing units (*processor*, *shader* or *shader programs*) executed in parallel [144]. *Vertex shader* represent the first programmable pipeline stage, which can be used to modify per-vertex attributes such as position, texture coordinates, or normal vectors. It computes object-space to camera-space as well as the projection transformation. For example, it can be used to perform uniform and non-uniform global deformations [7]. After primitive setup and assembly, a *geometry shader* can be used to amplify geometric primitives [167], perform layered rendering, as well as primitive conversion operations. Finally, a *fragment shader* performs operations on per-fragment basis, e.g., lighting and shading, clipping, or alpha-tests. It can assign output information to different texture layers also known as G-Buffers [222]. This operation is denoted as render-to-texture (RTT) or off-screen rendering [273], a technique often used throughout this thesis. Figure 1.4 shows an example for an image-based data structure and denotes terms used for a G-Buffer comprising color, depth, object identifier, as well as normal vectors.



**Figure 1.4:** Exemplary G-Buffer configuration comprising multiple layered rendering targets of a cube-map texture.

**Rendering Techniques** Given a graphics API, a *rendering technique* comprises strategies, algorithms, and data structures, for the interpretation and evaluation of rendering components [64] with the purpose of synthesizing an image. Rendering techniques are based on the following major rendering components:

**Shader Programs** The configuration of the programmable units of the rendering pipeline (shader source or operation code) as well as the definition of the data that is exchanged between these units are denoted as shader program. They can be compiled and linked at runtime. A rendering techniques can comprise multiple shader programs.

**Geometry and Raster Buffers** One can distinguish between two types of buffers: *raster buffers* and *geometry buffers*. Raster buffers contain raster data (e.g., aerial images, facade textures) and can additionally be used to encode global information of a 3D scene for the programmable units. The maximum resolution and numerical precision is limited and often depends on the hardware generation. Geometry buffers contain geometric primitives (e.g., points, lines, or triangles) and are used as input for the rasterization. Render-to-texture can be applied to generate image-based representation of 3D geometry [222, 74].

**Algorithms and Control Flow** The control flow or application logic of a rendering technique defines the order of operations and the respective *state* of the programmable and fixed functions stages of the rendering pipeline. The change of a configuration is referred to as *state change*. A major aim for designing a rendering technique is to minimize to number of state changes required. Therefore, batching [269] can be applied to reduce state changes per *rendering pass*.

A rendering technique can require multiple passes to render a final image. The number of required rendering passes characterizes the efficiency of a rendering technique and classifies them into *single-pass* or *multi-pass* rendering techniques [56]. Basically, one can distinguish between three types of passes, which usually differs w.r.t. the data types processed and the computational complexity of the operations performed:

**Scene-Rendering Pass** The main function of this type is the rasterization of the complete 3D polygonal scene to the frame buffer or to number of raster buffers. The performance of such pass mainly depends on the geometric complexity of the scene (e.g., the number of geometric primitives or state changes). The results of this pass can be stored in multiple *render targets* consisting of multiple *render layers* [159] (cf. Fig. 1.4).

**Post-Processing Pass** A post-processing pass describes an imaging operation using raster-buffers as input and output. Application examples for post-processing passes are compositing, deferred shading and lighting [145], or screen-space ambient occlusion [123]. Such operations can be performed by rendering a screen-aligned quad textured by a raster buffer [222]. There is a tendency towards image-based algorithms because computations are performed on visible fragments only. This lowers the computational efforts but usually exhibits high memory consumptions. A concept that is often used to reduce memory consumptions are ping-pong-buffers [66].

**Geometry Pass** A geometry pass includes all pipeline processing stages prior to rasterization. The results of these stage operations are written back to geometry buffers using *transform feedback* [230]. This type of pass enables advanced, hardware-accelerated geometry processing operations, such as primitive tessellation and subdivision to ensure sufficient vertex density.

**Forward and Deferred Rendering Pipelines** With respect to the implementation of a rendering technique, one can distinguish between using a *forward rendering pipeline* (FRP) or a *deferred rendering pipeline* (DRP). Both concepts have advantages as well as disadvantages with respect to performance, flexibility, and implementation complexity. This thesis makes use of both concepts (cf. Chapter 3).

The forward rendering pipeline applied in a scene-rendering pass refers to the classic approach using rasterization. One of its problems is less-than-optimal performance for fill-limited 3D scenes, i.e., scenes having a high overdraw ratio (many polygons overlap each another in the projection [223]) and high per-fragment processing costs. A further problem represents the code complexity of the shader programs used. With forward rendering, the transformation of a vertex and the performed lighting computations are bound. This requires all shader programs for a 3D scene to incorporate a high amount of lighting code, which result in shader code that is both complex and hard to manage. Thus, this approach is suitable for low complexity computation at the shader stages.

The deferred rendering pipeline, used to implement deferred shading and lighting, attempts to reduce some of the disadvantages of forward rendering by decoupling the transformation and the lighting of a vertex into two separate phases, i.e., it defers the lighting phase to a post-processing pass. Instead of rendering the 3D scene (including complex lighting operations), information about the 3D scene is gathered using G-Buffers. Deferred rendering has a number of advantages. In general, it is easy to implement and independent of the geometric complexity 3D scene, because only a single scene-rendering pass is required. This introduces only small amounts of shader runtime cost and state changes but requires large amounts of per-fragment data to represent in video memory (e.g., normals for lighting, Fresnel specular coefficients, anisotropic filtering). It can be fill-rate and memory-bandwidth intensive, due to the possible number of post-processing passes introduced.

In this thesis, a mixture of FRP and DRP is applied for the implementation of the rendering techniques. The application of FRP is suitable because most of the geodata visualized does not require complex shading and lighting since the facade textures and aerial images often contain shadows and lighting artifacts. Since the rendering of 3D GeoVE in low perspectives, such as pedestrian views or similar, introduce a high amount of overdraw, DRP capabilities are also employed. The post-processing effects used (e.g., unsharp-masking the depth buffer [170], or image-based edge-enhancement [186]) contribute to provide visual clearance or are required by the respective visualization techniques.

To summarize, interactive visualization systems require real-time rendering techniques that can be efficiently implemented using hardware graphic-accelerators. These techniques are based on data structures and algorithms that take advantage of the massive parallelism of graphics hardware. With respect to this, the goal for an interactive rendering systems are the design of rendering techniques that use a minimal number of rendering passes, feature small memory footprints, and which are capable of being integrated into existing rendering pipelines.

## 1.4 Problem Statement and Contributions

Fuchs et al. argue for the combination of 2D maps and focus+context visualization metaphor [87]. Since data includes both geographic and abstract information at the same time, effective visualization of this data benefits from combining combination of maps with focus+context techniques. The application of focus+context visualization is faced by a number of challenges and problems of 3D GeoVEs. First, the principles can be used to direct the viewer's focus of attention by facilitating the preattentive cognition [49]. Second, it simplifies the exploration and analysis of structures with a large spatial extend. Third, the information visible in a single view can define the size of a problem to visualize. Thus, efficient use of screen space can increase the amount of information visible and thus increase size of the problem to visualize. The focus of this thesis is to develop efficient rendering techniques for focus+context visualization of 3D GeoVE. This section summarizes the conceptual and technical challenges as well as the contributions presented.

### Conceptual and Technical Challenges

The major challenge concerns how principles of existing 2D focus+context visualization techniques can be transferred to 3D GeoVEs, especially to 3D virtual city and landscape models. In contrast to 2D graphics, 3D graphics inherently define scale and ordering of the visualized data by a projection transformation. To perceive 3D space, a number of monocular and binocular depth cues are required (e.g., texture gradient, size gradient, occlusion) [255]. Altering scale and ordering (e.g., by changing object ordering or 3D projection) can cause confusion in the human's natural understanding of this

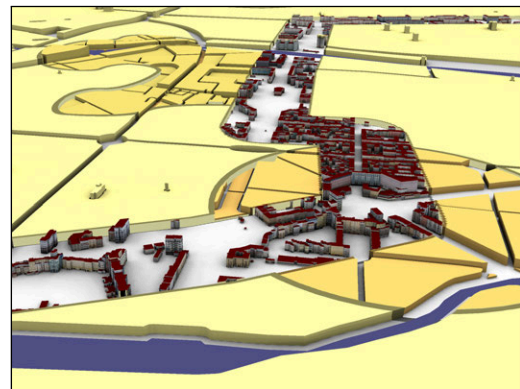
space. The goal is to maintain this understanding of 3D space and improve the communication of complex spatial information simultaneously [6]. A further challenge represents the integration of 3D focus+context visualization techniques into the visualization pipeline [267]. In particular this includes the support for multiple overlapping or nested foci as well as transitions between focus and context regions that are independent of discrete geodata features. Furthermore, the visualization techniques should maintain aspects of immersion, information intensity, intelligence of objects, and interactivity, which are fundamental characteristics of virtual environments [172].

Martin et al. describes the following requirements for a focus+context visualization framework [179]: (1) it should provide visualization at interactive frame rates, (2) has an open architecture for addition of new visualization methods, and (3) has the best possible exploitation of hardware resources. Despite the conceptual challenges of focus+context visualization techniques, the technical challenges for real-time enabled implementations are to maintain these properties. For the real-time image synthesis in particular, this concerns concepts and techniques that integrate not only into the visualization pipeline [267] but also into the real-time rendering pipeline [3]. Due to the high geometric and texture complexity, the rendering of 3D geovirtual environment is often a costly operation [36]. Therefore, the rendering techniques should be capable of processing these in real-time by trying to minimize the number of scene, geometry, and post-processing passes. Further, rendering performance depends on the data structures and their suitable representation for the programmable graphics pipeline: real-time rendering techniques often rely on pre-processed data. To access this pre-processed data at rendering time, suitable data structures are required to store this data efficiently. Therefore, efficient algorithms are required to traverse and use these data structures within a programmable graphics pipeline. In other words, these algorithms are required to work on SIMD architectures and should be applicable to different stages in the rendering pipeline.

## Contributions

The following paragraphs describe what kind of focus+context visualization techniques can be applied to 3D GeoVE. In general, this comprises the combination of real-time photo-realistic rendering techniques, non-photorealistic rendering techniques [38], as well as generalized variants of 3D geovirtual environments to enable saliency-based visualization [STKD12] and to facilitate the interactive exploration of complex information spaces [131]. Also, these techniques handle artifacts introduced by perspective projections, (e.g., dead pixel values) and deal with limited screen space and resolutions [136].

**3D Generalization Lenses** This type of lens-based focus+context visualization enables the combination of different geometric representations of a 3D GeoVE (e.g., different level-of-abstraction) within a single image. Its main purpose is to lower mental efforts of image understanding by reducing visual complexity in the context region while maintaining detailed information in the focus region (Fig. 1.5). A 3D or volumetric lens can be assigned to define a valid area of such a geometric representation within the 3D geovirtual environment. Technically, this real-time rendering technique enables the application of multiple, overlapping or nested 3D lenses using a hierarchical mapping. It further supports arbitrary lens shapes (i.e., convex and non-convex), which are required for lenses along routes. The implementation is based on multiple rendering passes and a volumetric parity test, to perform pixel-precise clipping of an input geometry.



**Figure 1.5:** Example for 3D generalization lenses applied to a virtual 3D city model of Berlin.

**Surface Lenses** Similar to 3D generalization lenses, 2D surface lenses can be used to facilitate perception to guide the user’s attention to highlight selected contents of 3D GeoVE. In contrast to the 3D generalization lenses, focus+context visualization using 2D surface lenses support to flexibly combine raster layer contents. It also facilitates smooth transitions between focus and context areas (Fig. 1.6). Technically, the concept is based on a rendering technique for the dynamic mapping of raster layers using fragment shader programs. This rendering technique enables hierarchical mapping and compositing of multiple raster layer (e.g., maps of different scale or aerial images). In contrast to generalization lenses, the rendering techniques requires only a single rendering pass. Besides focus+context visualization it is applicable to other visualization techniques, such as highlighting purposes or the mapping of dynamic data, which is a requirement for spatio-temporal visualization techniques.



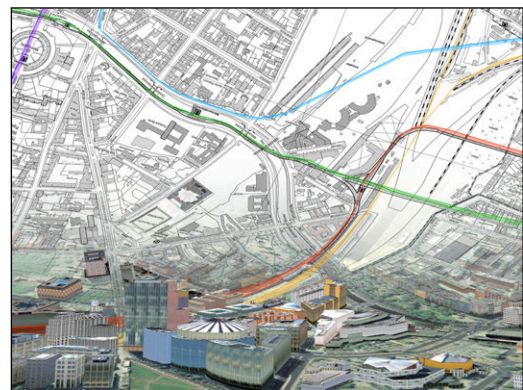
**Figure 1.6:** *Two 2D surface lenses that combine different raster layers along a route.*

**Screen-space Deformations** A classical image-based fish-eye view distorts a single image generated from a 3D scene [275]. To enable multi-focal fish-eye lenses for 3D geovirtual environments, it is required to capture and modify a representation of the complete 3D scene. With respect to focus+context visualization, this contribution presents an image-based approach to efficiently generate multiple 2D viewport lenses and non-planar projections of arbitrary 3D scenes in real-time. Technically, the rendering technique is based on dynamically created raster buffers in combination with shader programs required to compute the specific projections (Fig. 1.7). The rendering technique can be applied within a single rendering pass, is easy to implement, and exploits the capability of modern programmable graphics hardware completely. Further, the presented approach enables the customization and combination of different planar as well as non-planar projections.



**Figure 1.7:** *A single focus fish-eye lens visualization of a virtual 3D city model of Berlin.*

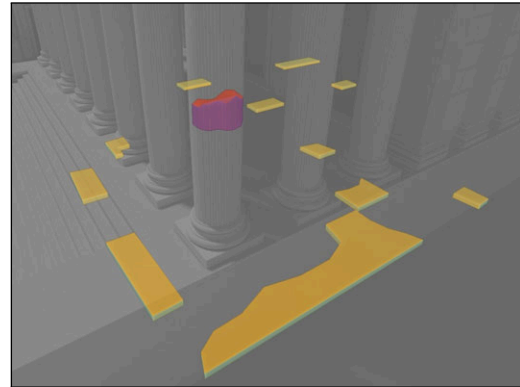
**Object-space Deformations** Multi-perspective projections or views enable visualizations similar to panorama maps, which increase overview and information density in 3D depictions of complex 3D geovirtual environments. Panorama maps seamlessly combine easily readable maps in the foreground with 3D views in the background, both within a single image. Such nonlinear, non-standard 3D projections enable focus+context visualization and new types of cartographic projections that optimize the use of available image space and facilitate information transfer (Fig. 1.8). Technically, the presented technique relies on global object-space deformations [7] to model multi-perspective views while using a standard linear projection for rendering, which enables single-pass processing by graphics hardware. The technique supports different distortion schemata



**Figure 1.8:** *Multi-perspective view using different visual representations.*

beyond classical panorama maps and can seamlessly combine different visualization styles of focus and context regions. Further, this contribution comprises an approach for the automatic and view-dependent interpolation of different configurations of multi-perspective views to enable users to smoothly and steadily interact with, and navigate through multi-perspective 3D GeoVEs

**Highlighting** Highlighting functionality is essential for user interaction and represents an important component of a visualization framework. With respect to focus+context visualization, highlighting enables a user to easily perceive active or selected objects (Fig. 1.9). With respect to 3D GeoVE, highlighting has additional applications: it can be used to visualize data-base queries and to implement navigation aids by highlighting way points or routes. Further, applications of virtual 3D city models (e.g., in car navigation systems, city marketing, tourism, and gaming) rely on effective points-of-interests (POI) visualization. POIs typically represent features relevant for specific user tasks and facilitate effective user orientation and navigation through a 3D geovirtual environment. Therefore, a rendering framework is presented to highlight points-of-interest effectively.



**Figure 1.9:** *Object highlighting for archaeological findings in the project Colonia3D.*

## Remainder

The remainder of this thesis is structured as follows. Chapter 2 gives an overview of previous and related work in the field of focus+context visualization, rendering techniques, and systems that represent the foundations of visualization techniques described above. Chapter 3 present concepts and principles required for the interactive focus+context visualization of 3D GeoVE. Chapter 4 introduces a rendering technique for 3D generalization lenses based on generalized clipping approach. Chapter 5 describes a rendering technique for for implementing 2D surface lenses based on dynamic mapping of raster data to 3D geovirtual environments. Chapter 6 a describes an image-based rendering technique for viewport deformations. Chapter 7 introduces a novel rendering technique for multi-perspective views. Chapter 8 describes rendering techniques for on-screen and off-screen highlighting. At the end of the thesis, Chapter 9 presents application examples and projects that utilize the visualization techniques and underline their relevance for modern 3D geovisualization. Finally, Chapter 10 concludes the thesis and describes future research directions.





## Chapter 2

# Overview of Focus+Context Visualization for Interactive Virtual Environments

This chapter gives an overview to the state-of-the-art of focus+context visualization for 2D and 3D geovirtual environments. The remaining sections presents previous and related work in these categories by briefly introducing the respective concept as well as the implementation and discuss differences and common aspects with respect to the rendering techniques presented in this thesis. In particular, Section 2.1 reviews related work of lens-based focus+context visualization techniques. Section 2.2 presents an overview for distortion-based techniques. Finally, Section 2.3 discusses previous work with respect to object highlighting. The related work concerning the specific rendering techniques are discussed in the respective chapters of this thesis.

The metaphors of focus+context visualization are of broad use in different application domains. Despite geovirtual environments, they are often used to facilitate the navigation in and visualization of documents [210, 121], in (medical) volume rendering [48, 33, 263, 220], as well as for the visualization of large graphs [128] or hierarchies [154]. Leung and Apperley [160] as well as Cockburn et al. [46] present an thoroughly overview and taxonomy of focus+context metaphors not limited by the application domain. With respect to these taxonomies, there are basically three categories of focus+context visualization techniques relevant to the scope of this thesis: (1) *distortion-based visualization techniques*, (2) *lens-based visualization techniques*, and (3) *highlighting techniques*.

## 2.1 Lens-based Focus+Context Visualization

Focus+context interaction techniques based on the lens metaphor are used to navigate and interact with objects in large information spaces. They provide in-place magnification of a region of the display without requiring users to zoom into the representation and consequently lose context [196]. Lens-based visualization providing capabilities for in-place presentation of details in a global context. Interactive lens-based visualization can be applied to explore continuous geospatial representations, as well as non-geospatial visualizations such as network diagrams [5]. Carpendale argue for distinction between *discrete* and *continuous* lenses and provides a taxonomy for the first case [185]. In the latter case, the magnified region is often integrated in the context using smooth transitions based on spatial distortion in order to avoid occlusion of its immediate surroundings.

The *magic lens* metaphor and *Toolglasses* have been introduced by Bier et al. [15]. They describe widgets as interface tools that can appear between an application and a traditional cursor. These visual filters can "modify the presentation of application objects to reveal hidden information, to enhance data of interest, or to suppress distracting information" [16] in the region of interest, which is determined by the shape of the lens. The *through-the-lens* metaphor [240] presents a set of tools that enable simultaneous exploration of a virtual world from two different viewpoints. One is used to display the surrounding environment and represents the user, the other is interactively adjusted to a point-of-interest. The resulting image is displayed in a dedicated window or viewport, similar to overview+detail visualization.

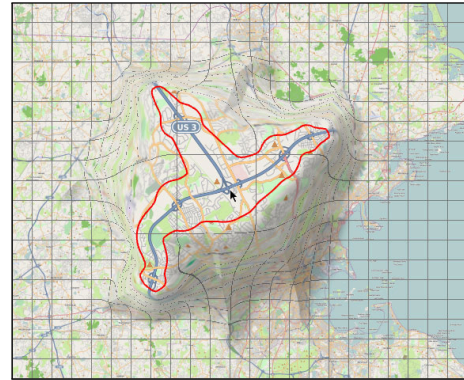
Related work for rendering techniques of lenses can be categorized by the following aspects: the lens type (e.g., volumetric vs. flat or surface lenses), the support of multiple, intersected or nested lenses, as well as lenses of arbitrary shape. For the application in geovirtual environment, the class of lens-based visualization techniques can be differentiated into lens-based visualization techniques for *2D* and *3D* interfaces.

**Lenses for 2D Interfaces** The need to examine and manipulate large surface models is commonly found in many science, engineering, and medical applications. However, on a desktop monitor, depicting the complete model in detail is not always possible. Using a magnifying lens to locally enlarge the focal region of a detailed object is usually helpful and perhaps even necessary. Such local magnification and distortion approaches are also useful for the visualization of a high-resolution model on a low resolution display device. These methods expand the region of interest through the theory of optical lens or other distortion methods to convey a complete visual message to the user and reminds the user of the overall perception of the model at all time, while the user's attention is focused on a local region [266]. Thus, focus+context lens-based techniques smoothly integrate two levels-of-detail using spatial distortion to connect the magnified region and the context [197].

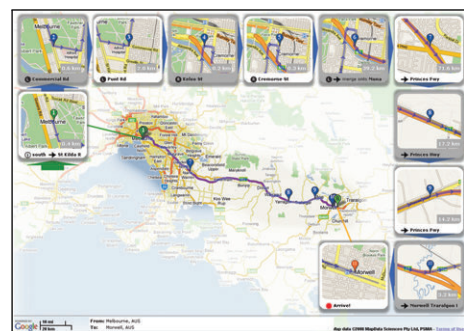
Pietriga et al. present a framework for viewport lenses that can be easily integrated into existing frameworks [196]. It based on the displacement and compositing of pixels from two renderings at different scales uses transparency and image blending to overcome tight coupling between lens techniques and the underlying graphics framework. Spindler et al. presents *camera textures*, a real-time rendering technique for camera deformations that can be applied to lens distortions and non-realistic projections [237]. The technique is based on vertex shader textures and presents a hybrid approach working in image and object space. As the primal deformation is achieved by vertex deformation in object space, the approach does not exhibit artefacts known from pixel-based magnification techniques [275]. However, it requires well tessellated 3D models to produce distortions of high quality. This limitation is caused by the "straight-line segment" problem, which can be partially solved using adaptive tessellations executed on the graphics hardware.

Lam et al. study the effects of 2D geometric transformations on visual memory to provide guidelines for interface design. They report that polar fish-eye transformations had less effect on accuracy than their rectangular counterparts [153]. Further, Appert et al. report on the limited magnification range of focus+context techniques [4]. They enhance interaction techniques by switching between discrete precision levels, decoupling cursor and lens center, and integrating speed-dependent visual behaviors. While distortion guarantees visual continuity, it causes problems of interpretation and focus targeting, partly due to statically-defined, regular lens shapes. Pindat et al. counterbalance this by dynamically adapt the lens shape to the shape of the objects-of-interest [197] (Fig. 2.1).

Brosz et al. introduces the *undistort lens* [31] to complement existing distortion-based techniques to provides a local and separate presentation of the original geometry without affecting any distortion-based lenses used for an detail-in-context visualization. It enables interactive access to the underlying undistorted data within the context of the distorted space, and facilitates a better understanding of the distortions. The work describes the implementation of a generic back-mapping mechanism that enables the implementation of undistort lenses for arbitrary distortion based



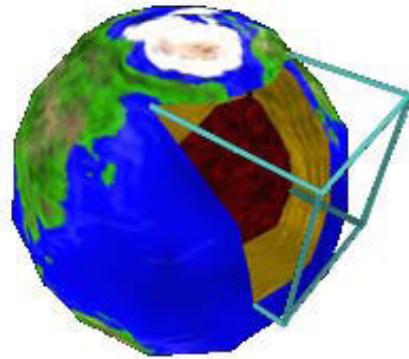
**Figure 2.1:** Example of a content-aware adaptive lens ([197]).



**Figure 2.2:** Example for a route map with detail lenses taken from [141].

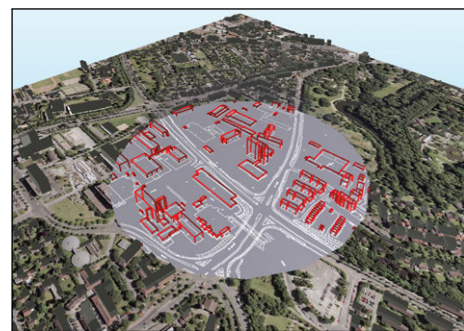
techniques, including those presented in the lens literature. More related to overview+detail visualization and map labeling, Karnik et al. present a method and system to generate printable versions of a route map (Fig. 2.2) that shows the overview and detail views of the route within a single, consistent visual frame [141]. The proposed multi-focus visualization technique layout detail lenses, which encapsulate points-of-interest at a finer geospatial scale, to avoid occlusion with the route and each other, and to optimally utilize the free space around the route rendering (overview). A set of layout metrics are defined to evaluate the quality of a lens layout for a given route map visualization and standard lens layout methods are compared.

**Lenses for 3D Interfaces** Cignoni et al. [45] first published *MagicSphere* metaphor that translates the 2D concept described by Bier et al. [16, 15] to interactive 3D environments. It represents an insight tool for 3D data visualization which is restricted to a spherical shape. The analytical approach classifies the geometry upon its relation to the lens shape (inside, outside, on the border). The rendering is done within two passes, each for every classification. In each pass different visual appearances can be applied. The border geometry is rendered in both of them. The *MagicSphere* metaphor generates visual artifacts at the lens border. Later on, Viega et al. [261] identify two basic categories: *flat* lenses and *volumetric* (3D) lenses (Fig. 2.3). The term *volumetric* is used to describe a 3D lens's volume, and not the nature of the contained data set (here: volume data vs. polygonal data). In contrast to flat lenses, 3D lenses offer a true three-dimensional focus region that is independent of a user's point of view and may therefore be used with head-tracked views and in collaborative VR systems [24]. The implementation is based on infinite clipping planes for the volume faces. This approach is computationally expensive for complex lens shapes. Ropinski and Hinrichs [218] present an algorithm for real-time rendering of volumetric magic lenses that have an arbitrary convex shape, which is fully hardware-accelerated. It supports the combination of different visualization appearances in one scene. The approach uses multi-pass rendering and shadow-mapping to separate focus from context data. A sophisticated overview of 3D magic lenses and magic lights is given in [233]. This work applies this metaphor to immersive building services.



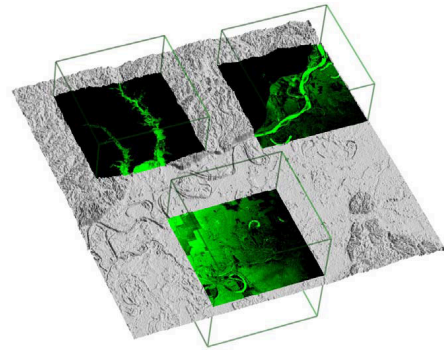
**Figure 2.3:** A volumetric lens cuts into a model of the Earth to reveal the inner structure ([261]).

**3D Lenses for Geovisualization** In [219], a brief overview of focus+context visualization using 3D lenses for geovisualization (Fig. 2.4) is presented. Based on [218], an image-based multi-pass algorithm to separate focus from context regions is presented [216]. The algorithm supports arbitrarily shaped lenses in real-time, but does not handle overlapping or nested 3D lenses. The implementation is based on a two-sided depth test that requires multiple rendering passes [104]. The runtime performance for rendering a single 3D lens depends on the depth complexity of the 3D lens shape, which can vary with respect to the virtual camera [217]. This would result in a loss of interactivity if multiple lenses would be rendered. In [TD08c], a volumetric test is presented that can be used for focus and context separation. Using depth peeling [80], the approach transforms arbitrarily shaped solids (convex and non-convex) into layered depth images [235] and performs ray-casting in 3D texture space to determine the parity of a 3D point to test. This test can be applied at various levels (vertex, primitive, fragment) within the rendering pipeline.



**Figure 2.4:** Example of a flexible and tangible magic for augmented reality [217].

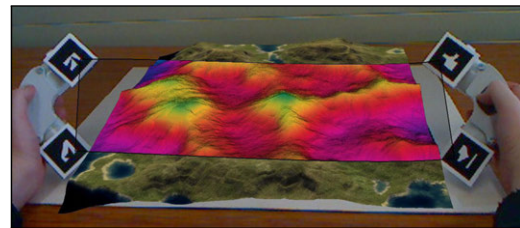
In 2009, Tiesel et al. describe composable volumetric lenses for surface exploration [251] as an interpretational tools for geological visualization such as high-resolution elevation datasets (e.g., LTDAR and SRTM). The lens represents a constrained focus region that provides alternative views of datasets to the user while maintaining the context of surrounding features (Fig. 2.5). The implementation is based on run-time composition of GPU shader programs [24] that implement per-fragment clipping to lens boundaries and surface shader evaluation. The approach supports the composition of lenses based on a user-influenced hierarchical mapping. The technique is extended by [250, 24] to achieve rendering within a single pass. However, the approach was not applied to virtual 3D city and landscape models and omit the usage of different geometries (e.g., level-of-abstraction [93]) for the lens content.



**Figure 2.5:** Example rendering showing three cubic color map lenses [24].

**3D Lenses for Augmented Reality with Applications to Geovirtual Environments** Looser et al. [165] present an approach for 3D flexible and tangible 3D magic lenses in augmented reality in the form of a flexible sheet. It is suitable for focus+context visualization of virtual globes and 3D digital terrain models (Fig. 2.6). The implementation support multiple, non-overlapping lens surfaces that can be directly manipulated using associated interaction techniques.

Similar to this, Kalkofen et al. present an interactive focus+context visualizations for augmented reality, which implementation is based on G-Buffer compositing [140]. They demonstrate how focus+context visualizations are used to affect the user's perception of hidden objects by presenting contextual information in the area of augmentation. The synthetic data is overlaid on top of the real world imagery by taking into account the information that is about to be occluded. Based on the Magic Lens metaphor interactive separation of focus from context is supported.



**Figure 2.6:** Example of a flexible and tangible magic for augmented reality [165].

## 2.2 Distortion-based Focus+Context Visualization

Distortion is a method often used for focus+context visualization. Distortion-based visualization techniques aim to solve problems of presenting large amounts of data in a confined space. The data which is not associated with a focus is suppressed and distorted, while the data of interest is depicted larger and clearer, but may still be distorted, e.g., in poly-focal views.

### Object-space Distortion for 2D Maps

Hauert and Sering present an approach for drawing 2D road networks with focus regions [114]. To avoid that the map gets cluttered, it applies distortion in areas where the road network is sparse (Fig. 2.7). The road network is represented as a graph whose edges are the road segments. A spatial mapping is computed using a graph-based optimization approach that minimizes the square sum of distortions at edges. The optimization method is based on a convex quadratic program, which can be solved in polynomial time. Important requirements on the output map (e.g., no edge crossings) are expressed as linear inequalities. The approach is not real-time capable and the selection of distorted road is



**Figure 2.7:** Scaled focus region (left) of the original map (right) with minimized distortions [114].

capable and the selection of distorted road is

not adaptive to the users context. Further, Wang and Ming-Te present a real-time distortion-based approach for the visualization of complex 2D metro maps on small displaying areas as provided by mobile devices [265]. The important stations along a given travel route are enlarged while other stations are rendered smaller and closer to fit the complete map into a screen. Various map characteristics such as octilinear transportation lines and regular station distances are formulated as energy terms. Least squares solving [175] is used to compute the optimal layout. In addition, stations are labeled according to human preferences, while occlusions, and consistencies of label positions are computed using the graph cuts method [30].

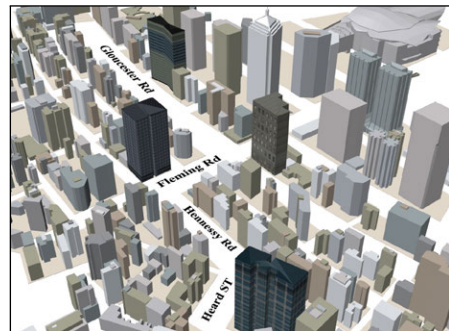
In 2008, Böttger et al. augment schematic maps of transportation systems by superimposing them on street-level maps that are fitted using image-warping techniques [26] to obtain an easily readable transportation network map, which includes all typical city map features such as rivers, streets, and parks without compromising on the schematization. The technique for fitting a street map to a schematic map is based on moving least squares [224] in combination with an overlap control technique. For the interactive exploration, zooming is coupled with warping and control over the level-of-detail. In the same year, Böttger et al. applied nonlinear magnification techniques for the detail-in-context visualization of highly complex satellite and aerial imagery [35] based on the complex logarithm as a transformation for the visualization and navigation [25]. The resulting depictions show details and context with different scales in one seamless image while avoiding local distortions and combine perspective views and classical map projections (Fig. 2.8). The real-time implementation is based graphics hardware and an extended clipmapping technique [245]. However, the proposed technique is not applied to 3D geometry and leverage its full potential only vertical views of a 2D geovirtual environment.



**Figure 2.8:** *Complex logarithmic view of the Metropolitan Museum in New York, in context of Earth [35].*

To facilitate the perception of 3D models, which are distorted or deformed using viewport lenses, Wang et al. presents an approach that enables a user to magnify an area-of-interest while deforming the rest of the scene without perceivable distortion [266]. The method is based on an energy optimization model that shrunk the context area in order to use as little of the screen space as possible to keep the entire model displayed on screen. It rescales different regions of polyhedral model non-homogeneously, which prevents the global bounding space from being expanded. For a given 3D model, a uniform grid space is constructed in a way that the model vertices are embedded in the respective grid cubes. While enlarging the cubes covering the user-specified focal region, the system automatically reduces the other cubes to keep the entire model within the global bounding space. To minimize resulting distortions, a set of energy terms is used to form an optimization system and solve for the grid vertex positions of the deformed space in a least-squares sense. The deformed model is reconstructed by computing each model vertex as a linear combination of its respective set of cube vertices in the deformed grid space.

Similar to this approach, Zhao et al. present a conformal magnifier, an interactive focus+context visualization technique that magnifies a region-of-interest using conformal mapping [283]. The work supports magnifiers of arbitrary shape to magnified the ROI local shape preservation (angle distortion minimization) while globally deforming the context region without cropping and deforming the transition area between the focus and context regions smooth and continuously. For virtual 3D city models, Huamin et al. present a focus+context zooming technique that enables users to zoom into a route and its associated landmarks in a 3D urban environment (Fig. 2.9), which is depicted from a static 45-degree bird’s-eye view [202]. A



**Figure 2.9:** *Example of focus+context route zooming (from [202]).*

grid-based zooming technique is then used to enlarge the landmarks and to minimize distortions to the context buildings. Further, an occlusion-free route visualization scheme adaptively scales the buildings occluding the route to make the route always visible to users.

### Non-Linear Image-based Deformations and Multi-perspective Views

Computer graphics knows three approaches to achieve a panorama effect: multi-perspective images, deformations, and reflections on non-planar surfaces [258]. Multi-perspective images either use non-linear, non-uniform projections or combine multiple images from different viewpoints to create the final rendering. Compared to standard camera models, "multi-perspective camera models are defined less precisely. In practice [...] they are described by constructions" [278]. Deformations distort the landscape before rendering the final image using a standard projection, which implies recomputation of all geometric data for every image. Finally, reflections on non-planar surfaces use standard projections showing an intermediate object that in turn reflects the landscape.

This section gives an overview of research in the fields of single center-of-projection projections and distortions. There is a vast amount of literature covering foundations and applications of non-planar as well as non-linear projections; in [32] a sophisticated overview is presented. To achieve distortions or special projections of the 3D scene, the pinhole camera is extended in several ways. In [11] a procedure is proposed that is based on the computation of new absolute coordinates to be transformed through an adaptive projection matrix. A flexible adaptive projection framework is described by Brosz et al. [32] that enables the modeling of linear, non-linear, and hand-tailored artistic projections. It uses ray-casting and scan line rendering algorithm, whereby polygonal coordinates are changed by a vertex shader. The generality of that framework makes efficient projection difficult, especially for large scale scenes.

Distortions as sub-category image warping are discussed in [107] and [97]. A warping function is applied to each pixel to determine its new color value. An image stitching approach for panorama image generation can be found in [243]. In [256] a method is demonstrated to generate environment maps from fisheye photographs. Besides the issues of nonlinear perspective deformation described in [275, 109, 276, 242]. These approaches use a regular mesh textured with a 2D texture that contains the rendered scene or an image. The displacement of the mesh vertices together with the texture mapping process generates the particular distortion effect. These approaches are limited regarding the FOV which can be achieved. Carpendale researched the usage of image deformation in the context of information visualization [42]. The application of fisheye views in information visualization is discussed in [204]. Applications for view distortions in ray-tracing software are described in [1, 50].

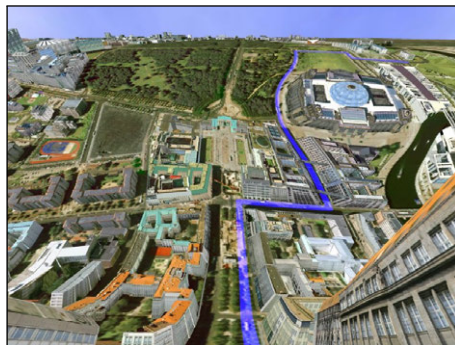
**Panoramic Imaging and Non-linear Perspectives** Panoramic maps were introduced by H.C. Berann [12]. He combined handcrafted geographic with terrestrial depictions and different projection techniques to generate a new kind of map, which assists the user in his orientation task. This work was time-consuming and tedious. Premoze introduced a framework for the computer aided generation of panoramic maps [201]. It offers tools to assist the map-maker in the work flow of the hand-tailored maps. A semi-automatic approach to generate panoramic maps, which relies on global deformations, is presented in [244]. Falk et al. introduced a semi-automatic technique based on a force field that is extracted from the terrain surface [82]. Degener and Klein concentrate on parameters like occlusion and feature visibility in their automatic generation of panoramic maps [54]. All approaches combine non-linear perspectives in one final image, but rely on different techniques. Non-linear perspectives can be achieved with different techniques: (1) using non-standard, non-linear projection to produce a non-linear perspective image, or combine several images taken from different perspectives ([2, 236]); (2) reflection on non planar surfaces and (3) local or global space deformation [258]. The combination of different images to one final image as used in [2] and [236] can also be expressed by a space-deformation as introduced in [7].

The Single Camera Flexible Projection Framework of [32] is capable of combining linear, non-linear and handmade projections in real-time. The projections are described by a deformed viewing volume. Similar to free-form deformation (FFD [229]), the view frustum serves as lattice.

Objects or viewing rays are deformed according to the deformation of the lattice. For the occlusion free visualization of driving routes Takahashi et al. rely on global space deformation [244].

On the one hand the mentioned techniques offer a broad and flexible definition of the projections, which enables the user to control nearly every facet of the final perspective. On the other hand a large number of non-intuitive parameters have to be controlled. Brosz et al. abstracts from these parameter by using a lattice [32].

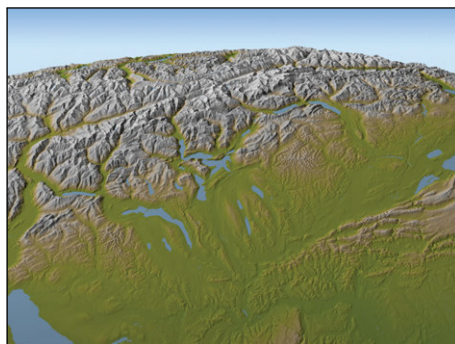
**Global Deformations** Previous work has shown that global deformation applied to such environments can be used to assist way finding and navigation by making effective use of the available image space [182, 169] and by reducing occlusions [202, 55]. Grabler et al. [100] demonstrate that the usage of multi-perspective views in combination with cartographic generalization techniques such as simplification and deformation is suitable to convey important information with in 3D tourist maps. Techniques implementing multi-perspective views can be classified as multi-pass or single-pass. Multi-pass techniques create several intermediate images that are blended in a final compositing step. Each intermediate image requires separate data processing, which is rather expensive when it comes to complex 3D geovirtual environments. Specifically, out-of-core algorithms can incur additional penalties because rendering of intermediate images often significantly reduces caching efficiency. Additionally, image quality suffers due to resampling in the compositing step. Single-pass techniques do not exhibit these disadvantages, yet they require customization of the rendering process available only in software rendering (e.g., ray tracing) until recently. With the advent of a programmable rendering pipeline on GPUs the implementation of interactive single-pass multi-perspective view techniques becomes feasible.



**Figure 2.10:** *Example of a multi-perspective visualization taken from [182].*

The work of Lorenz et al. [169] uses global deformation to generate non-linear perspectives. The geometry is mapped on two different planes, which are connected by a Bézier surface. The planes may vary in tilt, allowing for a combination of two different perspectives. Similar to panorama maps, a mixture of cartographic maps and aerial images is used. The different stylization are seamlessly blended in the transition between the planes. Möser et al. [182] extend this idea by using a more flexible Hermite curve to control the deformation (Fig. 2.10). They rely on a combination of aerial images and topological maps to apply a kind of generalization in the more distant parts of the scene.

In the area of cartography, Jenny and Patterson introducing the plan oblique relief [13, 129]. This digital technique renders 3D terrain on otherwise planimetric (conventional flat) maps. Plan oblique relief presents topography on small and medium scale maps. Here, mountains are shown in a side-view, in partial profile. It uses a conventional orthometric ground plane, that presents the mapped area without distortion. This results in major map features, such as country borders or river networks, to appear as in conventional 2D maps while elevations are seen laterally in 3D. This work was extended further by Jenny et al. By enabling the interactive design of 3D maps with progressive projections [130], that are traditionally used by panoramic landscape painters. They describe advantages of this specific projection and review implementation approaches. The interactive bending of a terrain model into a progressive view, by using global deformations with options to add a curved horizon (Fig. 2.11), to vertically exaggerate the terrain, and to create a 360° strip panorama.



**Figure 2.11:** *Example of a Progressive perspective with curved horizon (from [130]).*

Rademacher introduce a view-dependent variations of multi-perspective views [203]. He defines key-deformation with associated key viewing points. Depending on the current viewpoint the key-deformations are interpolated. A similar approach is used by [40] for interactive stylized camera control. Another view-dependent variation of deformations is discussed in [178]. Here, the global deformation is modified by a view or distant-dependent control function that can depend on a virtual camera. Veas et al. use multi-perspective views to improve site understanding for outdoor augmented reality applications by extending overview, particularly over large areas [260]. Further, Schwarz et al. apply this principle to vertically curved displays [227]. A visualization concept called *perspective+detail* extends the conventional overview+detail pattern by combining perspective viewing with a text-based area containing partial details. They reveal that using a vertically curved display mitigates known problems and assists users by supporting the user's orientation within the information space.

## 2.3 Highlighting Techniques for Points-of-Interests

With the increasing amounts of geographic data visualized on a wide range of display sizes for variable applications demands the avoidance of too complex, too detailed and too dense visualizations [206]. Here, effective and efficient highlighting can enable users to quickly locate and easily decode relevant geographic information. In [190] the highlighting of scene elements, such as local and global landmarks, is considered as navigation aid that "clearly helps improve orientation in the virtual model of a real city". In [136] it is argued that, despite building aggregation and simplification, appropriate highlighting can compensate the dead value areas in virtual 3D city models. Further, Bogdahn and Coors state that "highlighting of complete buildings using false colors might be a first step. However, in a dense urban environment and for pedestrians it could be necessary to provide more sophisticated visual hints, like highlighting the correct entrance to a big building or a building complex" [23]. Robinson documented the application of highlighting techniques for information visualization in 2D GeoVE [211].

**Applications to Landmark Visualization** The management of landmark objects in maps and map-like visualizations is an ongoing major challenge for effectively providing location-based services [44, 238]. Generally, many accentuation techniques have been developed like symbols, annotations, and hybrid perspectives [156], which are difficult to transfer to 3D geovirtual environments. Vinson presented design guidelines for design and placement of landmarks in virtual environments to ease navigation [262]. They comprise among others that (1) landmarks should be visible at all times, especially at all navigable scales; (2) they should be distinguishable from their environment, e.g., other buildings; and (3) a concrete depiction of objects should be preferred over abstract ones for landmarks. Elias et al. analyzed different graphical representations of landmark buildings ranging from photo-realistic to more abstract icons to plain text [75]. They introduced a design matrix to help choosing the appropriate representation for different categories of buildings (e.g., commercial buildings, visually outstanding buildings). Lee et al. suggested depicting landmark buildings by placing photographs in the scene, which have been took from a similar perspective [156]. Textual and 2D image landmark representations lack the depth and context needed for humans to reliably recognize 3D landmarks. *Worldlets* [79] describe a 3D thumbnail landmark affordance. It represents 3D fragment of a virtual world and enables first-person, multi-viewpoint representations of potential destinations.

### On-Screen Highlighting Techniques

This section focuses on highlighting of objects that are located on screen. In contrast to approaches for off-screen location visualization, on-screen highlighting techniques should enable the estimation of the 3D position, as well as the dimensions of an object. There are a number of different approaches for 3D object highlighting and selection preview, which are mostly inferred from 2D visualization. In early stages of 2D computer graphics, different visual representations of objects are used, i.e., sprites or billboards [3] with different hues. Such an approach is not appropriate for current large-scale data set common in GeoVE.



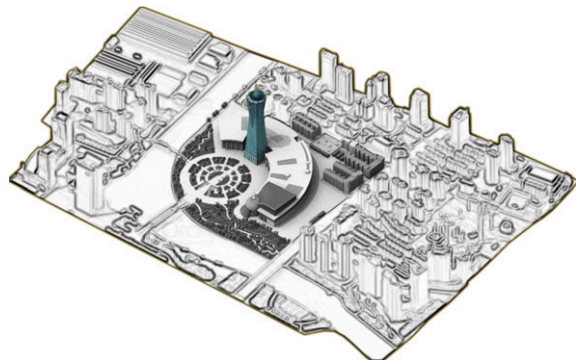
Existing on-screen highlighting approaches that be mainly distinguishes between three types: *style-variance*, *outlining*, and *glyph-based* techniques. Style-variance techniques are based on modifying the appearance of an object that results in an obvious distinction from the rest of the scene. The highlighting hint depends directly on the object and is therefore referred to as *direct hint*. Outlining techniques achieve this effect by enhancing the outline or silhouette of an object. This setting is denoted as *attached hint*. Finally, glyph-based techniques rely on icons or glyphs that are attached to the object to be highlighted. These kind of techniques deliver an *indirect hint* to the user.

Probably the most widespread highlighting techniques are instances of style variance techniques. The highlighting effect is achieved by modifying the appearance in which an object or scene is usually depicted. Such appearance modification can be arbitrarily complex, or as simple as overdrawing the same object geometry with a dominant color that can easily be distinguished from all other colors in the depicted scene. Modifications can be applied to an object or area that should be highlighted (focus-based) or to the remaining scene (context-based).

**Focus-based Style Variance** In 3D virtual environments, focus-based style variance techniques are suitable to be applied to objects that are not occluded. Besides color overlays, the appearance can be modified by using different rendering styles, such as wire frame rendering known from standard 3D modeling tools. Another classic rendering approach, which was often used in 2D sprite-based games, enfoldes different representations of the same sprite/billboard [3]. Due to the potentially massive amount of additional data, the usage of different geometric or image-based representations of each single object is hardly manageable in large-scale 3D GeoVE. Therefore, the style variance is created dynamically, e.g., by blending the standard appearance with a highlighting color. More advanced rendering techniques can be used, such as real-time non-photorealistic rendering (NPR) effects [49] that can be controlled manually by an artist or automatically via style transfer-functions [34].

**Context-based Style Variance** Techniques of this category convey the appearance of the objects to highlight, while modifying the surrounding scene context in a way that the objects-of-interest are emphasized, e.g., using vignetting or semantic depth-of-field (SDOF) [149, 151].

Many applications of geovirtual environments involve map visualization but almost all of them display the map in the same style, which often cause information overloading [STKD12]. For 2.5D geovirtual environments, Wang et al. present a user-centered 2.5D focus+context map visualization technique (Fig. 2.12) that enable the customization of a map according to user requirements [265]. The system automatically constructs a hierarchical representation of a virtual city model according to the user's focus point by using a R-trees data structure [108]. The map is presented in multiple rendering styles: focus information is highlighted and less important information is deemphasized. They approach also uses an additional "landmark margin" containing thumbnails of photographs for nearby points-of-interests, which illustrates the context information and enables users to maintain a macro view of the city. With respect to 2D and 3D maps based on geovirtual environments, Zipf et al. present the concept of *focus maps* [285, 111]. Focus Maps support the user in reading a map by emphasizing regions that are relevant to the user, whereas regions, which are not relevant, are de-emphasized to avoid distraction of the user's attention. Emphasis or highlighting can be achieved by different cartographic scopes of design, e.g. different coloring or different detail in representation of important regions. Cole et al. generalized this idea and take virtual 3D city models into account [49]. They present an advanced rendering technique that modifies the quality and intensity of edges, color saturation, and contrast.



**Figure 2.12:** Example of a 2.5D focus+context map visualization taken from [265].

Finally, *semantic depth-of-field* (SDOF) rendering utilizes a well-known method from photography and cinematography (depth-of-field effect) for information visualization, which is to blur different parts of the depicted scene in dependence of their relevance. Independent of their spatial locations, objects of interest are depicted sharply in SDOF, whereas the context of the visualization is blurred [149, 150]. Evaluations of this technique prove that the SDOF concept is pre-attentive and that it supports directly the perception of sharp target items when the context is blurred. SDOF can significantly support users in focusing on relevant data and guide their attention [89].

**Outline-based and Glyph-based Highlighting Techniques** Depending on the application, the change of style is not always appropriate or possible. Especially in 3D GeoVE, it can be desirable to convey the appearance of the object and the surrounding scene, such as its facade information which could be essential for orientation and navigation. In such use-cases, outlining techniques can be applied that enhance or modify the contours or silhouettes of an object only. Such enhanced contour can possess different line styles and sizes. For real-time rendering purposes, these silhouettes can be rendered using image-based [186] or the more recent geometry-based [117] approaches. Further, rendering techniques for image-based glow [191] are often applied for outlining objects. The attached hint is one of the major advantage of outline techniques: an increased visibility can be gained by increasing the thicknesses of the outline. But increased thickness of the hint also introduces occlusion of the surrounding object area. This can be balanced partially by using a drop-off function, which results in smaller occlusion than occurred for the alternative glyph-based techniques. The application of glow can be considered as a generalized variant of the outlining technique.

Another technique performs highlighting by using additional glyphs or icons, which are placed on top or aside an object, depending on the perspective. The highlighting effect is achieved by the difference between presence or absence of a glyph. This technique is frequently used in strategic games or games employing a third person perspective. Here, usually the orientation of the virtual camera is fixed or limited. One can distinguish between *static* and *dynamic* glyphs, e.g., visugrams introduced in [87]. The latter one includes additional information about the status of an object.

### Off-Screen Highlighting Techniques

Off-Screen highlighting techniques are approaches to emphasize objects located outside the viewport, and hence not visible to the user. Gustafson et al. [105] as well as Burigat et al. [39] classify off-screen awareness approaches into *Overview+Detail* and *Focus+Context* visualization techniques. One can distinguish between three classes of off-screen highlighting techniques: (1) *distortion-based techniques* [134], (2) *partially-out-of-frame techniques* [9], and (3) *glyph-based techniques*.

The concept of partially-out-of-the-frame visualization exploits the human visual system for proxy recognition and interpretation. All approaches described above, rely on an explanation or legend for the user to interpret the abstract shape. These explanations require either screen space or have to be memorized by the user. To overcome these problems, Baudisch and Rosenholtz introduced the 2D *Halo* visualization [9]. Distance and direction are implicitly conveyed by circles around POIs, that reach into the viewport. These partly visible circles are automatically completed through *amodal completion* conducted by the human visual system. The user gets an intuitive imagination of the POI's distance and direction. This *partially-out-of-the-frame* approach is borrowed from cinematography [177]. The benefit is an intuitive and efficient visualization of distance and direction.

Nevertheless, for a high number of POIs, the Halo visualization suffers from cluttering induced due to overlapping arcs. To reduce cluttering Gustafson et al. introduced *Wedge* visualization [105]. A Wedge is an acute isosceles triangle in which the tip coincides with the off-screen POI and the two other corners are on-screen. Consequently, the triangle legs are partly visible and the user is able to determine distance and direction of the associated POI. The overlapping is reduced by the distance-independent and direction-independent adjustment of the Wedge's opening angle and its rotation around the POI. However, Wedges suffer from overlapping and cluttering, too, if the number of POIs is further increased. Besides the off-screen awareness, visualization of POIs has to deal with the problem of occluded on-screen POIs. Elmqvist et al. identify several occlusion

management patterns for reoccurring occlusion problems based on their taxonomy of occlusion management techniques [78]. Glyph-based techniques use abstract shapes that acts as proxy to visualize off-screen points-of-interest. In virtual environments these proxies indicate traditionally direction and distance of the associated POI. For example, popular visualization techniques are Arrows [39], *City Light* [281], and *EdgeRadar* [106].



## Chapter 3

# Focus+Context Visualization of 3D Geovirtual Environments

This chapter introduces the foundations of the interactive rendering techniques for focus+context visualization of 3D geovirtual environments. Section 3.1 presents an overview and classification of these techniques. Section 3.2 states preliminaries and assumptions made for the design of the rendering techniques. Section 3.3 presents an overview and comparison of the applied focus types while Section 3.4 describes their computer graphical representations. Section 3.5 describes the usage of different visual representations that are used to implement focus+context visualization of virtual 3D city and landscape models. Section 3.6 explains methods for separating focus from context. Section 3.7 closes the chapter by presenting a comparison of the rendering techniques based on the presented classifications.

## 3.1 Categorization of Rendering Techniques

Figure 3.1 shows an overview of the rendering techniques for focus+context visualization of 3D geovirtual environments presented in this thesis. It provides a classification into the categories of focus+context visualization as described by [46] and [160]:

**Distortion-based Techniques** With respect to the image synthesis, techniques for *distortion-based* or *distortion-oriented* focus+context visualization [160] can be classified into *object-space* and *screen-space* approaches according to respective reference coordinate system (Section 3.3). The object-space approach distorts geometry prior to rasterization, e.g., by using global deformations [7] applied for the implementation of multi-perspective views in Chapter 7. The screen-space approach uses image-warping techniques [42, 275] that are generalized using projections tiles presented in Chapter 6.

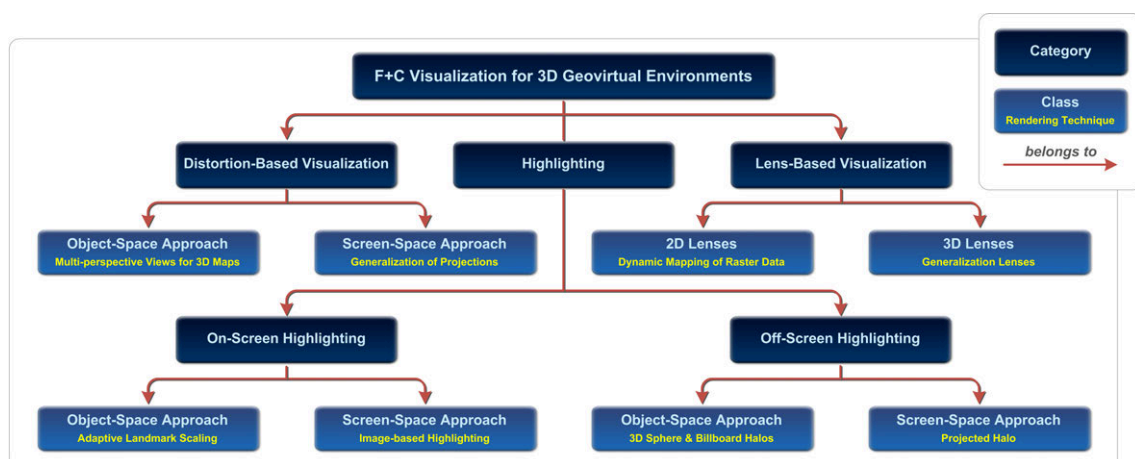
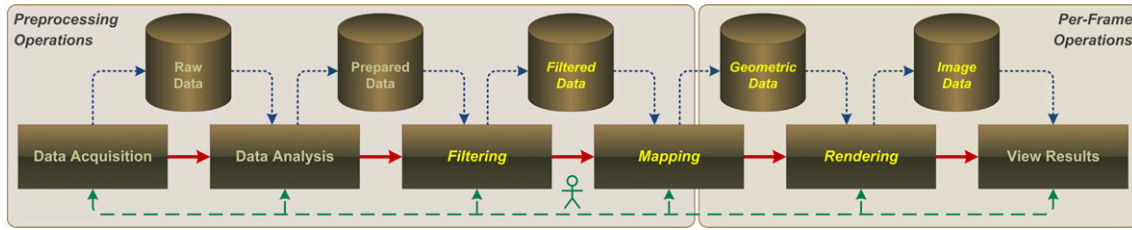


Figure 3.1: Overview of interactive focus+context visualization techniques presented in this thesis.



**Figure 3.2:** Overview of the visualization pipeline after [267]. It describes the step-wise process of creating visual representations of input data. The items marked in yellow are affected by the rendering techniques for focus+context visualization in 3D geovirtual environments.

**Lens-based Techniques** Techniques that use a lens metaphor can be distinguished with respect to the dimensionality of lens contents. For example, 2D surfaces lenses are suitable for applying the lens metaphor to surface related data, such as aerial images or topographic maps applied to a digital surface or terrain model. They can be implemented using the rendering technique described in Chapter 5. A more general approach for applying the lens-based metaphor in 3D GeoVEs are 3D generalization lenses introduced in Chapter 4. They enable the combination of different geometric representations of a virtual 3D city model.

**Highlighting Techniques** Not included the review of focus+context visualization techniques [46, 160] are highlighting techniques. However, Kosara et al. [149] argue for using highlighting techniques to guide a user’s attention to important objects by modifying their context or focus areas respectively. With respect to this, one can distinguish between on-screen and off-screen highlighting techniques to support both, the highlighting of important object that are inside (Section 8.2) and outside the view frustum (Section 8.3).

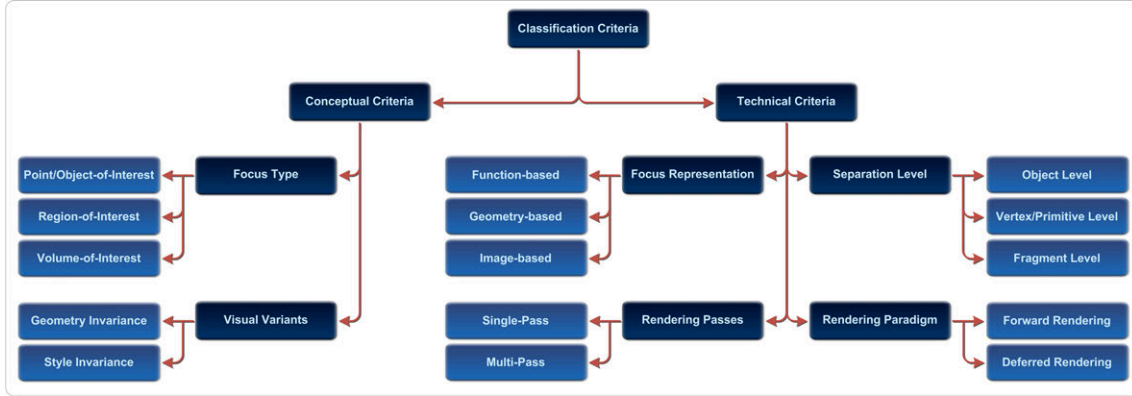
Each of the presented rendering techniques comprises a specific rendering pipeline which is embedded in the general visualization pipeline that is described below.

**Visualization Pipeline** Figure 3.2 shows an overview of the visualization pipeline [267]. It comprises a series of chained data transformations and mappings that can be controlled by the user. The pipeline converts raw data to suitable visual representations for human perception.

In the *data acquisition* stage, the data is analyzed, filtered, and transformed into relational descriptions and representations (including meta data). In the *data analysis* stage, *raw input data* is prepared for visualization (e.g., by applying smoothing filter, interpolating missing values, or correcting erroneous measurements). This stage is usually computer-centered with little or no user interaction. Subsequently, the *prepared data* is filtered in a usually user-centered *filtering stage*, by selecting data portions to be visualized. After this, the *filtered data* is mapped to geometric primitives (e.g., points, lines, polygons) and their attributes (e.g., color, position, size). This *mapping stage* represents a critical step for achieving expressiveness and effectiveness of visualization and is a starting point for visualization design [174]. Finally in the *rendering stage*, *geometric data* is transformed to image data. The rendering techniques presented in thesis mostly affect the *filtering*, *mapping* and *rendering* stage:

**Filtering Stage** In this stage, the prepared geodata is filtered with respect to important (focus) objects or georeferenced regions-of-interest or volumes-of-interest. This includes the selection and identification of focus data, i.e., the portions of the data that should appear at high detail to convey its high information density. It includes the identification of landmarks (Chapter 8) and the gathering of information required for the computation of cell-based generalized variants of virtual 3D city and landscape models used in Chapter 4 and 7.

**Mapping Stage** In this stage, data required for generating visual variants (Section 3.5) for focus and context regions are computed. This comprises cell-based generation of virtual 3D city and landscape models [GTD12], as well as abstracted facade textures [STKD12] or geometry required for edge-enhancement [SHTD12]. Also the preprocessing of focus representations



**Figure 3.3:** Classification criteria of the rendering techniques for focus+context visualization of 3D geovirtual environments.

(Section 3.4) for the respective focus types (Section 3.3) is partially performed in at this stage (e.g., for 3D generalization lenses).

**Rendering Stage** During runtime, the separation methods (Section 3.6) for the respective focus representations (Section 3.4) are applied. The rendering stage also performs transformations of focus representations according to user feedback (Section 8.2).

## 3.2 Preliminaries and Classification Criteria

The rendering techniques presented in this thesis are based on an number of assumptions w.r.t. the geometric input data and the real-time rendering using GPUs.

**Preliminaries** The implementations of presented techniques assume a 3D scene represented by polygonal geometry. More specific, the main rendering primitives are triangles, or related polygonal representations such as triangle strips, or indexed variants. The presented techniques work independent of using *in-core* (i.e., the complete 3D representation fits into main and video memory) or *out-of-core* rendering techniques (i.e., the 3D representation does not fit into main and video memory and strategies for dynamic load of geometry are required [125]). Further, the programmable graphics hardware used for real-time rendering is based on the *unified shader model*. It uses a consistent instruction set across all shader types. Thus, all shader types have (almost) the same capabilities and a common subset of instructions, i.e., they can read from buffers and can perform the same set of arithmetic instructions.

Furthermore, it is assumed that every geometric object can be locally approximated by a plane, denoted as *reference plane*. Every object or feature of a 3D GeoVE is a discrete phenomenon represented by one or more geometric objects that can be identified at runtime using unique identifier. If such instance information is missing for geometry or features, they will be assigned during preprocessing in the mapping stage of the visualization pipeline. Furthermore, the techniques do not use optimizations that would limit or restrict the user during navigation within the 3D geovirtual environment: this includes frequent changes to the orientation of a virtual camera and the application of different projections.

**Classification of Rendering Techniques** The interactive rendering techniques for focus+context visualization can be classified using a number of different criteria shown in Fig. 3.3. On a conceptual level, the techniques support different *focus types* and *visual variants* how focus and context can be visually discriminated or how the data of a focus or context region is presented:

**Focus Types** described in Section 3.3 define the dimensionality and thus the extend of focus regions. In this thesis, focus types are distinguished between *point-of-interests* (POI) and *object-of-interest* (OOI) which refer to feature instances of the 3D GeoVE. Further, 2D *regions-of-interest* (ROI) and 3D *volume-of-interests* (VOI) are distinguished. The focus type

effects technical criteria such separation methods, focus presentation, as well as the rendering paradigm and rendering passes required.

**Visual Variants** described in Section 3.5 can be categorized into *style variances* and *geometry variances* of the 3D input data. The type of visual variant also determines the separation methods, as well as the rendering paradigm and the required rendering passes.

In addition to the conceptual criteria described above, the implementations of the presented rendering techniques can be categorized with respect to the following technical criteria:

**Focus Representations** introduced in Section 3.4 describe how the respective focus types are represented on GPUs using data structures. They are categorized into *functional*, *geometric*, and *image-based* representations.

**Separation Methods** described in Section 3.6 can be classified into *discrete* and *continuous* methods. Continuous methods enable smooth transitions between focus and context, w.r.t. style and geometric scale by preserving the structural coherence of a 3D scene. Discrete methods introduce hard transitions and do not necessarily preserve structural coherence.

**Separation Level** also explained in Section 3.6 describes at which programmable stage of the graphics pipe a separation method is applied in order to discriminate between focus and context. The separation level can be either *per-object* on client-side (CPU) or *per-vertex*, *per-primitive*, and *per-fragment* on server-side (GPU).

**Rendering Paradigm** describes how a separation method is applied and how the resulting visual artifacts are combined into a final image. This thesis distinguishes between using a *forward rendering pipeline* and *deferred rendering pipeline* (compare to Section 1.3).

**Rendering Passes** depend on the visual variant, the separation level, and the rendering paradigm used. Basically, rendering techniques can be categorized into *single-pass* or *multi-pass* rendering techniques. Here, a rendering pass denotes the number of scene rendering pass (compare to Section 1.3).

The remainder of this chapter describes the specification of the major categories and finally present a comparison of the presented rendering techniques in Section 3.7.

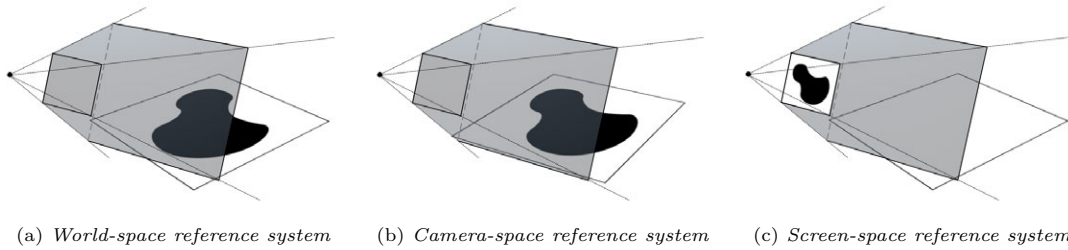
### 3.3 Focus Types for 3D Geovirtual Environments

For 3D focus+context visualization it is important to model and represent the area or regions that define the extend of a focus or the context. This section presents different focus types required for the focus+context visualization of 3D GeoVE. A focus type can possess different focus representations, which in turn determines the separation method and level. Focus types condition the algorithms that use these representations to enable the separation between focus and context regions during real-time rendering (Section 3.6).

**Points-of-Interest and Object-of-Interest** *Point-of-interest* (POI) is a term, typically used to refer to an interesting point on a map – for example city centers or petrol stations. For 3D GeoVE, a POI can be located anywhere in a 3D scene but often near the surface (on digital terrain model or reference plane). Given a 3D view frustum [85], a POI can be visible *inside* the frustum or occluded. Further, it can be *outside* the view-frustum or both (partially-out-of-frame) [9]. Chapter 8 presents strategies and rendering techniques to highlight POI inside the frustum (Section 8.2) as well as outside (Section 8.3), which are especially for 3D GeoVE.

In general, POIs have no geometric extend and are represented by a point in 3D. Therefore, proxy geometry is used for their visualization. Further, a point-of-interest can be a specific object or feature within a 3D GeoVE. In this case a point-of-interest is denoted as *object-of-interest* (OOI). The geometric extend of an OOI can be considered as the region-of-interest for this object. This assumption is necessary to apply appearance-based highlighting techniques presented in Section 8.2.





**Figure 3.4:** Comparison of reference coordinate systems for focus types by the example of a region-of-interest. The depictions shows the center of projection and the view frustum (gray) of a virtual camera.

**Region-of-Interest** In general, a *region-of-interest* (ROI), can be denoted as a selected subset of samples within a dataset identified for a particular purpose. In this thesis, a ROI describes a 2D area of convex or non-convex shape that can encode the degree-of-interest for every point located in that area. 2D region-of-interests require a *reference plane* described in a *reference coordinate system*. Usually, these are screen-space coordinate systems for screen-aligned lenses or world-space coordinate systems when using a reference plane to approximate a 3D geovirtual environment.

Regions-of-interest are used for the rendering of 2D surface lenses and the parametrization of 3D iso-contours (Section 5.4), as well 2D viewport lenses (Section 6.8), or for image-based highlighting (Section 8.2). Region-of-interest are often used for focus+context visualization of 3D GeoVE, because a number of techniques do not really require a 3D representation of a focus. However, there are use-cases that require an additional dimension to define volumetric areas of interest [216].

**Volume-of-Interest** *Volumes-of-Interest* (VOI) are a complex focus type to represent and to handle but enables general applications for 3D geovirtual environments. All point enclosed by a volume are considered as points within the focus. The shape complexity of VOIs vary from simple, convex shapes (e.g., spheres or boxes) to non-convex one, which are able to describe complex structures such as regions around roads [TGBD08].

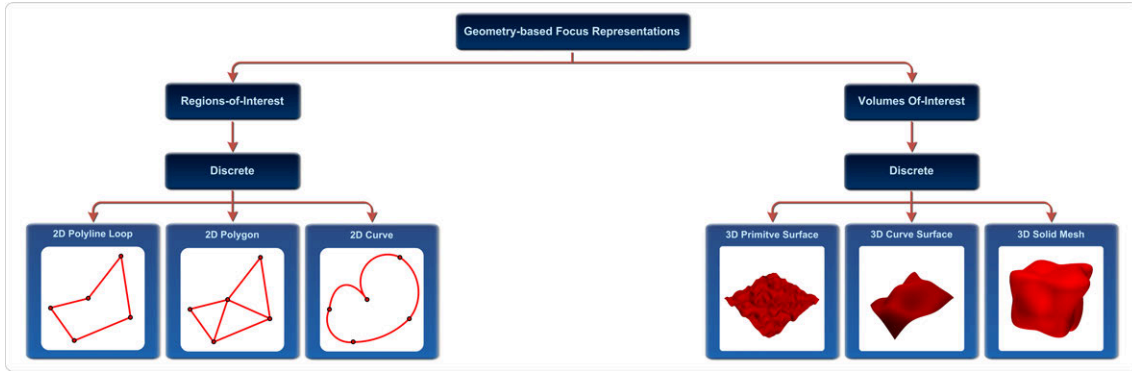
**Reference Coordinate Systems** In contrast to 2D GeoVE, the focus types describe above are applied within a 3D GeoVE with respect to a particular *reference coordinate system* [3] that determines the position and orientation of a focus type with respect to the virtual 3D environment, the orientation of the virtual camera, and the 2D canvas or projections plane. The rendering techniques in this thesis uses three different reference systems (Fig. 3.4).

The *world-space reference system* (WSRS) shown in Figure 3.4(a) is used to anchor a focus type with a fixed position and orientation in the 3D virtual environment that does not change with respect to the moving virtual camera. Further, the *camera-space reference system* (CSRS) shown in Figure 3.4(b) defines the position and orientation of the focus representation with respect to the configuration of the virtual camera. This means, the position, size, and orientation of a focus representation are adapted and transformed into the camera coordinate system prior to the application of a separation method. Furthermore, the *screen-space reference system* (SSRS) is an often used reference system: the position of the focus is relative to the screen (Fig. 3.4(c)).

## 3.4 Representation of Focus Types

After introducing different focus types for focus+context visualization of 3D GeoVE, this section describes different data types and data structures for their computer graphical representation on graphics hardware. These are categorized into *function-based*, *geometry-based*, and *image-based* representations. In general, there are two possibilities for the creation of the focus type representations:

**Manual Definition** The user explicitly models or define the shape of the focus. This can be done using tools, such as 3D modeling or 2D imaging software, or using interaction techniques integrated in the visualization framework. For example, a user draws the shape of the focus



**Figure 3.5:** Overview and classification of geometry-based representations of ROI and VOI focus types suitable for discrete separation methods only.

on 2D viewport, 3D terrain model, or by selecting objects-of-interest directly. Examples of explicitly created focus representations can be found in Chapter 5, 6 and 7.

**Automatic Definition** Focus representations can be automatically derived using *importance-driven algorithms* [263] based on the configuration of the virtual camera, the spatial or user context, as well as feature geometry of a 3D geovirtual environment. Examples for this representations can be found in Chapter 4 and 8.

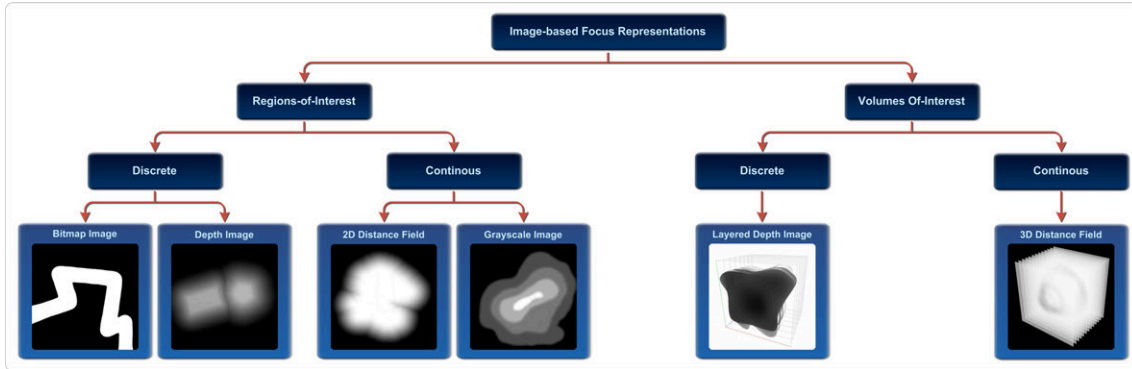
The remainder of this section briefly describes the three variants for focus representations w.r.t. the criteria above.

**Function-based Focus Representations** This category of focus representation describes POI, OOI, ROI, and VOI using a function that can be evaluated at the vertex, primitive, or fragment level of the programmable graphics pipeline by using shader programs. Function-based focus representations feature a low space complexity and a high numerical precision, since modern GPU offer 32bit IEEE floating point hardware. For the evaluation during the execution of a respective shader program, the required parameters are stored using constant buffers. Function-based focus representations can be distinguished into *degree-of-interest* and *implicit functions*, which are briefly described in the following.

*Degree-of-interest* (DOI) functions are used to discriminate data in focus from context information [88] by assigning a 1D DOI-value out of the unit interval to each of the data items (e.g., 1 represents in-focus and 0 is used for context information). Furnas presents a decomposition of a DOI function for fish-eye views:  $doi(X|Y) = API(X) - D(X, Y)$ , where the user's degree-of-interest in a point, feature, or object  $X$  is the *distance*  $D(x, y)$  between  $X$  and the current focus point  $Y$ . Here,  $API(X)$  is the global *a priori importance* of  $X$ . The distance may correspond to the Euclidean distance or to a semantic distance [212]. With respect to 3D geovirtual environments the focus  $Y$  can represent a feature (e.g., a building and point-of-interest) while  $X$  represents the virtual 3D camera. The on-screen highlighting approach of adaptive landmark scaling (Section 8.2) and the off-screen highlighting approaches (Section 8.3) use function-based focus representations.

*Implicit functions* [228] are another alternative to describe or encode focus types. An implicit function  $f(P) = 0$  for a point  $P \in \mathbb{R}^n$  as a form of functional representation is computational effective to evaluate, but hard to specify and control by a user. In computer graphics, *metaballs* are a classic concept to describe algebraic surfaces in a general approach [18] for  $n = 2, 3$ . They can efficiently represented on GPU using arrays of point and radii to describe regions-of-interest or volumes-of-interests.

**Geometry-based Focus Representations** In contrast for function-based or image-based representations, *geometry-based focus representations* rely on geometric primitives to specify boundaries (closed, non-self-intersecting lines, and surfaces) between focus or context. Figure 3.5 shows an overview of possible representation for ROI and VOI. For example, closed 2D poly-line loops, 2D curves, and tessellated variants such as 2D polygons can be used to describe a regions-of-interest.



**Figure 3.6:** Overview and classification of image-based representations of ROI and VOI focus types for discrete and continuous separation methods.

For volumes-of-interest, one can distinguish between representing *open* or *closed boundaries*. Closed boundaries are represented using solid (also known as "water-tight") 3D meshes. Open boundaries are described by 3D half-spaces that can be represented using 3D surfaces such as triangulated irregular networks (TINs) or parametric NURBS surfaces. The implementation of relief-clipping planes (Section 9.1) uses both, open as well as closed volumetric representations for rendering.

However, geometry-based focus representations are hard to evaluate in the respective stages of the programmable graphics pipeline. In the scope of this thesis they are mostly used to represent proxy objects (e.g., for image-based object highlighting in Section 8.2) or as intermediate representations, (e.g., 3D generalization lenses in Chapter 4), which are converted during runtime into image-based focus representation.

**Image-based Focus Representations** Image-based representations encode the boundaries for regions-of-interest and volumes-of-interest, or the respective degree-of-interest using raster images. Figure 3.6 shows an overview, a classification, and examples for image-based focus representations used in this thesis. They can be classified into four categories depending on its content and the reference space used by their associated separation methods. In general, one can distinguish between image-based representations that support *discrete* and *continuous* separation methods (Section 3.6).

*Bitmap and grey-scale images* encode the degree-of-interest values directly as texel value that can be accessed during rendering using texture sampling. Further, *depth-images and layered depth images* encode the distance of a fragment to the virtual camera. Further, layers of unique depth complexity [187] are often used in combination with a camera-space or screen-space reference system. The main principle is to separate a 3D space into two half spaces: *front* or *back*, w.r.t. a depth image, or *inside* or *out-side* w.r.t. a layered depth image. A major drawback for such focus representations is a possible lack of visual quality due to sampling artifacts. Image-based representations of regions-of-interest are used by techniques described in Chapter 5 and 6. Furthermore and in contrast to texture masks, the degree-of-interest can be computed from sampled distances encoded in 2D and 3D distance fields [101]. Using bi-linear sampling and interpolation of sampled distances, distance fields exhibit less sampling artifacts compared to 2D texture masks. Further, distance fields can be efficiently created during runtime using jump flooding algorithms [214]. and used for image-based object highlighting approaches presented in Section 8.2.

These image-based representations can be derived automatically from 2D or 3D implicit functions or geometric representations, using a preprocessing step in the mapping stage of the visualization pipeline. Further, image-based representations that encode the degree-of-interest directly can be easily created by a user using DCC tools (e.g., Photoshop and GIMP) and thus can be easily integrated into existing content creation pipelines.

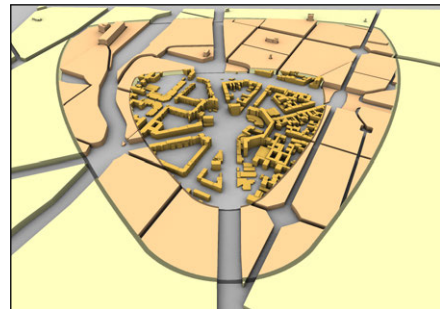
In addition to their straight-forward creation, management, and usage, image-based representations are characterized by the following two advantages: (1) they are able to encode various degrees of interests, or distances; (2) they require only a sampling operation for ROI representations and, in

case of layered depth-images, linear search (ray-marching) [TD08c] (Section 4.3) for evaluation. Despite their advantages, image-based representations exhibit a number of drawbacks. The visual quality depends on the resolution of the texture image and the precision of the data types used for internal storage. Thus, to obtain a high visual quality, a high resolution is required which yield high space-complexity. This complexity can be reduced using lossless compression algorithms [TD08b].

### 3.5 Visual Variants for Focus and Context

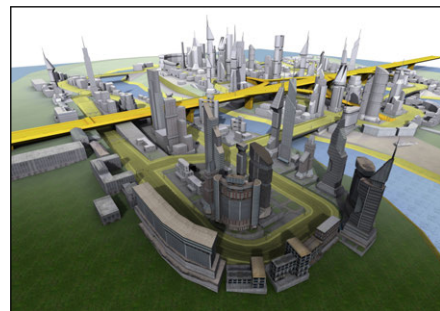
In addition to focus type and representation, a focus+context visualization technique can be classified w.r.t. the content that is presented in the respective context and focus region. Per definition, focus regions contain contents of high detail or high information density, while the contents of the context has a lower information density or is depicted in an abstracted way [188]. Generally, this can be achieved by creating *visual differences* or *visual variances*. These differences support a user to visually discriminate between focus and the context. This can be achieved by leveraging the humans pre-attentive image cognition using differences in appearance to guide to users focus [49] or by reducing the information density of the data set [76]. By shifting the cognitive load from the user to the application, abstract structures facilitates the assimilation and retrieval of information. With respect to the rendering process, there are basically two approaches for creating visual variants for focus and context: using *geometric variances* and *style variances*.

**Geometry Variances** Variants of the depicted input geometry can be achieved by varying the *scale* or the *density* of objects or features within a virtual 3D city or landscape model. With respect to 3D GeoVE, variations in scale can be distinguished into variations of *geometric scale* or *cartographic scale*. Variations in geometric scale can be created using uniform or non-uniform scaling [202] to facilitate the visibility of objects, e.g., by maintaining their screen size constant and thus reduce their occlusion at the same time. One example is the adaptive landmark scaling approach described in Section 8.2. For the variation of cartographic scale, a combination of different geometric representations for the same geolocation can be used. Such representations can be the results from level-of-details (LoD) as defined in CityGML [103] or level-of-abstraction (LoA) as the results from the generalization of buildings [139], building complexes [93], or street networks [133]. Figure 3.7 shows an example for combining different LoA using the concept of 3D generalization lenses presented in Chapter 4. Further, the data density can be modified by (1) adding data to the scene (e.g., labels) or (2) remove data to prevent visual clutter. Such changes in the data density requires altering the geometry representation of a 3D scene that is usually performed during mapping.

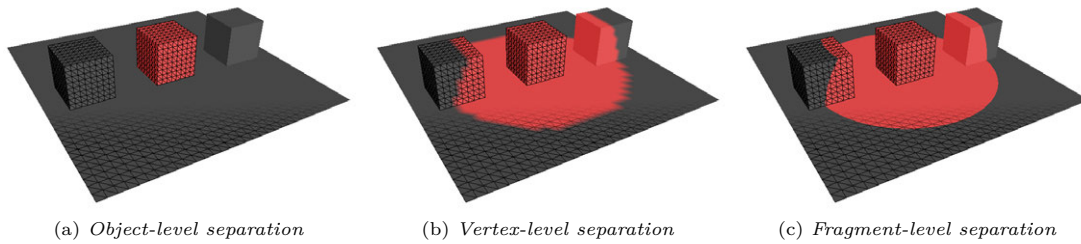


**Figure 3.7:** Example for geometry variances. Different level-of-abstraction are combined using 3D generalization lenses.

**Style Variances** *Style variances* alter the material or style in which a geometric presentation is rendered. This enables a user to visually discriminate between focus and context regions by maintaining structural features of a 3D scene simultaneously [49]. Figure 3.8 shows an example of a focus+context visualization using distinct visual styles for focus and context regions without modifying the scene geometry. These approaches do not modify or alter the geometric representation and are applied in the rendering stage of the visualization pipeline. Thus, style variances can be used if no LoAs of the virtual 3D city or landscape models are available or not required. The transitions between the rendering styles can be discrete or continuous. The latter



**Figure 3.8:** Application example for style variances. Facade texture are blended with a color-coded.



**Figure 3.9:** Comparison of object-level, vertex-level, and fragment-level separation that are used in combination with a discrete separation method.

can be achieved by mixing the color that represents the appearance of the focus with the color of the context. The difference in appearance can concern object material such as color (gray scale, opacity), texture (e.g., solid, stipples), lighting models (e.g., Phong [3], Gooch [99], or global illumination models [61]), as well as shading (e.g., flat shading, Gouraud shading). In general, variances of photo-realistic and various non-photorealistic rendering styles can be used [STD11]. Variations of material and texture can be computed in the rendering stage of the visualization pipeline (e.g., using post-processing approaches) or in the mapping stage by preprocessing of texture data [232].

### 3.6 Separating Focus from Context

For focus+context visualization it is necessary to decide what data is part of the focus or part of the context. In this thesis, the term *data* can refer to one of the following three representations: (1) an object of a 3D scene, i.e., an instance of a geometric representation of a discrete geo-feature (e.g., using triangulated meshed that can be encoded by a number of vertex buffers in the video memory); (2) a geometry primitive consisting of vertices representing parts of the geometric representation of an object; and (3) a fragment as result of the rasterization process of the primitives.

#### Separation Level and Rendering Pipeline

Based on the focus types and their representations, a particular method or algorithm is required to separate focus from context regions. Techniques for discriminating focus from context can be considered as key components for focus+context visualization. Therefore, it is necessary to decide if a point of a 3D scene belongs to a focus or the context, in order to apply visual variants and/or context.

In general, one can distinguish between two types of separation methods: *discrete separation methods* and *continuous separation methods*. Discrete separation methods are limited to hard transitions between focus and context. Continuous separation methods enable smooth transitions between focus and context regions. These methods can be performed per-object in the mapping stage of the visualization pipeline, or during the rendering process using shader programs within the programmable rendering pipeline.

In addition to the classification of separation methods with respect to the visual appearance of the transition between focus and context, they can be further distinguished between the implementation level in the interactive rendering pipeline (Section 1.3). Figure 3.9 shows a comparison between the following four separation methods:

**Object-Level Separation (OLS)** is applied per object or feature (Fig. 3.9(a)). It assumes geometric separation and uniqueness of feature instances within their computer graphical representation. If the feature geometry is optimized for rendering (e.g., using batching [269]) the separation has to be performed on the respective bounding volumes [3], thus the precision of the test depends on bounding volume representation. The landmark scaling technique in Section 8.2 is an example of this separation type.

**Vertex-Level Separation (VLS)** using *vertex shader programs* operates on single vertices (Fig. 3.9(b)). This method is fast but its precision depends on the vertex density of the scene

geometry. Since vertex shader cannot destroy geometry, vertex level separation methods are usually applied for culling purposes.

**Primitive-Level Separation (PLS)** using *geometry shader programs* operates on geometric primitives (also Fig. 3.9(b)). One advantage is that a geometry shader can omit the rasterization of primitives.

**Fragment-Level Separation (FLS)** using *fragment shader programs* represent a fast and precise method. Fragment shader are capable of discarding fragments as well as manipulating the transparency of a fragments. This enables discrete as well as continuous focus and context separation methods.

This work does not consider separation methods in the sub-sample domain as well as the programmable tessellation units. To summarize, OLS is only suitable for discrete separation methods (next section) and can be performed on the rendering client (CPU) or on the rendering server (GPU). Further, due to the lack of visual quality (Fig. 3.9(b)), VLS and PLS should only be used as optimization approach prior to FLS to reduce unnecessary separation computations at fragment level. To conclude, FLS should be preferred because of visual high quality for discrete as well as continuous separation methods.

The rendering of geometric variants depends on the focus type and its representation. For discrete focus types (e.g., for 3D generalization lenses), forward rendering and clipping is applied. For continuous types, (e.g., for multi-perspective views), color blending is applied [3]. The blending operation can be performed using the color buffer for forward rendering or in a post-processing pass for deferred rendering. These approaches require multi-pass rendering if the abstract geometric variants cannot be derived during rendering, i.e., they must be computed explicitly such as cell-based generalizations of virtual 3D city models. Using layered rendering, image-based representation for the different abstract representation for foci or context can be derived in a single rendering pass. An example for combining different geometric representations within a single rendering pass are 3D iso-contours (Section 5.4).

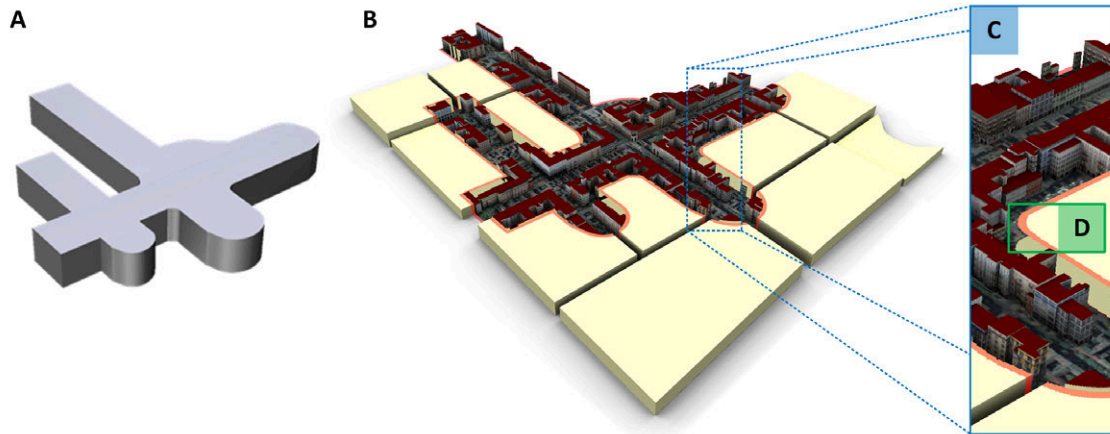
## Discrete Separation Methods

Discrete separation methods implement hard transitions between focus and context. Computer graphic algorithms for discrete separation using forward rendering are based on (1) multi-pass rendering and (2) the ability to omit the display of primitives or fragments during a rendering pass. A general approach for the multi-pass rendering for focus+context visualization using discrete separation methods is described in Section 4.5. For completeness, this section discusses discrete separation methods that are suitable for both, the fixed-function and the programmable rendering pipeline.

### Methods Using Fixed-Function Rendering Pipeline

Discrete separation methods for the fixed-function rendering pipeline use a combination of state sets [3] to omit the display of fragments associated either to focus or context. Despite approaches that use *constructive solid geometry* (CSG) [147], the approaches can be classified into (1) combinations of texture mapping and per-fragment operations, such as alpha, stencil, or depth tests [230] or (2) user-defined clipping planes.

**Alpha, Stencil and Depth Tests** Using a fixed function rendering pipeline, fragment discard can be implemented using alpha-testing [3] and texturing mechanisms. Here, texturing assigns an alpha value to each fragment. Geometric clipping using boolean textures [166] or bitmap images are examples for this strategy. To avoid the computation of the texture coordinate mapping in the mapping stage of the visualization pipeline and to enable movable filter, its is possible to apply projective texture mapping [81]. To fight alpha-testing artifacts, distance fields [101] can be used instead of Boolean textures. The same can be achieve using multi-pass rendering in combination with stencil and depth tests: (1) the geometric focus representation is rendered with activated writing to depth or stencil buffer respectively; (2) adjust depth or



**Figure 3.10:** Application example for a discrete separation method: A volumetric depth sprite (A) is used to represent a volume-of-interest for a 3D lens. Generalized clipping and multiple rendering passes are used to create a generalization lens visualization (B). This method introduces visual artifacts at the boundaries (D) between focus and context shown in the close-up (C).

stencil test to discard fragments that belong to the focus or context, e.g., that are closer to the camera or lie within a stencil region for rendering context geometry.

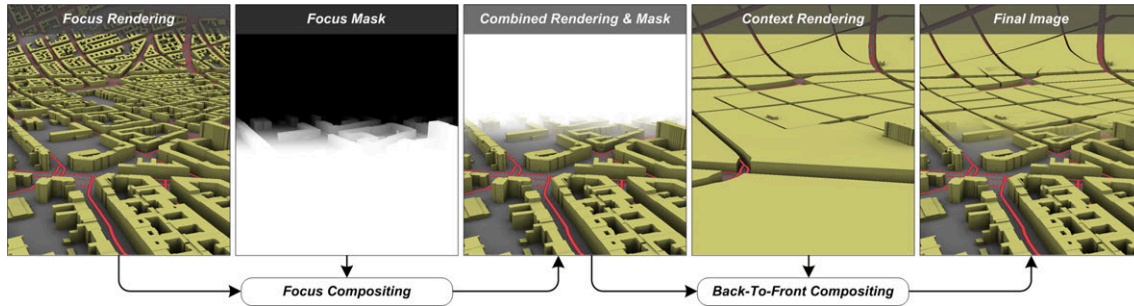
**Clipping Planes** This approach uses hardware-supported, user-defined clipping planes [3]. Here, a focus volume can be described using up to six clipping planes. This limits the shape of a VOI to cuboids, often rectangular cuboids, and enables only convex-shaped VOI. The user defines six planes in model space. During rasterization, fragments are discarded if they lay outside the volume that is described by the particular half spaces defined by the planes. This approach works for older hardware generation and is described in [261]. Thus, focus and context separation can be performed using two-pass rendering approach: the first rendering pass renders the focus geometry with enabled clipping planes and the second pass renders the context geometry using inverted parametrization of the clipping planes.

Since the increasing deprecation of fixed-function pipeline features, the rendering techniques in this thesis fully rely on concepts of the programmable graphics pipeline.

### Fragment Discard using Programmable Graphics Pipeline

*Fragment discard*, or *tex kill* is a basic functionality provided by programmable graphics pipeline. In contrast to fixed-function approaches, fragments are discarded prior to the per-fragment operations. This offers less complex implementations and reduces the number of state changes required for image synthesis. This section briefly describes applications for function-based, and image-based focus representations that are based on the functionality of fragment discard.

For function-based focus representations, 2D and 3D implicit functions are evaluated using the position of the fragment: based on the computational result, the fragment is discarded or not. For geometry-based focus representations such as 2D polyline loops, a "left-of-test" can be used to determine if a fragment is inside or outside the loop. This test can be efficiently implemented in real-time with a runtime complexity of  $O(n)$ . For image-based focus representations, such as bitmap or gray-scale images, the raster representation is sampled using texture coordinates. This texture coordinates can be given explicitly or can be computed procedurally at runtime by transforming the respective fragment coordinate into a given reference coordinate system (Section 3.3). After obtaining the sampled value, it is thresholded to decide if the fragment is discarded or not. Rendering techniques using this approach are described in Chapter 5 and Section 8.2. For image-based focus representations, such as depth-images and layered depth-images, the approach is similar to the one described above with the exceptions of the texture sampling and the procedural texture coordinate generation. The decision if a fragment is discarded or not is based on a Boolean parity computed for each fragment. This parity value is determined using a *volumetric parity test*



**Figure 3.11:** *Compositing to implement continuous separation methods for 3D multi-perspective views. The renderings of focus and context regions are combined using a focus mask represented by a grey-scale image that encodes where focus (white) or context (black) is visible.*

(VPT) described in Section 4.3. Application examples of this approaches are generalization lenses (Chapter 4 and relief clipping planes (Section 9.1).

### Continuous Separation Methods

The discrete separation methods described previously are suitable for implementing geometry variances but introduces hard transitions between focus and context (Fig. 3.10). This can result in a lack of structural coherence between the content of the focus and the context and yield difficulties in the spatial perception or cause problems in orientation and navigation [136]. Further, discrete separation between focus and context regions are not sufficient due to the following reasons: (1) it does not allow the visualization and communication of *fuzzy or uncertain boundaries* between focus and context; (2) dealing with image aesthetics it can be necessary to support *smooth transitions* between focus and context style variants, and (3) a method to reduce and disguise visual artifacts for high-quality visualization can be required.

There are basically two concepts to enable rendering techniques for supporting continuous focus and context separation: (1) *image-based compositing* [199] using single or multiple rendering passes and (2) *geometric morphing* (also geomorphing), which can be implemented in a single rendering pass.

**Image-based Compositing** For using image-based compositing, the degree-of-interest represented by a focus representation is mapped to a scalar value (e.g., alpha, opacity, or transparency), that controls how the images, which represent the respective visual variances for focus and context, are blended by color interpolated [249]. With respect to the type of visual variance used, there are basically two approaches for image-based compositing: *single-pass compositing* applicable for implementing style variances and *multi-pass compositing* required for implementing geometry variances:

**Single-pass Compositing** This type of compositing is performed on a per-fragment basis during a scene-rendering pass and does not require a post-processing pass. Thus, it is only suitable for style variances implemented by color transformations such as vignetting or de-saturation. It is applied for the interpolation of different terrain textures only (Section 7.2) and for texture variants derived at shader runtime (Chapter 5 and Section 8.2).

**Multi-pass Compositing** Figure 3.11 shows an overview of image-based compositing using multiple scene-rendering passes that are required because of the different geometric representations for focus and context. The compositing is performed using the frame-buffer or an additional post-processing pass by combing focus and context renderings using a focus mask. An application example for this type of compositing can be found in Section 7.4 and 8.2.

**3D Geometric Morphing** Morphing, or metamorphosis, aims to generate a smooth shape sequence which transforms a source shape into a target shape [274]. In computer graphics, geomorphing is the process of smoothly interpolating between 3D geometric models of different level-of-details or level-of-abstraction to lessen the effect known as "popping". With respect to 3D geovirtual environments,



Glander et al. introduce concepts for geometric morphing between different level-of-abstractions of generalized 3D city models [92]. Problems arising for the morphing of multiple buildings to building cells hinders the application of this technique. However, continuous separation methods can be applied to generalized 3D landscape models [GTD11] based on the idea of 3D iso-contours [GTD10]. Section 5.4 presents an application example that generates smooth transitions of stepped terrain using 2D distance fields for the image-based focus representation.

Despite the pleasing visual effect and the avoidance of popping artifacts, smooth transitions for geometry or style variances introduce also a drawback for such visualization: the resulting images meaningful information such as dead-value pixels [136] within the transition zone. This can lead to a lack of visual clarity. Thus, the size of the transition zone is a trade-off between the quality of visual appearance and the loss of information [PTD11].

### 3.7 Summary

This chapter introduces the underlying concepts and classifications for the focus+context rendering techniques described in this thesis. An overview of different focus types is given and their possible representations for GPU are discussed. Based on these representations, respective separation methods to discriminate between focus and context regions are presented and briefly compared. Table 3.1 classifies and compare the rendering techniques of Chapter 4 to 8 according to the presented categories.

To summarize, most of the rendering techniques rely on image-based focus representation (IFR) for volume-of-interest (VOI) and regions-of-interest (ROI) focus types. This originates from its simple handling and the possibility for implementing fast separation methods. For points-of-interest (POI) or objects-of-interest, the function-based representation (FFR) is used. Geometry-based focus representations (GFR) are only used as intermediate representation. Despite the exceptions of image-space distortion and highlighting, the rendering techniques are implemented using the forward rendering (FRP) and deferred rendering pipeline (DRP) in equal parts.

Due to the functionality of multiple rendering targets, most techniques can be implemented using a single scene rendering pass (SP), with the limitation that only style variants (SV) are supported. This makes them especially suitable for 3D geovirtual environments of high geometric complexity. However, if geometry variances (GV) are required to implement a focus+context visualization, multi-pass rendering (MPR) is required to combine different level-of-detail or level-of-abstraction. Thus, the number of rendering passes and consequently the rendering performance depends on the choice of visual variants for focus+context visualization.

**Table 3.1:** Classification of the rendering techniques for focus+context visualization of 3D geovirtual environments with respect to the following criteria: the focus type (*POI* – *Point-of-Interest*, *ROI* – *Region-of-Interest*, *VOI* – *Volume-of-Interest*), the computer graphic representation of focus types (*FPR* – *function-based focus representation*, *IFR* – *image-based focus representation*, *GFR* – *geometry-based focus representation*), the reference systems for the focus representation (*WSRS* – *world-space reference system*, *CSRS* – *camera-space reference system*, *SSRS* – *screen-space reference system*), the separation methods used (*DM* – *discrete separation method*, *CM* – *continuous separation method*), the visual variants (*GV* – *geometry variance*, *SV* – *style variance*), the respective separation level (*OLS* – *object level separation*, *PLS* – *primitive level separation*, *FLS* – *fragment level separation*), the used rendering paradigm (*FRP* – *forward rendering pipeline*, *DRP* – *deferred rendering pipeline*), and finally the number of rendering passes (*SPR* – *single-pass rendering*, *MPPR* – *multi-pass rendering*)

Chapter or Section	Visualization Technique	Focus Type	Focus Representation	Reference System	Separation Method	Visual Variant	Separation Level	Rendering Paradigm	Rendering Passes
4	3D Generalization Lenses	VOI	IFR	WSRS & CSRS	DM	GV	FLS	FRP	MPP
5	Dynamic Mapping of Raster Data	ROI	IFR	WSRS & CSRS	DM & CM	SV	FLS	FRP	SPP
6	Image-space Distortions	ROI	IFR	SSRS	DM & CM	SV	FLS	DRP	SPR
7	Multi-perspective Views	ROI	FRR	CSRS	CM	SV & GV	FLS & PLS	FRP & DRP	SPR & MPPR
8.2	Adaptive Landmark Scaling	POI & OOI	FRR	WSRS	DM	GV	OLS	FRP	SPR
8.2	Image-based Highlighting	ROI & OOI	IFR	SSRS	CM	SV	FLS	DRP	SPR
8.3	Off-screen Highlighting	POI & OOI	FRR	WSRS & SSRS	DM	SV	OLS	FRP	SPR

## Chapter 4

# 3D Generalization Lenses

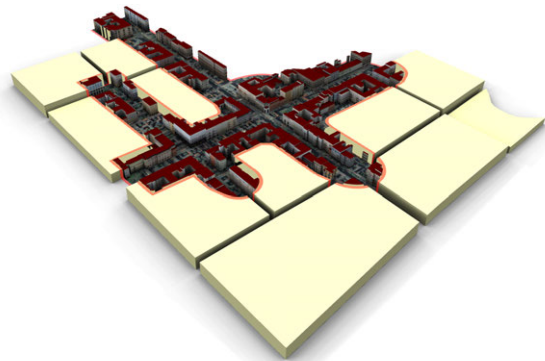
This chapter presents 3D generalization lenses, a visualization technique that combines different levels of structural abstraction of a virtual 3D city model. In an automatic preprocessing step, a generalized representation of a given city model is derived. At runtime, this representation is combined with a full-detail representation within a single view, based on one or more 3D lenses of arbitrary shape. Focus regions within lens volumes are shown in full detail while excluding less important details of the context area. The technique supports simultaneous use of multiple lenses associated with different abstraction levels, handle overlapping and nested lenses, and provides interactive lens modifications.

The remainder of this chapter is structured as follows. Section 4.2 introduces the fundamentals for generalizing virtual 3D landscape and city models. It briefly reviews the automatic generation of abstraction levels based on [93]. Section 4.3 present a concept and rendering techniques that enables the pixel precise clipping against arbitrary shaped and solid 3D meshes, which are required for the combination of different abstraction levels within a single output image. It is based on an image-based data structure that can be efficiently represented on GPUs and an algorithm capable of deciding whether a 3D point is inside or outside a 3D mesh. Based on that, Section 4.4 present the concept of 3D generalization lenses. Further, Section 4.5 briefly describes its implementation. Finally, Section 4.6 present usage scenarios and a discussion.

## 4.1 3D Lenses and Level-of-Abstraction

For a 3D geovirtual environment, 3D lenses can be used as a metaphor to control the focus+context visualization [261]. They direct the viewer's attention to a focus region, i.e., the model inside the lens, and simultaneously preserve the context information, i.e., the model outside of the lens. Focus+context visualization enables the user to access both, high-level context information and low-level details. Figure 4.1 shows a 3D lens containing a detailed version of a virtual 3D city model integrated into a generalized context. The presented rendering technique is a 3D instance of the *Magic Lens*<sup>TM</sup>[16, 261] metaphor. This metaphor is applied to focus+context visualization on structural scale [267] of a virtual 3D city model.

To use 3D lenses effectively, their interactive manipulation is required, enabling a user to position, rotate, and scale the lens dynamically. When dealing with complex models, the demand for interactivity leads to a number of challenges: First, to achieve a wide scope of possible lens configurations, the visualization should not be restricted to a single lens. Therefore, the visualization concept and technique have to deal with multiple, intersecting, overlapping and nested 3D lenses, which can be combined and moved in-



**Figure 4.1:** Application example for a single, arbitrarily shaped 3D lens. The focus preserves important details along a specified route. The context region is generalized.

dependently. The behavior in these cases should be configurable and consistent; Second, in a given visualization scenario, the core parts to be highlighted cannot be assumed to form only a simple shape. Therefore, a system should enable arbitrary lens shapes and users should be able to customize the shape of the lenses to fit the actual core parts appropriately; Third, The underlying rendering technique must perform fast enough to cope with large amounts of geometric primitives and texture data and to support rendering as well as lens manipulation in real-time. As a matter of principle, existing techniques that rely on image-based multi-pass rendering usually have performance problems when applied to complex 3D GeoVE with multiple 3D lenses. The presented interactive visualization technique supports multiple, dynamic, and arbitrarily shaped volumes-of-interest.

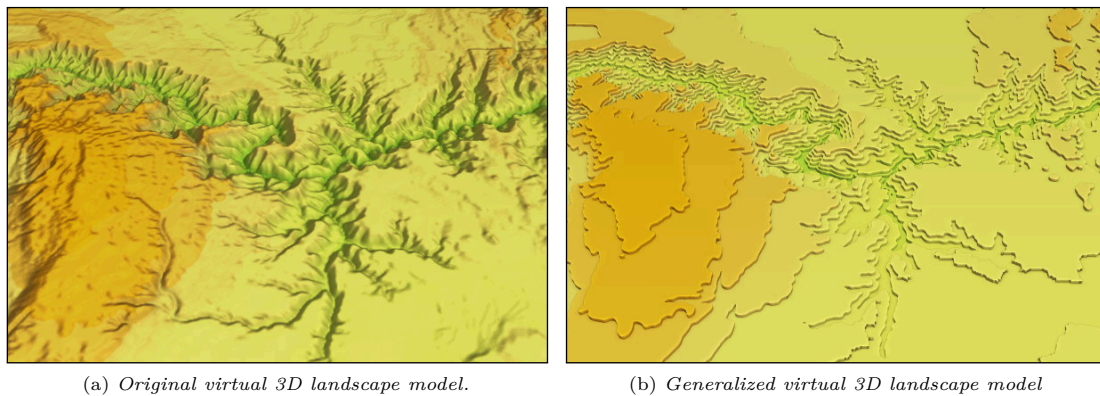
Today's large-scale virtual 3D city models are characterized by a large number of 3D objects of different types, manifold structures and hierarchies among them, and a high degree of visual detail [61]. Thus, they represent a high amount of information, e.g., encoded in facade textures, aerial photographs, building models, infrastructure models, and city furniture. This frequently leads to perceptual and cognitive problems for the user due to visual noise and information overload and, therefore, impairs tasks and usability, e.g., with respect to orientation and navigation in 3D geovirtual environments [94]. To facilitate comprehension, interaction, and exploration of a city model, the cartographic principle of generalization can be applied to create an abstract representation of the city model [94, 181]. Thus, the overall amount of information is reduced while the most important structures are preserved and highlighted. For large-scale city models, generalized representations at different levels of structural abstraction are required to achieve appropriate representations at different scales. The abstraction merges objects that are closely related into higher-level visualization units and preserves selected landmark objects. The generalization, however, does not provide means for local modifications of the abstraction level, e.g., to integrate more details along a specific route or within a specified area.

## 4.2 Generalization of Virtual 3D Landscape and City Models

In cartography, the term *generalization* means to abstract meaningful. It describes the process of reducing details of the depicted spatial information to a degree that is appropriate to scale, task and viewer [112]. A level-of-abstraction (LOA) [95] refers to the spatial and thematic granularity at which model contents is represented. It can be dependent on the current 3D camera settings, the current task or process, the user profile, or any other criteria from the underlying application domain. The model contents should be adapted to a given LOA on demand. Since generalization is a typically costly process that can hardly be performed in real-time, a number of LOA variants of a 3D GeoVE can be processed and stored. For the implementation of generalization techniques, level-of-detail (LOD) techniques from computer graphics [47] can be used, but LOD is fundamentally different to LOA since LOD focuses on an optimized computer graphics representation for real-time rendering purposes.

**Generalization of 3D Landscape Models** Virtual 3D landscape models serve as frameworks for representing geographic and thematic aspects of landscapes. They are based on 3D geodata including high-precision terrain and surface models (DTM, DSM), 3D building, site, and city models, 3D vegetation and water models, as well as aerial images to model the appearance of these components. This way, virtual 3D landscape models achieve a high degree of realism as required by a number of applications in virtual reality, design, and architecture.

A high level of geometric detail and photo-realistic appearance, however, is not an ultimate quality of a landscape model: There are numerous applications and systems requiring virtual 3D landscape models that simplify, aggregate, categorize, and abstract their contents to a specific coherent level-of-abstraction. These generalized virtual 3D landscape models facilitate, for example, understanding landscape form, conceptual design of landscapes, collaborative development of landscape models, and comparison of model variants. Generalization of 3D landscape models also enables their use and reuse for simulation and analysis processes and as computational tools, which commonly require a homogeneous spatial, thematic, and semantic resolution of the model



**Figure 4.2:** Example for a generalized variant of a virtual 3D landscape model of the Grand Canyon.

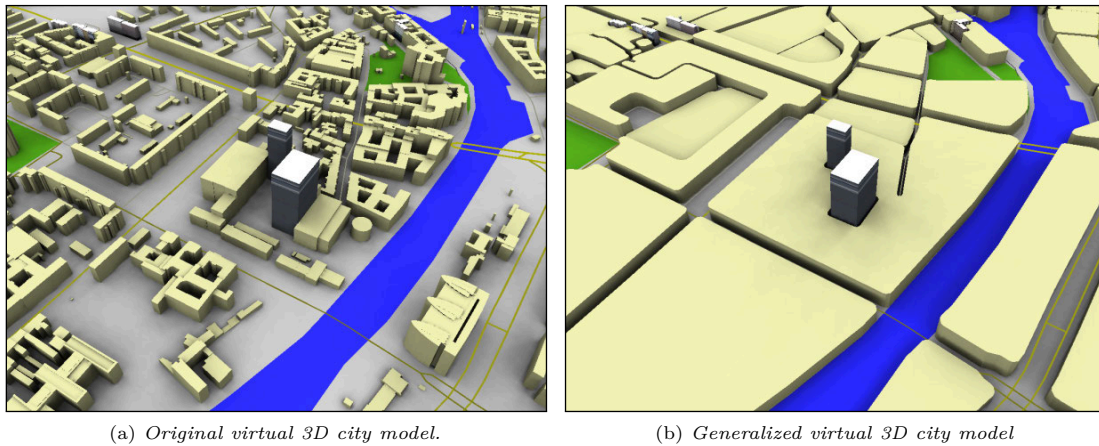
components and information density. Consequently, landscape generalization is a technique to cope with compactness, complexity, heterogeneity, and diversity of geodata that constitute today's digital landscape models.

Landscape generalization denotes a fundamental operation that transforms and presents landscape models at a given level-of-abstraction (LOA). The transformation and presentation techniques are based on a combined use of generalization operators including combination, reclassification, class selection, simplification, collapse, amalgamation, elimination, enhancement, displacement, enlargement, and typification [84]. Generalization operators can be applied (1) to the original geodata of a virtual 3D landscape model, i.e., it can be performed in the filtering stage of the visualization pipeline (Fig. 3.2), leading to a generalized primary landscape model. It can also be applied (2) to the mapped data, i.e., during the mapping stage of the visualization pipeline, leading to a generalized cartographic landscape model. Furthermore, it can be applied (3) during the rendering stage of visualization pipeline, leading to a generalized graphics representation. In practice, specific generalization techniques use a combination of these generalized models, in particular, if they have to provide adaptive, dynamic generalized models for interactive 3D systems, generalization operators for all three stages are required.

Glander et al. [GTD10] present a generalization technique that generates 3D stepped terrain models, which can be used in schematic terrain visualization to communicate relief structures of mountain landscapes through isocontours (Fig. 4.2). A set of typically equally spaced isovalues define height intervals that are highlighted by isocontours on the terrain model. The real-time generalization technique creates schematic visualizations from standard triangle-based terrain models exploiting graphics hardware. During rendering, the schematic visualization is obtained on-the-fly through (1) translating terrain vertices and (2) creating new step geometry. Hereby, all vertices of the input terrain are translated to their nearest isolevel by setting their height accordingly. This step results in planar triangles except at thresholds between two isovalues. Then, each input triangle crossing one or more thresholds is replaced by step geometry derived from the triangle's individual configuration. The step geometry is created by computing the intersection points per threshold and creating appropriate triangles. The new geometry adapts to the course of the isoline and thus reproduces it smoothly.

Further, an additional interpolation schema for single vertices facilitates smooth transitions between classical 3D terrain rendering and its stepped variant. A straightforward solution is to linearly blend between the vertices' original heights and their quantized height, using a control parameter. The parameter can be defined globally (constant), locally (additional layer) or view dependent. Thus, the technique allows flexible application of the effect including general activation/deactivation, highlighting regions-of-interest and distance-based styling.

**Generalization of 3D City Models** For the 2D case of a virtual city model, generalization approaches have been developed, e.g., based on agents [69] or based on least squares adjustment [234]. For 3D building generalization, existing approaches focus on the simplification of single



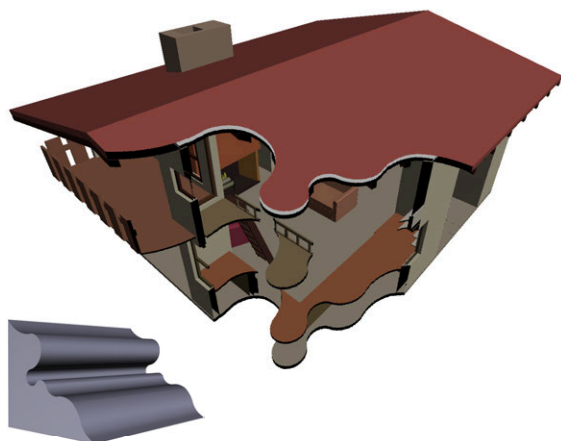
**Figure 4.3:** Example for a generalized variant of a virtual 3D city model.

buildings. For this, they remodel a building with a set of characteristic planes [139], or split it along characteristic planes into a Constructive-Solid-Geometry tree representation [248]. Morphological operations have also been applied to building generalization [86]. In [205], the generalization depends on line simplification performed on the projected walls. The approaches described above are limited to single objects and disregard aggregation of multiple objects. In previous work, a 3D generalization technique is introduced that performs aggregation of multiple objects [93, 94]. In this work, this technique is applied as a preprocessing step.

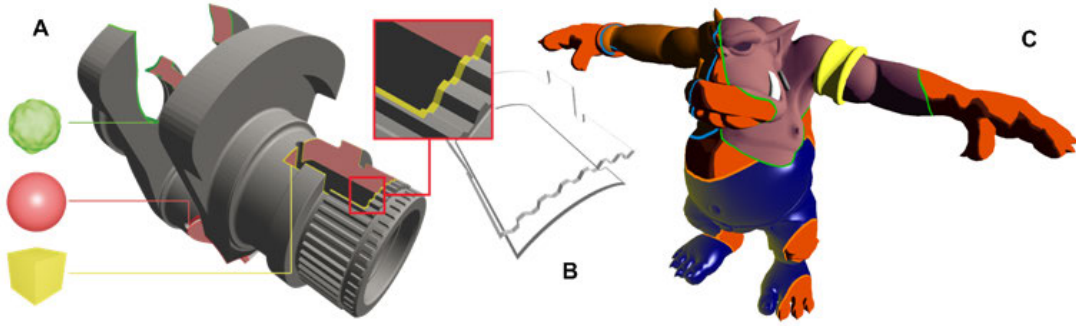
This generalization technique [95] generalizes a given virtual 3D landscape model according to a given hierarchical infrastructure network (e.g., hierarchical street network). That network defines the spatial clustering of geographic space by geographic cells (e.g., city blocks or districts). If a high hierarchy level is used as LOA, the resulting cells will be coarser. At the lowest level, the original model is unchanged. The technique takes as input the original (detailed) 3D model, and generates model variants for each hierarchy level. Thereby, it automatically aggregates single buildings into building blocks formed by the cells, i.e., the partitions of a given infrastructure network, handling separately green space and water areas. Local landmark buildings are detected, e.g., by their geometric properties, and preserved in their appearance (Fig. 4.3(b)).

### 4.3 Generalized Clipping

This section presents an approach for performing efficiently 3D point-in-volume tests for solid and arbitrary complex shapes. It classifies a 3D point as inside or outside of a solid specified by 3D polygonal geometry. This technique implements a basic functionality that offers a wide range of applications such as clipping, collision detection, interactive rendering of multiple 3D lenses as well as rendering using multiple styles (Fig. 4.4). For example, for rendering 3D volumetric lenses [261] it is necessary to decide which fragment, primitive, or vertex is inside the lens volume. The approach is based on an extension of layered depth images (LDI) [235] in combination with shader programs. An LDI contains layers of unique depth complexity and is represented by a 3D texture. The test algorithm transforms a 3D point into an LDI texture space and performs ray marching through the depth layers to determine its classification. Despite clipping and collision detection, the concept of



**Figure 4.4:** An example for using volumetric tests to perform pixel-precise clipping against a complex and non-convex 3D shape within a single rendering pass.



**Figure 4.5:** Application examples for *Volumetric Depth Sprites (VDS)* in combination with a *Volumetric Parity Test (VPT)*. Figure A shows pixel-precise clipping against three different VDS. The same approach can be used to extract cut-edges (Fig. B). Figure C shows rendering with hybrid styles. All images are rendered within a single pass.

VDS and the associated VPT has a number of applications. For example, it enables rendering with hybrid styles [132] per pixel and facilitates the generic usage of volumetric 3D lenses [261] without limitations concerning the volumes shape or the intersection of lenses.

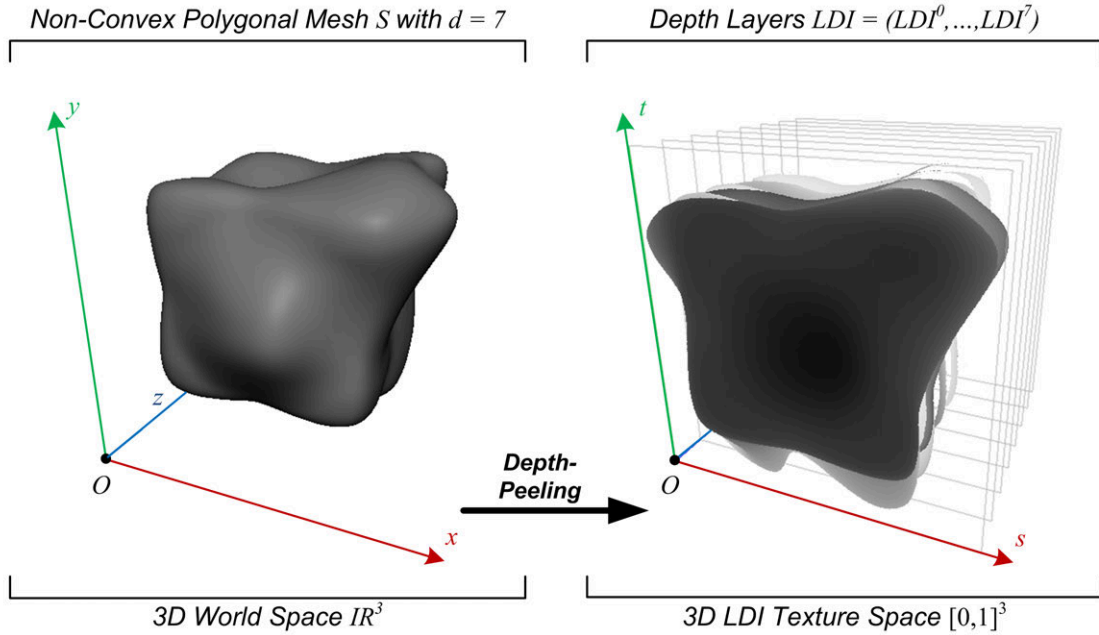
Current rendering techniques [218] enable such functionality using expensive image-based multi-pass rendering algorithms executed per rendering frame. Further, despite image-based *Constructive Solid Geometry (CSG)* [147], there is currently no approach that performs fast pixel-precise clipping against multiple arbitrary shaped meshes.

To enable such general functionality within a single rendering pass, a volumetric test algorithm is presented that targets real-time rendering applications in particular. This test is easy to implement and can be used in all programmable pipeline stages. For this purpose, an adequate data structure for the volumetric representation of static, solid meshes with arbitrary shapes is introduced that facilitates an efficient volumetric test. Since this representation can be created in preprocessing, no additional rendering passes at runtime are necessary. To perform the proposed test in real-time, the approach consists of two components: a data structure that is fully accelerated by graphics hardware and an algorithm that operates on instances of these. The algorithm is implemented in a shader program. The components are used in the following manner: first, an image-based representation of the solid shape are created that stores only its depth values along a viewing ray, which is aligned towards the negative  $z$ -axis. This representation is denoted as *Volumetric Depth Sprite (VDS)*. This step is performed during preprocessing and its result is stored by using high precision textures. Second, at shader runtime, a *Volumetric Parity Test (VPT)* is performed for each point. Therefore, a point is transformed into the specific VDS coordinate system and, then, is tested against all depth values stored in the particular VDS.

This approach requires solid shapes and, since this step is performed in preprocessing. Further, the shape's representation is assumed to be a static, non-animated polygonal mesh. Due to the recently established *Unified Shader Model*, which uses a consistent instruction set across all shader types, it became possible to apply the VPT in all programmable pipeline stages.

### Volumetric Depth Sprites

A Volumetric Depth Sprite (VDS) is an image-based representation of the shapes volume that stores its depth values along a viewing ray that is aligned towards the negative  $z$ -axis. A VDS extends the concept of LDIs [235] that contain layers of unique depth complexity. The idea of LDIs is presented in [235]. An LDI is a view of the scene from a single input camera view but with multiple pixels along each line of sight. The size of the representation grows only linearly with the observed depth complexity in the scene. Figure 4.6 shows an example of a VDS derived from a complex 3D shape. A VDS representation consists of the following components  $VDS = (P, LDI, d, w_i, h_i)$ . Where  $P \in \mathbb{R}^3$  denotes the position of the VDS in world space coordinates. The depth complexity of  $S$  is denoted as  $d \in \mathbb{N}_{\setminus\{0,1\}}$ . The layered depth image consist of  $d$  depth maps  $LDI = (LDI^0, \dots, LDI^{d-1})$ .



**Figure 4.6:** Example of an layered depth image representation of a non-convex polygonal mesh.  $S$  is depth-peeled into a number of slices, each containing depth maps of unique depth complexity.

The initial texture resolution of width and height is given by  $w_i, h_i \in \mathbb{N}$ . To obtain a depth value  $d_i \in [0, 1] \subset \mathbb{R}, 0 \leq i \leq d - 1$  in the  $i^{\text{th}}$ -depth layer for a 2D point  $(s, t) \in [0, w_i] \times [0, h_i]$ , the 3D texture is sampled in LDI texture space using the coordinates  $LDI_{(s,t)}^i = (s, t, i)$ .

There are various methods to create image-based representation from 3D geometry. Depth peeling can be used for order-independent transparency rendering, creating layered depth images, or layered distance maps. In [80] depth-peeling is performed using a second depth test implemented as shader program in combination with multi-pass rendering. This front-to-back method compares each fragment against the previous rendered layers of unique depth complexity. The runtime complexity to peel a shape of depth complexity  $n$  is  $O(n)$ . It is possible to re-use this layers by performing a render-to-volume technique [67]. To reduce the necessary rendering passes, multi-fragment depth peeling was introduced by [162]. It uses the functionality of multiple render targets in combination with fragment sort. The number of passes can be reduced to  $O(n = m)$ , where  $m$  is the number of available MRTs. The approach of dual depth peeling [10] reduces the runtime complexity to  $O(n/2) + 1$  by using a min-max depth buffer which peels two layers at a time, one layer from the front and one from the back. An extension of this method via bucket sort [163] uses MRTs. In [158] various memory layout options and optimizations are discussed. In this context, ray marching is a well known algorithm for interactive volume rendering [284]. A dynamic approach for slice-based object voxelization is presented in [73]. It can be applied in a single rendering pass but lacks accuracy.

The creation of a VDS is performed within a preprocessing step using multi-pass render-to-texture (RTT). Given a solid polygonal mesh  $S$ , the associated  $LDI$  is generated by performing the following steps: (1) Uniformly scale the shape to fit into the unit volume  $[0, 1]^3$ . A camera orientation  $O_{DP}$  and on orthogonal projection is set that covers this unit volume. The near and far clipping planes are adjusted accordingly. (2) Determine depth complexity  $d$  and create a 3D texture or 2D texture array with an initial resolution of  $w_i$ , height  $h_i$ , and depth  $d$ . The implementation uses a luminance texture format with a single 32Bit floating point channel. The texture is initialized with a depth of 1, which is further referred to as invalid depth value. (3) Perform depth-peeling [80] in combination with RTT. The solid  $S$  is peeled using linearized depth values using a  $W$ -buffer [155]. Listing 4.1 shows an OpenGL shading language (GLSL) implementation of the second depth test necessary for depth peeling.



```

1  uniform sampler3D LDI;          // layered depth image
2  uniform int      pass;          // number of current pass
3  varying float   linearDepth;   // linear depth interpolant
4
5  void main(void)
6  { // ignore first pass and fetch previous depth value at current pixel location
7    if( (pass > 0) && (linearDepth <= texelFetch3D(LDI, ivec3(gl_FragCoord.xy, pass-1),0).x) ){
8      discard; // depth value already peeled
9    }
10   gl_FragDepth = linearDepth; // write depth only
11 }

```

Listing 4.1: GLSL fragment shader implementation for depth peeling.

## Volumetric Parity Test

Real-time volumetric tests enable a multiple binary partition of a given arbitrary scene on vertex, primitive, and fragment level. They have a number of applications in real-time rendering and interactive visualization, such as pixel-precise clipping, collision detection, and rendering with hybrid styles [132]. This volumetric parity test (VPT) relies on an image-based representation of solid, arbitrarily shaped polygonal meshes (volumes). This representation is an extension of the concept of Layered Depth Images (LDI) [235]. Given a VDS, the Volumetric Parity Test (VPT) classifies a point  $V \in \mathbb{R}^3$  with respect to its position in relation to the shape's volume. It can either be inside or outside the volume. To model such test, a Boolean coordinate parity  $p_T \in \{0, 1\}$  is used. Before testing  $V$ , it must be transformed into the specific 3D LDI texture. For example, if  $V$  is a point in world space coordinates, the transformed coordinate  $T$  can be obtained by  $T = (T_s, T_t, T_r) = \mathbf{M} \cdot V$

The matrix  $\mathbf{M}$  represents the mapping of world space coordinates into LDI texture coordinates. It is defined by  $\mathbf{M} := \mathbf{T}(C) \cdot \mathbf{S} \cdot \mathbf{B} \cdot \mathbf{T}(-P)$ . Where  $\mathbf{B}$  is a rotated orthonormal base of the VDS.  $V$  is transformed into the LDI texture coordinate space ( $\mathbf{B} \cdot \mathbf{T}(-P)$ ), scaled by  $\mathbf{S}$ , and translated ( $\mathbf{T}(C)$ ) into the LDI origin  $C = (0.5, 0.5, 0.5)$  subsequently. A ray  $R = [(T_s, T_t, 0)(T_s, T_t, 1)]$  is constructed that marches through the depth layers  $LDI^i$  and compares  $T_r$  with the stored depth values  $d_i$ . Starting with an initial parity,  $p_T$  is swapped every time  $R$  crosses a layer of unique depth complexity (see Figure 4.7). This test can be formulated as  $p_T = VPT(T, LDI)$  so that:

$$VPT(T, LDI) = \begin{cases} 1, & \exists d_i \wedge \exists d_{i+1} : d_i \leq T_r \leq d_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (4.1)$$

$$d_i \in LDI^i_{(T_s, T_t)} \quad d_{i+1} \in LDI^{i+1}_{(T_s, T_t)}$$

The implementation of the ray-marching algorithm needs to iterate over the number of texture slices in the 3D texture. Therefore, it demands a shader model 3.0 compliant graphics hardware. Listing 4.2 shows the GLSL source code that implements the VPT. The performance of this algorithm depends on the number of VDS used, thus the number of samples the VPT has to perform. The presented volume test consists of less than 20 assembler instructions per executed loop. The time consumption for preprocessing a shape depends on its depth complexity and the desired texture dimension.

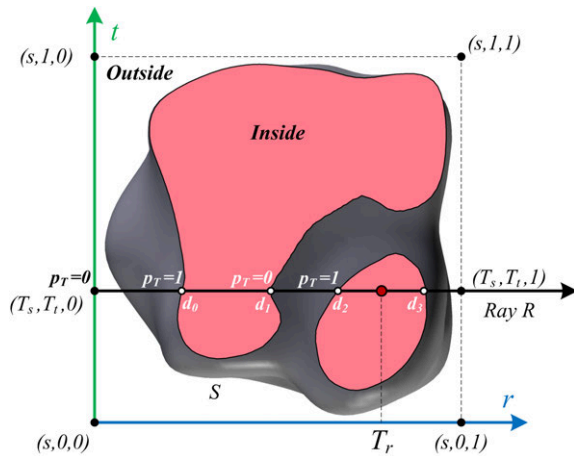


Figure 4.7: Ray marching through an LDI representation of the complex shape shown in Figure 4.6. A ray  $R$  intersects the depth layers  $LDI^i$  at four points and adjusts the rays parity  $p_T$  accordingly.

```

1  bool volumetricParityTest(in vec4    T,           // coordinate in LDI-space
2                          in sampler3D LDI,       // layered depth image LDIs
3                          in int      depth,      // depth complexity ds
4                          in bool     initParity){ // initial parity p
5      bool parity = initParity; // initial parity; true = outside
6      float offset = 1.0 / float(depth); // compute offset to address texture slices
7      for(float i = 0.0; i < float(depth); i++){ // for each texture layer do
8          if(T.r < texture3D(LDI, vec3(T.st, offset * i)).x) { // perform depth test
9              parity = !parity; // swap parity
10         }
11     }
12     return parity;
13 }

```

Listing 4.2: GLSL Implementation of the volumetric parity test.

## Efficient Representation of Volumetric Depth Sprites

One main drawback is the high memory consumption of  $M = w \cdot h \cdot d$  when representing an LDI as 3D texture. This is especially true for shapes with a high depth complexity  $d$ . Consequently, lowering the texture resolution  $w$  or  $h$  can result in a lack of precision when performing volumetric tests. Our goal is to determine optimal width  $w_i$ , height  $w_i$  and depth  $d$  to providing a high texture resolution simultaneously. A reduction of  $d$  implies a reduced number of depth-peeling passes which would speed up the dynamic creation of an LDI. To achieve this, three algorithms are proposed. The flowchart in Figure 4.8 describes the complete preprocessing including the following optimization algorithms. This section presents an efficient GPU representation in terms of minimizing the texture size and the necessary texture fetches. It presents three algorithms that facilitate the efficient creation and storage of an LDI. First, a method to determine an optimal viewpoint for the creation of an LDI for which the depth complexity is minimal. Further, A fast algorithm to determine the axis-aligned bounding box (AABB) of an LDI. It is used to crop unused coherent texture areas. Finally, a lossless compression algorithm that encodes the depth values of a 3D texture into a 2D texture and thereby achieves maximal texture utilization. The decompression can be performed using programmable hardware.

**Viewpoint Selection for LDI Creation** This step determines a camera orientation  $O_{DP}$  with a minimal depth complexity  $d_{min} \leq d_{max}$ . The approach is only effective for non-convex shapes with  $d_{max} > 2$ . A simple setup for viewpoint selection is used. The camera is placed onto a unit sphere that is constructed around the center of the shape  $S$ . The camera position is modified with respect to its horizontal and vertical position on the sphere. The pseudo code to determine the camera orientation  $O_{DP}$  for a shape  $S$ , an initial viewpoint  $O_C$ , and the number of horizontal  $s_H$  and vertical segments  $s_V$  is presented in Algorithm 1. For each segment on the sphere, the orientation

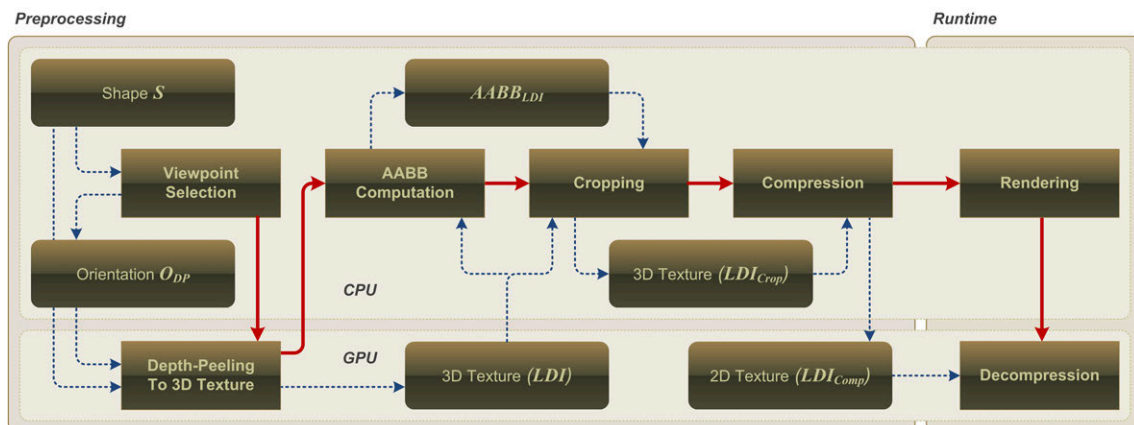


Figure 4.8: Conceptual overview and data flow between algorithms participated in the preprocessing of an input shape  $S$  into its compressed LDI representation. After proper viewpoint selection,  $S$  is depth-peeled, cropped, and compressed.

$O_C$  is computed by rotating the camera position around the  $x$  and  $y$ -axis with  $\theta = 360/s_H$  and  $\phi = 360/s_V$ . Subsequently, the depth complexity  $d$  and the coverage ratio  $c$  of the occupied and invalid texels are determined. The results of all segments are stored in a list that is sorted ascending by depth complexity afterwards. Under the preservation of this order, the coverage values  $c$  are sorted in a descending order to obtain the orientation with the minimal depth complexity  $d$  and the maximal coverage  $c$ . After  $O_{DP}$  is retrieved the VDS is created as described above.

---

**Algorithm 1** View-point Selection Algorithm
 

---

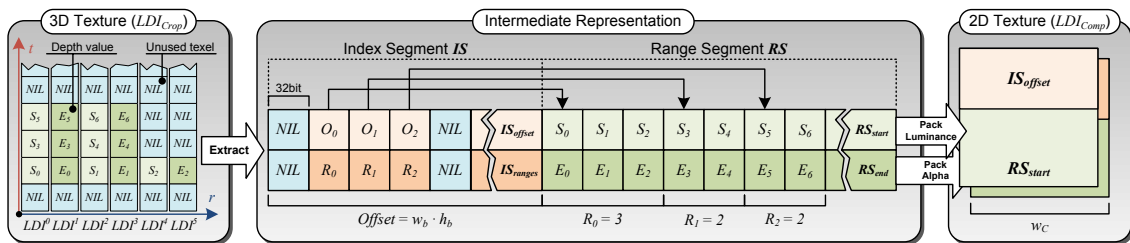
```

procedure orientation( $S, O_C, s_H, s_V$ )
1: [foreach segment on sphere]
2: for ( $h = 0 \rightarrow s_H$ ) do
3:   for ( $v = 0 \rightarrow s_V$ ) do
4:      $O_C \leftarrow$  adjustOrientation( $O_C, s_H, s_V$ ) [place virtual camera]
5:     [compute metrics]
6:      $d \leftarrow$  depthComplexity( $S, O_C$ )
7:      $o \leftarrow$  coverage( $S, O_C$ )
8:     append(list, ( $O_C, d, o$ )) [store result in list]
9:   end for
10: end for
11: [sort results]
12: sortDepthComplexityAscending(list)
13: sortCoverageDecending(list)
14:  $O_{DP} \leftarrow$  getOrientation(list, 0) [fetch first list item]
15: return  $O_{DP}$ 
  
```

---

**Bounding Box Computation and Cropping** Since hardware is not bounded to power-of-two texture dimensions [230], it is possible to optimize the texture storage on video memory by cropping the 3D texture to its 2D axis-aligned bounding box  $AABB_{LDI} = (x, y, w_b, h_b)$ , which includes all occupied (valid) texel in the LDI. This is particularly efficient if the shape has a main spatial extend along one of the two axis  $s$  or  $t$ , e.g., a torus. The AABB computation is performed on CPU using a scan-line algorithm [TD08b]. Prior to that, the LDI texture is fetched back into main memory. The algorithm needs to test every texel only once. After the  $AABB_{LDI}$  is computed, the 3D texture is cropped accordingly ( $LDI_{Crop}$ ).

**Lossless Compression of Volumetric Depth Sprites** Compressing the 3D texture representation of an LDI can decrease the amount of memory that sparsely occupied depth layers require on hardware. Compression can also increase the application performance by reducing the texture upload time and the number of texture samples: Due to the design of the VPT, the ray-marching algorithm (Fig. 4.7) has to take all depth layers into account to decide if a 3D point lies inside



**Figure 4.9:** Concept of compressing the 3D texture  $LDI_{Crop}$  into a 2D texture  $LDI_{Comp}$ . All depth values are extracted from the depth maps  $LDI^i$  and stored successively into a range segment  $RS$ . During this process an index segment  $IS$  is constructed. It stores an offset ( $O_V$ ) for a particular 2D coordinate ( $s, t$ ) that points to the start of the respective depth ranges within  $RS$  as well as the number ( $R_V$ ) of successive depth ranges  $DR_j$ . This intermediate representation is packed into a 2D texture  $LDI_{Comp}$ .

the volume or not. So, texture samples are retrieved for layers that may not contain any depth information. For sparse 3D textures, perfect spatial hashing [157] can be applied to loss less pack sparse data into a dense table. Since LDIs contain only depth values that describe a solid volume, a simpler alternative can be used for compression. Unlike existing compression algorithms for LDIs [68] our approach has not to deal with color information and exploits this specific property by storing only the structure of the LDI into a 2D texture. Consider a ray  $R$  as depicted in Figure 4.7. The depth values  $d_i, i = 0, \dots, d - 1$  at the intersection points of  $R$  and the  $LDI^i$  depth maps can be grouped into a number of depth ranges  $DR_j = (S_j, E_j)$ , with  $j = 0, \dots, d/2$ . The interval  $[S_j, E_j]$ , with  $S_j = d_{j \cdot 2}$  and  $E_j = d_{j \cdot 2 + 1}$  specifies the inside of the volume along  $R$ . The proposed compression algorithm consists of two phases: *extract* and *pack*. Figure 4.9 illustrates the process and involved entities.

The first phase extracts all available depth ranges and stores the values of the even depth layer into  $RS_{start} = (S_0, \dots, S_m)$  and the values of the odd layer into  $RS_{end} = (E_0, \dots, E_m)$  respectively. Simultaneously, an index segment  $IS$ , consisting of the vector  $IS_{offset} = (O_0, \dots, O_n)$  and  $IS_{ranges} = (R_0, \dots, R_n)$ , with  $n = w_b \cdot h_b$  is constructed.  $IS_{offset}$ , initialized with a zero offset, stores an offset into the  $RS$  for every 2D texel  $(s, t)$  in the first depth layer  $LDI^0$ .  $IS_{ranges}$  stores the number of depth ranges for the coordinate  $(s, t)$ . Therefore, a tuple  $I = (O_V, R_V)$  is denoted as an *index*, where  $R_V$  represents the number of depth ranges  $DR_i, i = 0, \dots, R_V$  for the ray coordinates  $(R_s, R_t)$ . The pseudo code displayed in Algorithm 2 provides details of the first phase of the compression algorithm, which computes the contents of the specific segments.

---

**Algorithm 2** Depth-Range Extraction Algorithm
 

---

**procedure**  $extract_{VDS}(VDS)$

```

1: list  $IS_{offset}, IS_{ranges}, RS_{start}, RS_{end}$ 
2: for  $s = 0 \rightarrow w$  do
3:   for  $t = 0 \rightarrow h$  do
4:     if  $LDI[s, t, 0] \neq NIL$  then [check if texel is set]
5:        $layers \leftarrow 0$  [iterate over depth]
6:       for  $r = 0 \rightarrow d$  do
7:          $depthValue \leftarrow LDI[s, t, r]$  [fetch depth value]
8:         if  $depthValue \neq NIL$  then
9:            $layers \leftarrow layers + 1$  [depth is valid]
10:          if  $r \% 2 = 1$  then
11:             $append(RS_{start}, depthValue)$  [set as start of range segment]
12:          else
13:             $append(RS_{end}, depthValue)$  [set as end of range segment]
14:          end if
15:        else
16:          break; [no more valid depth values]
17:        end if
18:      end for
19:       $IS_{offset}[t \cdot w + s] \leftarrow ||RS_{start}||$  [add start of range segment]
20:       $IS_{ranges}[t \cdot w + s] \leftarrow layers/2$  [add number of range segments]
21:    end if
22:  end for
23: end for

```

---

The second phase packs the  $IS$  and  $RS$  into a 2D luminance-alpha texture which is denoted as  $LDI_{Comp}$ . Therefore,  $IS_{offset}$  and  $RS_{start}$  are stored successively in the luminance channel and  $IS_{ranges}$  and  $RS_{end}$  in the alpha channel respectively. The texture resolution is given by  $w_c = h_c = \lceil \sqrt{|IS| + |RS|} \rceil$ . Due to the constraints of texture resolution, it is not possible to use 1D textures or texture arrays for our data structure. Often, the maximal texture size is limited

```

1  bool volumetricParityTestCompressed(in vec4      T,          // point in LDI texture space
2                                     in sampler2D  compLDI,     // compressed LDI
3                                     in vec4      bounds,      // crop bounds
4                                     in uvec2     size,        // resolution of uncompressed LDI
5                                     in bool     initParity){ // initial parity
6  bool parity = initParity; // initial parity; true = outside
7  if(testAABB2D(T.xy, bounds.xy, bounds.xy + bounds.zw)) // compensate cropping
8  {
9      T.xy = (T.xy - bounds.xy) / bounds.zw; // map cropped coordinates to [0,1]
10     uvec2 CTS = uvec2(textureSize2D(compLDI, 0)); // retrieve compressed texture size (CTS)
11     uvec2 UVC = uvec2(T.st * vec2(size)); // uncompressed vds coordinates (UVC)
12     unsigned int LUVVC = UVC.y * size.x + UVC.x; // convert to linearized UVC (LUVVC)
13     ivec2 ISC = ivec2(LUVVC % CTS.x, LUVVC / CTS.x); // coordinate of the index segment (ISC)
14     uvec2 IS = uvec2(texelFetch2D(compLDI, ISC, 0).ra); // sample from index segment IS
15     if(IS.x != 0u) { // depth ranges available ?
16         for(unsigned int i = 0u; i < IS.y; i++) {
17             // compute range sample coordinate (RSC)
18             ivec2 RSC = ivec2((IS.x + i) % CTS.x, (IS.x + i) / CTS.x);
19             vec2 DR = texelFetch2D(compLDI, RSC, 0).ra; // sample range depth range DR
20             if(T.z <= DR.x && T.z >= DR.y) { // perform parity test
21                 parity = !parity;
22                 break;
23             } } } }
24     return parity;
25 }

```

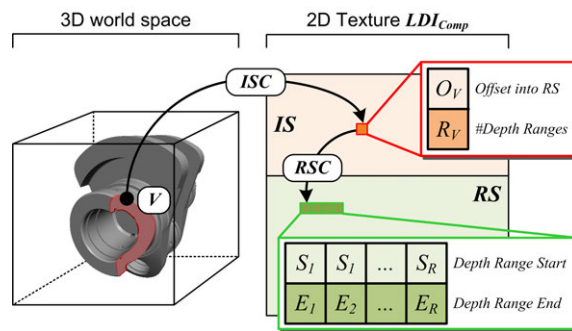
**Listing 4.3:** GLSL Implementation of the volumetric parity test using compressed LDI.

by the rendering hardware used. The limit for 1D textures for example (8192 pixel [230]) easily be exceeded. Therefore, the packaging uses 2D textures and need to introduce an additional un-mapping step for decompression.

**Decompression on GPU** The decompression of an  $LDI_{Comp}$  can be performed in all programmable hardware stages. Listing 4.3 shows a GLSL implementation of the VPT for compressed and cropped LDI. Figure 4.10 illustrates the process of fetching texels from a compressed LDI by unmapping texture coordinates. To obtain the coordinates  $ISC$  into the  $IS$ , a given point  $V$  is transformed using Equation 4.3. After scaling to counterbalance cropping,  $(T_s, T_t)$  is linearized with respect to the texture resolution of the original  $(w_i, h_i)$  and cropped LDI  $(w_b, h_b)$ . Then the linearized coordinate is re-interpreted with respect to  $w_c$ . After  $ISC$  is computed, the range segment offset  $O_V$  and the number of depth ranges  $R_V$  are retrieved. Now, the coordinate  $RSC$  into the range segment  $RS$  is determined and the depth ranges  $DR_j$  are successively sampled. The VPT of Equation 4.1 is implemented by swapping the input parity  $p_T$  if  $S_j < R_z < E_j$ .







#### Discussion of the Compression Algorithm

The test platform is a 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  $1600 \times 1200$  pixels. The test application does not utilize the second CPU core. The algorithms are tested with simple or complex, convex and non-convex input shapes of different geometric and depth complexity. Table 4.1 shows preprocessing results for different input shapes. The compression ratio is given by  $C_{ratio} = M_{comp}/M_{crop}$ . The proposed compression algorithm performs effectively for non-convex meshes with a high depth complexity. The presented approach is able to achieve compression ratios of 1:2-3, which is usual for lossless compression [68]. Compression should be avoided for symmetric convex meshes or meshes with  $d_{min} = 2$  since the compressed texture size  $M_{comp}$  is always larger than the cropped size  $M_{crop}$ . The runtime performance  $t_{view}$ ,  $t_{peel}$ ,  $t_{crop}$  and  $t_{comp}$  depends on the geometrical complexity of the input mesh



**Figure 4.10:** Coordinate transformation to access the depth ranges of a 3D point  $V$ .

**Table 4.1:** Performance results of our algorithms for input meshes of different depth complexity. The tests are performed with an initial texture resolution of  $w_i = h_i = 1024$  and  $s_H = s_V = 8$  segments for viewpoint selection. The time metric is seconds, the texture sizes are displayed in texel.

Shape	#Vertex	$d_{min}$	$d_{max}$	$t_{crop}$	$t_{comp}$	$t_{peel}$	$t_{view}$	$M_{crop}$	$M_{comp}$	$C_{ratio}$
	994	2	2	0.077	0.437	0.187	2.532	2,097,152	3,742,848	1.78
	768	2	6	0.078	0.297	0.187	3.085	2,097,152	3,011,058	1.44
	2,903	8	14	0.234	0.109	2.156	7.531	3,026,688	1,204,352	0.39
	6,146	6	12	0.203	0.219	25.766	7.172	5,683,200	3,317,888	0.58
	23,232	6	12	0.2	0.125	25.875	9.265	3,143,880	1,835,528	0.58
	34,344	10	18	0.313	0.078	37.078	12.187	3,358,720	1,089,288	0.32

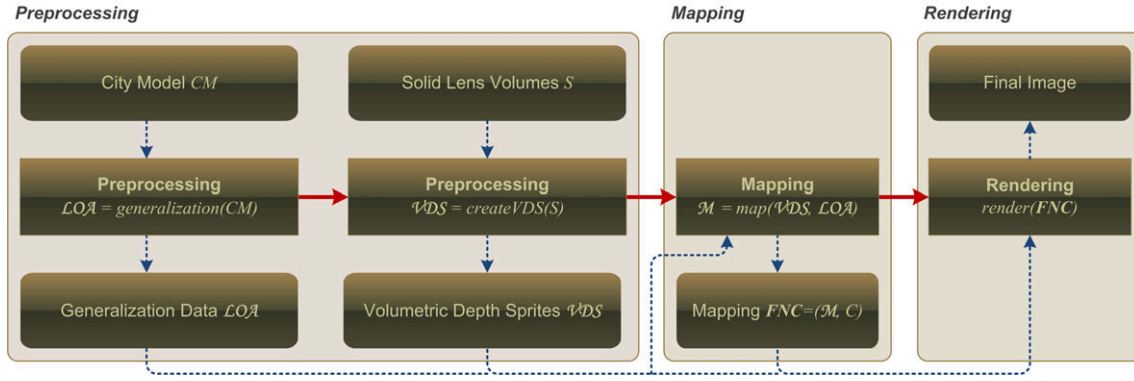
(#Vertex) and the initial resolution  $w_i, h_i$  of the LDI. The readback of the 3D texture from video memory to perform cropping and compression can become costly for large resolutions. To speed up the viewpoint selection, it can be performed with a lower resolution than the resolution required for the actual bounding representation.

The introduced depth range compression algorithm works only for LDIs that contain depth maps. If the user wants to incorporate additional per-text data, such as normal or color, perfect spatial hashing [157] can be used. Due to the regular step size for the alteration of the camera parameter, it cannot be guaranteed that an orientation with a minimal depth complexity can be found for every shape. The rendering performance depends on the number and depth complexity of the used LDIs, thus the number of samples the VPT has to perform, and the geometrical complexity of the rendered scene. Performance tests point out that using compressed VDS is slower than using uncompressed ones. This can be explained by the computation costs for the sampling coordinates of the depth ranges. Although, the computational complexity for the VPT reduces from  $O(d)$  to  $O(d/2)$  for compressed LDIs.

## 4.4 Concept of 3D Generalization Lenses

This generalization technique enables the seamless combination of different virtual 3D model variants of the same geographic area (e.g., LOA variants) within a single image using multiple rendering passes. It is based on a the general volumetric clipping technology introduced in the previous section, which enables pixel-precise clipping of an arbitrary polygonal scene representation against arbitrary 3D solid clip geometry. This polygonal clip geometry is converted into a layered depth image (LDI) during a preprocessing step. At runtime, the LDI combined with a volumetric depth test is used to determine if a fragment of the rasterized scene geometry is inside or outside the clip geometry represented by the LDI. This test can be implemented efficiently on modern consumer graphics hardware using shader technology. The data processing steps in required by the concept of 3D generalization lenses are be divided into a *preprocessing*, *mapping*, and *rendering* phase. It can be integrated into the visualization pipeline [267]. The flowchart in Figure 4.11 (next page) shows the respective pipeline stages, performed operations and participated data:

**Preprocessing Stage** This step prepares all necessary geometry for the rendering phase. This includes a generalization operation  $generalization(CM)$  of a give virtual 3D city model  $CM$  into a sequence of different levels of abstraction ( $\mathcal{LOA}$ ) using a cell-based generalization technique (Section 4.1). Further, the volumetric depth sprites  $\mathcal{VDS}$  for each lens volume  $S$  is derived using the algorithm described in Section (Section 4.3).



**Figure 4.11:** Visualization pipeline for the concept of 3D generalization lenses. A preprocessing stage prepares 3D generalization geometry and the 3D lens volumes for the rendering stage.

**Mapping Stage** Subsequently to preprocessing, the lens volumes and level-of-abstraction are associated  $map(VDS, LOA)$  to create an initial mapping  $\mathcal{M}$  between both. The final mapping ( $FNC$ ) contains also a distinguished context representation  $C$ , which is usually the most highest generalization level.

**Rendering Stage** During runtime, the mapping  $\mathcal{M}$  between levels of generalized geometry  $LOA$  and 3D volumes ( $VDS$ ) of the lenses can be specified and modified by the user. The focus+context mapping is rendered  $render(FNC)$  in subsequent passes by apply clipping against the respective lens volume. (Section 4.3). Every LOA geometry is rendered only once per frame.

The generalization technique creates a sequence of city model representations with increasing levels-of-abstraction. The generalization representation  $LOA_i$  of level  $i$  generalizes  $LOA_{i-1}$ . More specifically, one component from  $LOA_i$  aggregates a number of components from  $LOA_{i-1}$ . The technique focuses on aggregation, as it implicates the strongest abstraction compared to other generalization operators, e.g. simplification. The number of single objects is significantly decreased when turning to the next level-of-abstraction. For example, the city model used in this section contains 10,386 objects in  $LOA_0$ . It is reduced to 468 objects in  $LOA_1$  and to 66 objects in  $LOA_7$ . To obtain a homogeneous visualization, no facade textures are kept in the generalized representations. Textures are only used to provide depth cues using a lighting texture [61].

At runtime, the volumetric lens shapes are stored using respective VDS representations, and therefore, can be scaled, rotated and translated within the scene interactively. The system supports the generation of lens shapes from buffered 2D polygonal shapes and polylines. This enables the derivation of complex lens shapes directly from geo-referenced data. Further, lens shapes can also be modeled explicitly using 3D modeling software and imported via common interchange formats. The mapping  $\mathcal{M} = \{M_i | i = 0 \dots n\}$  between lens shapes  $VDS_i \in VDS$  and generalization levels  $LOA_i \in LOA$  can be described as a tuple  $M_i := (VDS_i, LOA_i)$ . Here,  $i$  denotes the priority of the mapping. This explicit order over the generalization levels is necessary to handle overlapping volumes correctly.  $VDS_i$  represents the focus volume whose contents is defined by the generalization level  $LOA_i$ . A particular generalization level represents the context  $C$ . The complete mapping is defined as  $FNC := (\mathcal{M}, C)$ .

## 4.5 Multi-pass Rendering

To render the mapping  $FNC$ , the technique performs multi-pass rendering with one pass per level-of-abstraction. Algorithm 3 shows the pseudo code for rendering the mapping described in the previous section. Starting with rendering the context geometry  $C$ , the algorithm performs pixel-precise clipping against all  $VDS_i$  within a single rendering pass. After that, the geometry of the generalization levels is rendered successively.

Starting with the highest generalization level  $n$ , the parity of the associated VDS is swapped (`setParity`), its associated geometry  $LOA$  is rasterized (`renderGeometry`), and finally the current

**Algorithm 3** Rendering Algorithm for 3D Generalization Lenses

---

```

procedure render(FNC)
1: for all  $M_i \in \mathcal{M}$  do
2:    $VDS \leftarrow VDS_i \in M_i$  [fetch mapping]
3:   setActive(VDS, true) [enable clipping]
4:   setParity(VDS, false) [clip inside of volume]
5: end for
6: renderGeometry(C) [render context geometry]
7: for  $i = n \rightarrow 0$  do
8:    $VDS \leftarrow VDS_i \in M_i$  [fetch VDS]
9:    $LOA \leftarrow LOA_i \in M_i$  [fetch geometry]
10:  if !culling(VDS) then
11:    setParity(VDS, true) [clip outside the volume]
12:    renderGeometry(LOA) [render geometry]
13:    setActive(VDS, false) [disable volume]
14:  end if
15: end for

```

---

*VDS* is turned off (setActive). This ensures that the geometry of higher abstraction levels does not interfere with the geometry of the lower ones. This algorithm is easy to implement and exploits the design of the VPT. To increase runtime performance, view-frustum culling (culling) is applied to the AABB of each 3D lens. If no corner vertex of the transformed bounding box of a *VDS* is inside the current view frustum, the active status of the respective *VDS* is set to false. The function renderGeometry uses the fragment shader shown in Listing 4.4. Basically, it fetches the GPU representation of the parametrization of *FNC* and performs a VPT on compressed LDI (volumetricParityTestCompressed) as described in Listing 4.3.

```

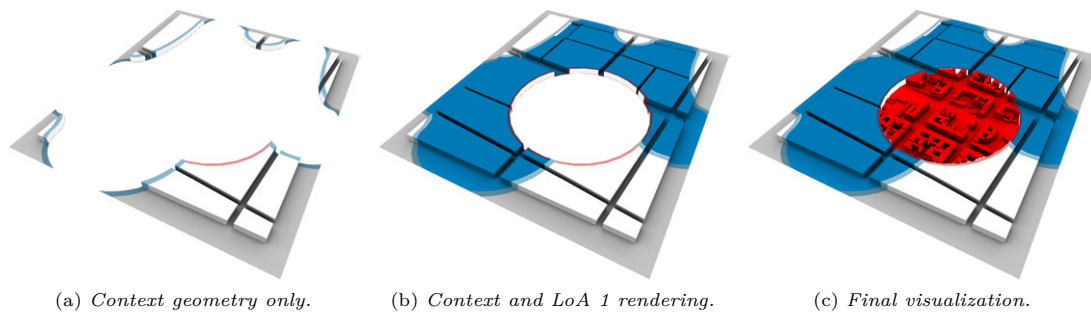
1  uniform mat4      vdsModelMatrix[MAX_VDS]; // VDS model matrix
2  uniform mat4      vdsConfigMatrix[MAX_VDS]; // VDS configuration
3  uniform sampler2D  vdsSampler[MAX_VDS];     // compressed VDS Sampler
4  varying vec4     vertex;                   // vertex coordinates to test
5
6  void main(void)
7  {
8      gl_FragColor = color;
9      for(int i = 0; i < MAX_VDS; i++) {
10         bool isOn = vdsConfigMatrix[i][0].x == 1.0; // check if VDS is active
11         vec4 T = vdsModelMatrix[i] * vertex; // transform into texture space
12         uvec2 size = textureSize(vdsSampler[i], 0); // retrieve LDI dimensions
13         bool parity = vdsConfigMatrix[i][0].y == 1; // set initial parity
14         if(isOn && volumetricParityTestCompressed( // perform volumetric test
15             T.xyz, vdsSampler[i], vdsConfigMatrix[i][1], size, parity)) {
16             discard; // discard fragment
17         } // endif
18     } // endfor
19     return;
20 }

```

**Listing 4.4:** GLSL Implementation for rendering 3D generalization lenses.

Figure 4.12 (next page) illustrates the above algorithm by showing the intermediate rendering results for an exemplary mapping  $FNC := ((VDS_0, LOA_0), (VDS_1, LOA_1)), C$ . Figure 4.12(a) shows the results of rendering only the context geometry *C* (white) with activated generalized clipping against all lens volumes  $VDS \in \mathcal{VDS}$ . The intermediate result of the second rendering pass (Fig. 4.12(b)) adds the geometry *LOA*<sub>1</sub> (blue) that is clipped against *VDS*<sub>0</sub>. Figure 4.12(c) shows the final visualization achieved by rendering the geometry of *LOA*<sub>0</sub> (red) clipped against *VDS*<sub>1</sub>.





**Figure 4.12:** Intermediate rendering results of the 3D generalization lenses rendering technique.

## 4.6 Usage Scenarios and Discussion

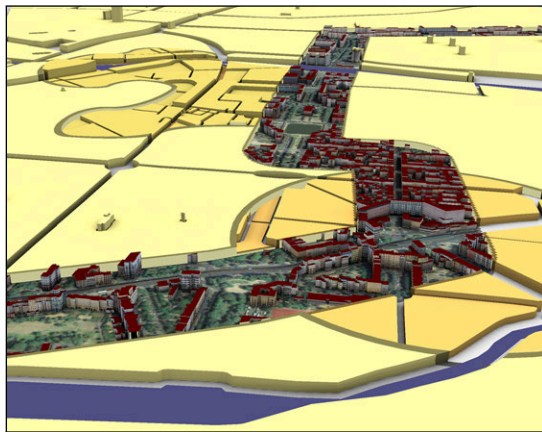
The presented technique supports the simultaneous use of multiple clipping volumes associated to different abstraction levels. It can handle overlapping and nested lenses, and provides their interactive modification: at runtime, the lens shapes can be scaled, rotated, and translated within the 3D scene. The system supports different methods for the lens shape creation. It enables the creation of complex lens shapes from geo-referenced data, such as buffered 2D polygonal shapes and polylines. Further, the lens shapes can also be modeled explicitly using 3D modeling software and imported using common interchange formats.

Figure 4.13 (next page) shows application examples for the interactive visualization of large scale virtual 3D city models. The input dataset comprises the inner city of Berlin with about 10,386 generically textured buildings on top of a digital terrain model. The approach was tested in different usage scenarios. Using only a single focus can be considered as the standard use case for the lens visualization technique. In this case the mapping is usually defined as:  $FNC = (\{(VDS_0, LOA_i)\}, LOA_j)$ , with  $j > i$ . Figure 4.1 shows an application example that emphasizes a single volume-of-interest. Using multiple foci implicate the handling of disjunctive, overlapping, and nested volumes-of-interest. Figure 4.13 shows examples of two overlapping regions-of-interest.

With respect to the lens interaction, the visualization technique supports two types of lenses. As described in [219] they can distinguished by the modification of the lens position during rendering with respect to the camera. The visualization technique supports both lens types simultaneously: (1) the position of a *scene lens* is independent from the user's orientation. The lens position can be fixed with respect to the 3D geovirtual environment or attached to a moving object in the scene (Fig. 4.13(a) and 4.13(b)); (2) a *camera lens* adapt its position with respect to the current user orientation. It can be used to assure that potential foci are always visible (Fig. 4.13(c) and 4.13(d)). This minimizes the effort for the user to interactively control the lenses.

To summarize, this chapter presents an interactive focus+context visualization technique that combines different levels-of-abstraction of a virtual 3D city model within a single image. The approach is based on an automated generalization algorithm for virtual 3D city models and pixel-precise clipping against multiple, arbitrarily shaped, polygonal meshes, which act as 3D lenses.

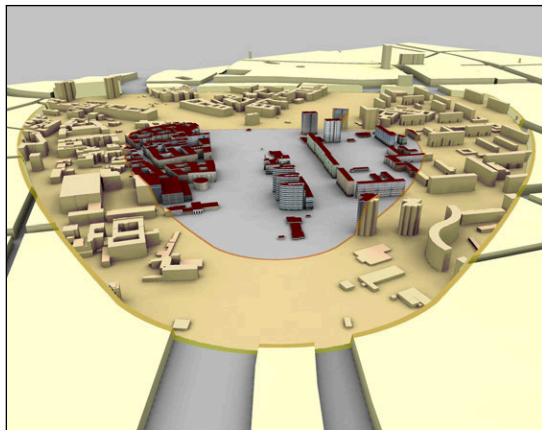
The main conceptual limitation refers to the one-to-one mapping between generalizations levels and lens volumes. Therefore, it is currently not possible to assign multiple volumes to a single generalization level. Further, the approach is limited by the memory consumptions of the *LOA* and *VDS*. The storage of high quality generalization levels, e.g., with precomputed lighting textures, exceeds easily the main memory size. Also, the 3D texture representation of the *VDS* can induce high GPU memory requirements. The main drawback concerns the rendering performance that depends on the number and depth complexity of the *VDS*. To reduce visual artifacts during clipping the visible backfaces are colored. Therefore it is not possible to apply back-face culling without causing visual artifacts.



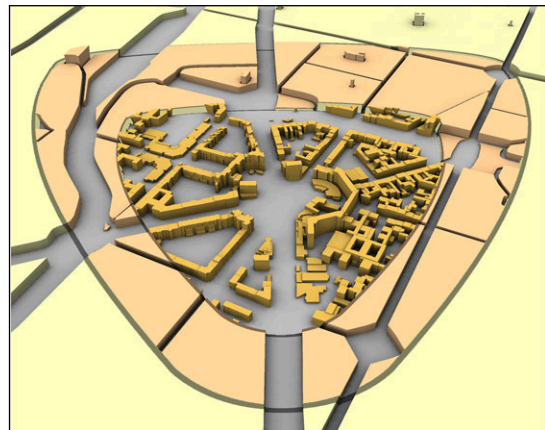
(a) Multiple overlapping scene lenses from a near ground perspective.



(b) Two overlapping scene lenses from a bird's-eye perspective.



(c) Two nested camera lenses from a near ground perspective.



(d) Two nested camera lenses from a bird's-eye perspective.

**Figure 4.13:** Examples for 3D generalization lenses in different usage scenarios.

## Chapter 5

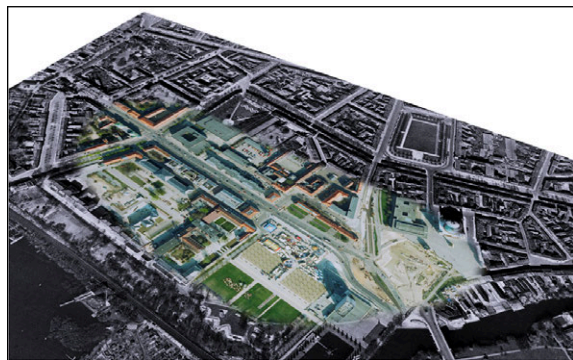
# Dynamic Mapping of Raster Data

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 this preprocessing step becomes a limiting factor. To counter this limitation, this chapter presents a concept for dynamically mapping multiple layers of raster data for interactive GeoVE. It represents a method of geographically oriented and earth surface related data processing for 2.5D representations. The implementation of the rendering technique is based on the concept of projective texture mapping and can be implemented efficiently using consumer graphics hardware. Further, this chapter demonstrates the flexibility of the presented technique using a number of typical application examples.

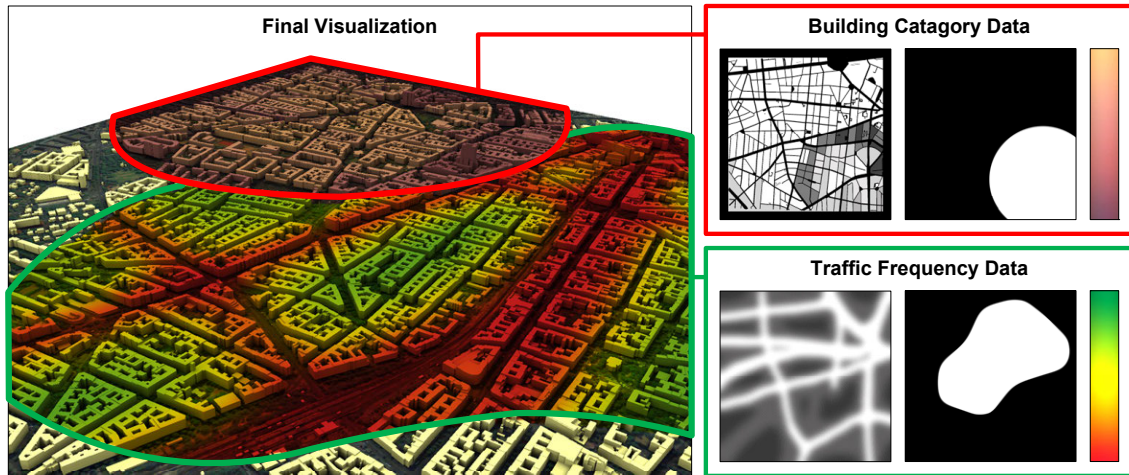
## 5.1 Decoupling Geometry and Texture Data

Today, the availability and acceptance of interactive 3D geovirtual environments 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 [17], e.g., coloring according to specific statistical data or focus+context visualization [46], typically rely on assigning information encoded in raster data [272] to the geometry of a 3D GeoVE. This assignment can be implemented using the concept of *texturing* [3], 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 models (BIM). For these models, the texture mapping is *static*, i.e., the necessary texture coordinates are computed or manually assigned in a preprocessing step. Consequently, design and appearance are inherent properties of these models and difficult to change and edit at runtime. 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 [164] use the concept of general-purpose computation on graphic processing units (GPGPU) to compute physical simulations of gases or fluids in real-time. These techniques use textures as main data type.



**Figure 5.1:** Examples of a 2D surface lenses combining two different layers containing temporal contents.



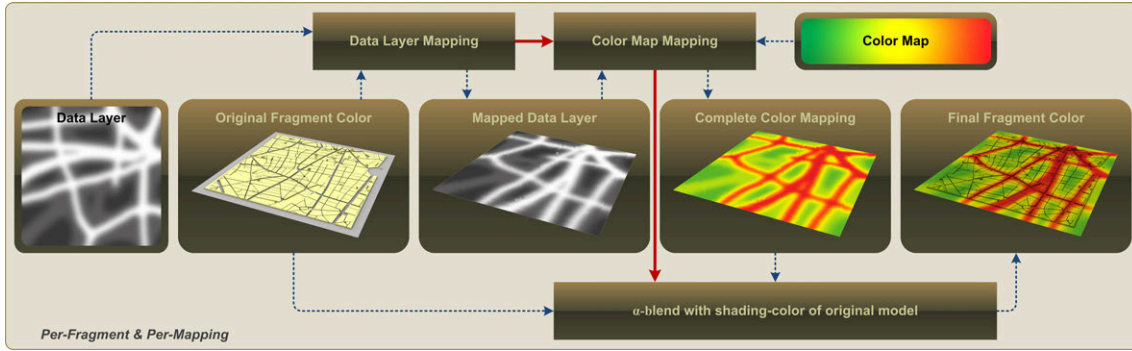
**Figure 5.2:** Application example for dynamically mapping of thematic raster data layer to 3D geovirtual environments. Building category data overlaps traffic frequency data.

The concept of *projective texture mapping* (PTM) [231] 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 APIs and hardware. The required *dynamic texture coordinate generation* (DTCG) is performed by applying a projection matrix to each vertex or fragment of a 3D scene. Debevec et al. use it for *view-dependent texture mapping* (VDTM) of oblique aerial images onto 3D geometry [52]. Due to the design of hardware APIs, existing implementations of PTM [81] are limited with respect to the number of projective textures, i.e., the number of simultaneously available texture units (varying between 8 and 32 on modern graphic cards). Further, PTM suffers from back-projections and artifacts caused by mismatches in sampling frequencies when perspective projection matrices are used. In existing geovisualization frameworks [143], 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* [96] 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. Subsequently, 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 [253] or to highlight salient features of a virtual scene. Bruckner & Gröller introduce style transfer functions for volume rendering [34]. The approach uses texture data for representing transfer functions and shader programs for their evaluation at runtime.

PTM [231] can perform DTCG, but existing implementations [81] are limited with respect to the number of mappings that can be applied in a single rendering pass. This limitation can be counterbalanced by using multiple rendering passes, but this decreases the performance especially for virtual 3D models of high geometrical complexity. Furthermore, the available concepts of texture mapping do not meet the requirements in terms of flexibility and scalability that are 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, the presented approach performs the dynamic mapping process by projecting the raster-data into the virtual scene. In combination with a



**Figure 5.3:** Illustration of the color mapping process for projective mappings. The data layer is mapped to the geovirtual 3D environment using projective texture mapping. The respective per-fragment data value is mapped to a specific color value and blended with the original fragment color.

framework for color transfer functions, the presented approach can be used to render most surface-related phenomena. The implementation provides a number of advantages: (1) the number of textures used are not limited by texture units provided by the rendering hardware (usually 16-32) but by the maximum number of 2D texture layers within a 3D texture or 2D texture array (usually 128 or more) and (2), it does not require multi-pass rendering for the image synthesis.

## 5.2 A General Concept for Projective Mappings

Together with a concept of color transfer functions that are specific to 3D GeoVE, this chapter presents a combination of projective texture mapping and texture bombing to overcome the limitations of the respective approaches. It 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 world-coordinates and the plane normal vector  $N_R \in \mathbb{D} = [0, 1]^4 \subset \mathbb{R}^4$ . Given this, the concept comprises three main components:

**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  $\mathcal{L} = (L_0, \dots, L_m)$ . The contents of these layers can further be mapped to final colors using 1D color look-up tables  $\mathcal{C} = (C_0, \dots, C_k)$ .

**Texture Coordinate Generation:** This component computes the texture coordinates for each scene point using projection matrices that are derived from a parametrization described below. The step assumes that an application already performed the geo-referencing for each parametrization, and thus, provides world-space coordinates for rendering.

**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 a color transfer function.

Figure 5.3 shows an overview how these components interact during the real-time image synthesis. A data layer is mapped to the virtual 3D environment using projective texture mapping. The respective per-fragment data value is subsequently mapped to a specific color value and blended with the original fragment color to obtain the final color.

**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  $\mathbb{D}$ .

**Mapping Parametrization** To enable a compact parametrization of PTM, the structure of a *projective mapping* (PM) is represented 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  $\mathbf{M}_T$ . Here,  $CM$  represents the parameter set for the color transfer function described in the next section.

A set of  $n$  projective mappings is denoted as  $\mathcal{PM} = \{PM_i | i = 0, \dots, n - 1\}$ . The explicit order is necessary for the correct compositing of the transfer function results during the evaluation process of each PM (Section 5.3). Given the above parametrization, instances of  $\mathbf{M}_O$  and  $\mathbf{M}_P$  are derived as follows:

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

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

**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  $CM = (f_{B_S}, f_{B_D}, f_{C_S}, f_{C_D}, \mathbf{M}_C, f_\alpha)$ . For a projective mapping  $PM$ , the source color  $C_S = f_{C_S}(S)$  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 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:  $C_O = f_B(C_{B_S}, C_S, C_{B_D}, C_D)$ . Here,  $f_B$  is the standard color blending operation [3], based on the blend functions for source color  $C_{B_S} = f_{B_S}(C_S, C_D)$  and destination color  $C_{B_D} = f_{B_D}(C_S, C_D)$ . Thus, standard combination modes, such as color replacement, multiplication, or blending, are supported.

### 5.3 Real-time Rendering of Projective Mappings

The prototypical implementation uses OpenGL [230] with the support of shader programs [144]. It 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 parametrization of  $\mathcal{PM}$  into suitable GPU data structures.

**GPU Data-Structures** To enable an efficient and scalable implementation, *2D texture arrays* are used 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 a 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, *texture buffers* are used to store the settings of  $\mathcal{PM}$ . Our implementation transfers the scalar and vector parameters of each active  $PM$  (Section 5.3) 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). 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.

**Per-Fragment Mapping Evaluation** 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 Algorithm 4 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), 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).

---

**Algorithm 4** Per-Fragment Mapping Evaluation
 

---

```

procedure evaluate( $P, \mathcal{PM}, \mathcal{L}, \mathcal{C}$ )
1:  $C_O \leftarrow \text{sceneColor}()$  [fetch color resulting from texturing and shading]
2: for all  $PM_i \in \mathcal{PM}$  do
3:    $PM \leftarrow PM_i \in \mathcal{PM}$  [fetch projective mapping]
4:    $T \leftarrow T_i \in PM$  [fetch projection matrix]
5:    $S = T \cdot P$  [compute texture coordinates]
6:   if testAABB2D( $S$ ) then
7:      $L \leftarrow L_i \in \mathcal{L}$  [fetch data layer]
8:      $C \leftarrow C_i \in \mathcal{C}$  [fetch color look-up table]
9:      $M_C \leftarrow M_{C_i} \in CM_i$  [fetch color mapping]
10:    [compute blending]
11:     $C_S = f_{\alpha_i}(S, f_{C_S}(S, L, C))$ 
12:     $C_D \leftarrow f_{C_S}(C_O, \mathcal{L}, \mathcal{C})$ 
13:     $C_{B_C} = f_{B_S}(C_S, C_D)$ 
14:     $C_{B_D} = f_{B_D}(C_S, C_D)$ 
15:     $C_O = f_B(C_{B_S}, C_S, C_{B_D}, C_D)$ 
16:   end if
17: end for
18: setFragmentColor( $C_O$ ) [set output color]

```

---

**Rendering Optimizations** Iteration and sampling are costly per-fragment operations, especially if performed for a high number of projective mappings. To increase the rendering performance, two CPU-based culling techniques are applied to each mapping  $PM$ , before it is encoded into the GPU data structure. At first, view-frustum culling is used to disable the mapping, if its respective bounding area does not intersect the view frustum of the virtual camera. Subsequently to this test, a screen-space error metric is applied. 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

```

1  uniform mat4      configurations[MAX_MAPPINGS]; // projection configuration
2  uniform mat4      projector[MAX_MAPPINGS];    // projection matrices
3  uniform mat4      viewInverse;               // view-inverse matrix
4  uniform sampler2DArray dTArray;              // data texture array
5  uniform sampler1DArray cmTArray;             // colormap texture array
6  uniform int       mappings;                  // number of projective mappings
7  varying vec4      eyeSpacePosition;         // position of fragment in eyespace
8
9  // implementations of blend equations
10 vec4 blendFunction(in int modeSRC, in int modeDST, in vec4 SRC, in vec4 DST);
11 // axis-aligned bounding-box test
12 bool testAABB2D(in vec2 v, in vec2 b1, in vec2 b2);
13 // computation of color appearance
14 vec4 layerAppearance(in vec4 sceneColor, in mat4 config, in vec2 coords,
15                     in sampler2DArray data, in sampler1DArray colorMap);
16 // computation of mask appearance
17 vec4 layerAlphaAppearance(in vec4 color, in mat4 config, in vec2 coords,
18                           in sampler2DArray data, in sampler1DArray colorMap);
19
20 void main(void) {
21     vec4 result = gl_Color;
22     vec4 sceneColor = result;
23     for(int i = mappings-1; i >= 0; i--){ // evaluate projective mappings
24         mat4 config = configurations[i]; // fetch current config matrix
25         // compute sampling coordinate
26         vec2 coords = (projector[i] * (viewInverse * eyeSpacePosition)).xy;
27         // test for 2D bounds...fetch only valid tex coords
28         bool bbTest = bool(configuration[0][0]);
29         if(!bbTest || testAABB2D(coords, vec2(0.0), vec2(1.0))){
30             // compute layer color
31             vec4 appearance = layerAppearance(sceneColor, config, coords, dTArray, cmTArray);
32             // compute layer alpha value
33             appearance = layerAlphaAppearance(appearance, config, coords, dTArray, cmTArray);
34             // fetch blend functions and perform blending
35             result = blendFunction(int(config[3][1]), int(config[3][2]), appearance, result);
36         } // endif
37     } // endfor
38     gl_FragColor = result;
39 }

```

**Listing 5.1:** GLSL fragment shader implementation for the dynamic mapping process.

of the mapping, a test function (`testAABB2D` in Alg. 4, Line 8) verifies, 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.

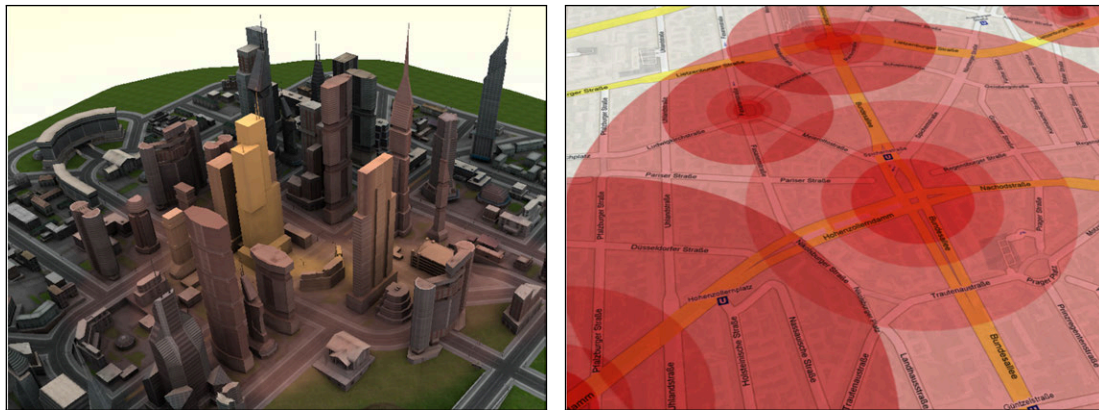
**Performance Evaluation** The test platform is an NVIDIA GeForce 8800 GTS with 640 MB video memory and Athlon 64 X2 Dual Core 4200+ with 2.21 GHz and 2 GB of main memory at a viewport resolution of  $1600 \times 1200$  pixels. The test application does not utilize the second CPU core. Table 5.1 provides the performance measurements of the examples in Section 5.4 that can

**Table 5.1:** Performance evaluation for various projective mappings (in frames-per-second).

Fig.	#Vertex	#PM	$PM_{off}$	$PM_{on}$	$\Delta(\%)$
1	1,040,503	3	10.41	9.87	5.19
3.A	41,032	4	52.88	51.20	3.18
3.B	61,756	27	156.10	27.94	82.10
3.C	6	4	1063.83	236.97	77.72
3.E	37,404	16	206.21	45.70	77.84
3.F	6	6	1075.27	110.54	89.72

be rendered within a single rendering pass. The performance evaluation showed that the 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 functions is the main computational bottleneck.





(a) Propagation data projected onto a virtual 3D city model

(b) Rendering of multiple, overlapping, non-uniform transmitter ranges.

Figure 5.4: Application examples for the projective mapping of spatial-temporal raster data.

## 5.4 Application Examples

The concept of dynamic mapping of raster data enables a hierarchy-based combination of different mappings and can be applied to render different standard visualizations. An area of application is the mapping of dynamic spatial-temporal data, e.g., crime data, noise data and environmental pollution data (Fig. 5.4(a)), transmitter coverage, or fire-, explosion-, and flood zones. It can further be used to visualize building heights, isolines, and the positions of moving objects. By changing the color transfer functions of  $CM$  accordingly, it is possible to communicate the overlapping of scalar values (Fig. 5.4(b)). This section discusses three applications examples for focus+context visualization of 3D GeoVE and their respective parametrization. The presented approaches can be used to implement lens-based focus+context visualization techniques such as 2D surface lenses as well as highlighting. Interactive movable filters [241] are supported by adjusting the projector matrix accordingly, i.e., by using direct manipulation techniques.

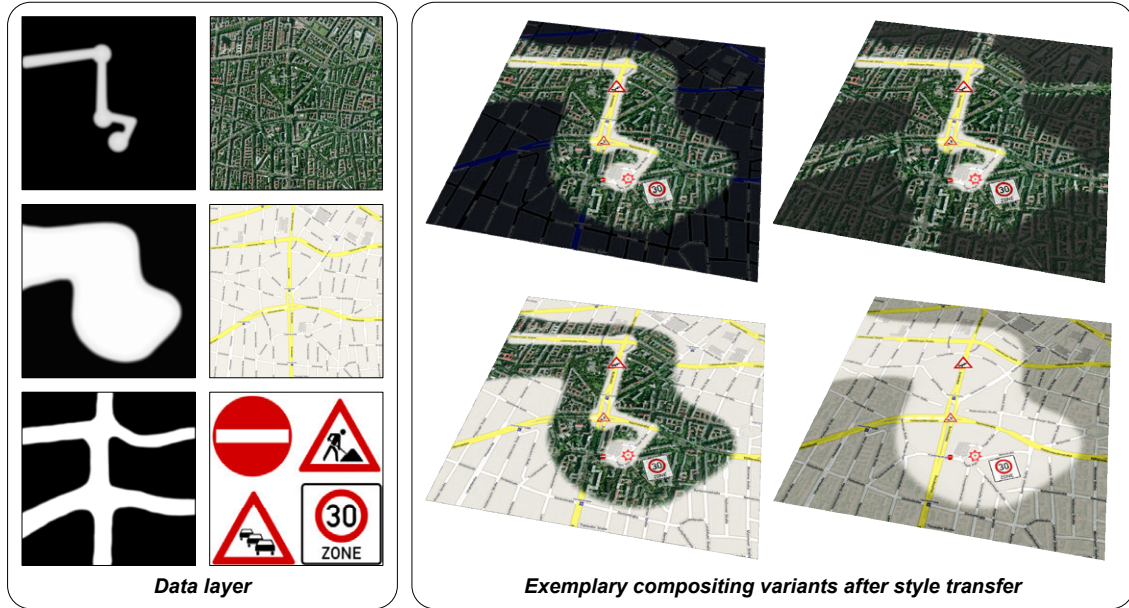
**Glyphs and Object Highlighting** Since the approach enables the rendering of a high number of projective mappings, it can be used to annotate 3D GeoVE with cartographic symbols (Fig. 5.5(a)) to facilitate the creation of 3D digital maps. The advantages of projecting symbols onto the city model geometry are the emphasis of the symbol-object relationship, the reduction of interpretation ambiguities, and the removal of label-scene occlusions. This delivers acceptable visual results for generalized versions of virtual 3D city models [95], viewed from a bird's-eye or a plan perspective.



(a) Exemplary set of symbols projected onto the surfaces of a generalized virtual 3D city model.

(b) Object-highlighting using various mask representing paths and landmarks.

Figure 5.5: Application examples for annotations (A) and object highlighting (B) of 3D geovirtual environments using the projective raster mapping technique.



**Figure 5.6:** Examples of 2D surface lenses implemented using the concept of projection raster mapping.

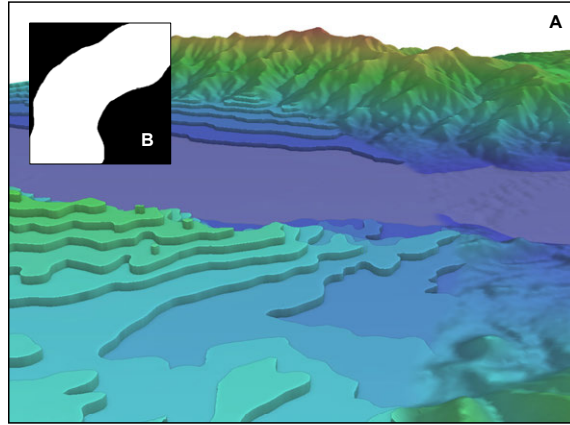
The icons can be exchanged, scaled, rotated, or faded view-dependently to ensure an optimal readability. It further enables view-dependent switching between different levels-of-abstraction (LOA) without adapting the projective mappings. A variant of the value mapping is used to highlight specific features 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 5.5(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  $\mathcal{W} = \{W_0, \dots, W_n\}$ , a point radius  $r \in \mathbb{R}_+$ , and an edge height  $h \in \mathbb{R}_+$ ,  $n$  projective mappings  $PM_i, i = 1, \dots, n$  for the way points are computed 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,  $n - 1$  mappings  $PM_j, j = n, \dots, 2n - 1$  for each edge between the way points  $W_S = W_{j-n}$  and  $W_E = W_{j-1-n}$  are computed. The respective setting is computed as follows:  $O_j = (W_S + W_E)/2$ ,  $A_j = \|W_E - W_S\| \cdot h$ , and  $B_j = N_P \times A_j \cdot |W_E - W_S|$ . The respective PMs uses a circular mask for each way point and a rectangular mask for each edge. This enables only the visualization of straight routes. This limitation can be compensated by deriving appropriated mask layers using a given street network.

**2D Surface Lenses** Projective mappings can be used for masking and blending different texture layers (Fig. 5.6). They can be used to implement 2D surface lenses [58]: multiple 2D textures represent independent layers of information, e.g., shading, land use information, and street networks while artificial textures represent masks and blending filters. To enable the rendering of 2D surface lenses without the use of multi-texturing, data layers  $L$  can represent *content data layer*, such as street and land-use information represented as independent thematic information layers, and *mask layer* presenting the regions-of-interest for which the content data layer is valid for. The particular data layer masks can be derived from feature data or drawn directly by the user onto the digital terrain model. In such use-cases, our approach has a number of advantages. It enables a continuous smooth transitions between data layers defined by 2D masks, which can have arbitrary shapes. The DTCCG enables the interactive movement of a lens, while the implicit hierarchy supports the usage of overlapping and nested lenses.

**Parametrization of 3D Iso-contours** Isocontours (also isopleths, isolines, level sets) are commonly used to visualize real-valued data defined over a 2D plane according to a set of given isovalues. To support the 3D landscape metaphor for information visualization, a 3D stepped terrain can be derived by lifting and extruding isolines to their particular isovalue, but typically requires triangulation of the resulting surface representation in a preprocessing step.

In [GTD10], a concept and rendering technique for triangle-based terrain models is presented that provides interactive, adaptive generation and visualization of such stepped terrains without preprocessing. The fully hardware-accelerated rendering technique creates additional step geometry for each triangle intersecting an iso-plane on-the-fly. Further, an additional interpolation schema facilitates smooth transition between established 3D terrain visualization and its stepped variant. Lenses are used to show different data representations in the same geographic space using a lens metaphor. The presented interpolation schema enables the application of a smooth lens to the visualization, which shows the original continuous terrain model inside the lens, and the abstracted, quantized model outside (Fig. 5.7). The implementation is based on an additional grayscale texture to represent the lens by values in the interval  $[0, 1]$ . By transforming the texture matrix or interactively drawing into the lens texture, the lens can also be dynamically adapted.



**Figure 5.7:** Exemplary visualization (A) for parametrization of 3D iso-contours using a 2D surface lenses (B).

## 5.5 Conclusions and Future Work

This chapter presents a general concept for dynamically mapping 2D raster-data to 2.5D and 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 even 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. Previous approaches for are limited by the number of texture units available or required multi-pass rendering for the image synthesis process.

In principle, the presented concept suffers from the same problems and limitations 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 orthographic projector matrices with a DOP along  $N_R$  are used, the 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 to 3D GeoVE that can be approximated by a planar surface.

For future work, the implementation can be extended to support out-of-core rendering for the texture atlases and 3D textures representing 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. Finally, the implementation can be optimized by performing the texturing in a deferred approach by using G-Buffers [222]. Thus, the computations are performed on visible fragments only, which decrease the runtime complexity for scene with high overdraw.



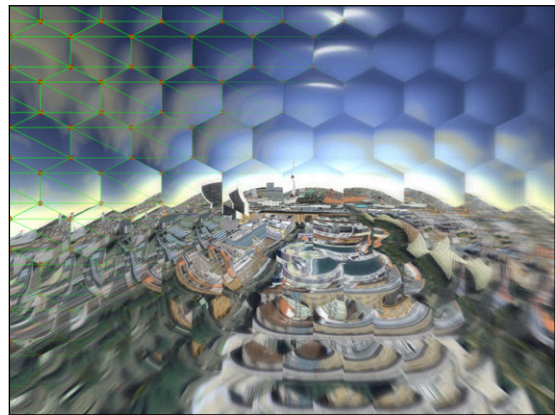
## Chapter 6

# Rendering Technique for Image-space Deformations

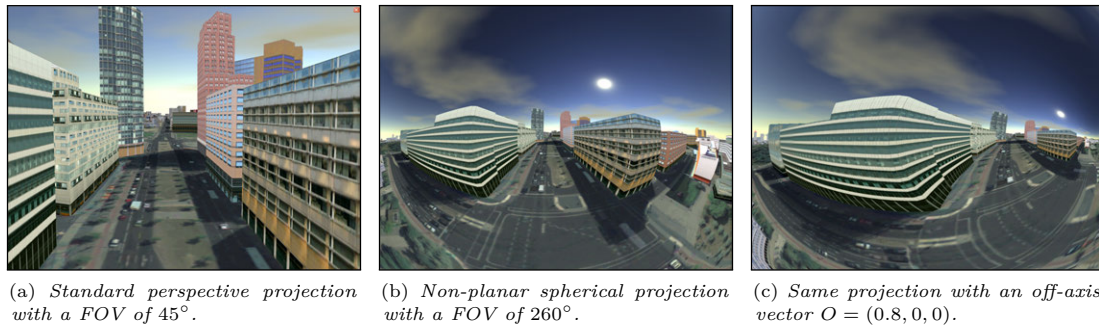
This chapter presents an image-based approach to efficiently generate multiple non-planar projections of arbitrary 3D scenes in real-time. The creation of projections such as panorama images or fisheye views has manifold applications in 3D geovirtual environments. The proposed rendering technique is based on dynamically created image-based representations of the geovirtual environment surrounding the virtual camera (cube map textures) in combination with shader programs that compute the specific projections. Based on this principle, an approach to customize and combine different planar as well as non-planar projections is presented. The technique can be applied within a single rendering pass, is easy to implement, and completely exploits the capability of modern programmable graphics hardware.

## 6.1 Non-planar Single-center Projections

This chapter presents an image-based rendering technique that overcome the field-of-view (FOV) limitations of the classical pinhole camera rendering pipeline and enables the application of non-planar projection on standard consumer graphics hardware in real-time. Examples are omni-directional panorama for non-planar screens, spherical dome projections, or more complex projections (Fig. 6.1). It further can be used to easily implement 2D focus+context lens techniques such as multi-focal fish-eye views. In particular, the rendering technique focuses on real-time modifications of perspective views that become possible due to recent hardware developments [20]. The presented approach is limited to single-center projections (SCOP). In spite of non-planar projection screens [184], this technique can also be used for rendering effects in games as well as to improve visibility in virtual landscapes [204] or virtual city environments. Thus, applications can apply extreme perspectives [90] and high FOV angles by having a reasonable survey of the scene [91] as well as a better size and depth perception [198]. The approach exploits dynamic environment mapping in combination with the programmable GPU. The separation of projection computation and cube map texture creation enables a broad range of optimization techniques. Existing image-based approaches for non-planar projections suffer mainly from the lack of interactive capabilities when used with complex geometric scenes or large viewports common for 3D GeoVE. This can be explained by the trade-off between the generality of the proposed frameworks and the efficiency of the rendering techniques. Furthermore, the parametrization are complex and cannot be intuitively controlled by the user [32].



**Figure 6.1:** *Interactive artistic rendering of a compound eye that is rendered using the concept of projection tiles (shown as green lines).*



**Figure 6.2:** Comparison between perspective and spherical projections.

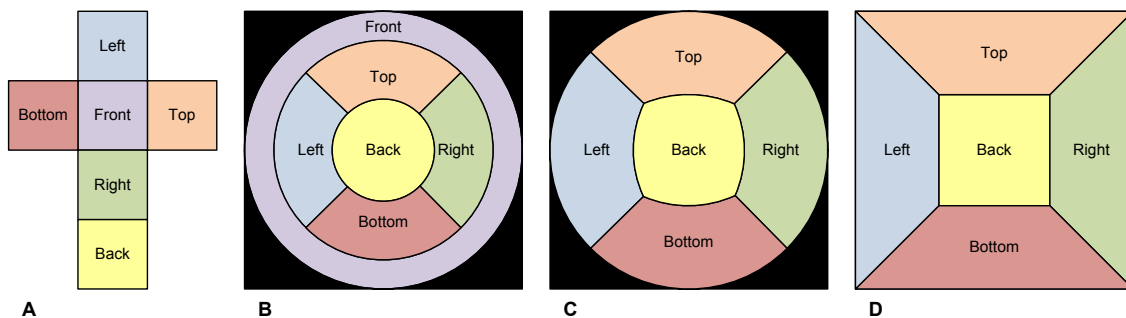
## 6.2 Real-time Rendering of Non-planar Projections

Before discussing *projection tiles*, this section describes a basic approach to generate multiple non-planar projections in real-time (Fig. 6.2), inspired by the idea described in [259]. This CPU-based technique renders six views with  $90^\circ$  FOV in each direction. Afterward, a mapping table is used to transform these pixels into a single view according to spherical or cylindrical projections. This approach can be transferred to a completely GPU accelerated implementation using cube map textures [189] and fragment shader functionality [207] that comprises of the following three components:

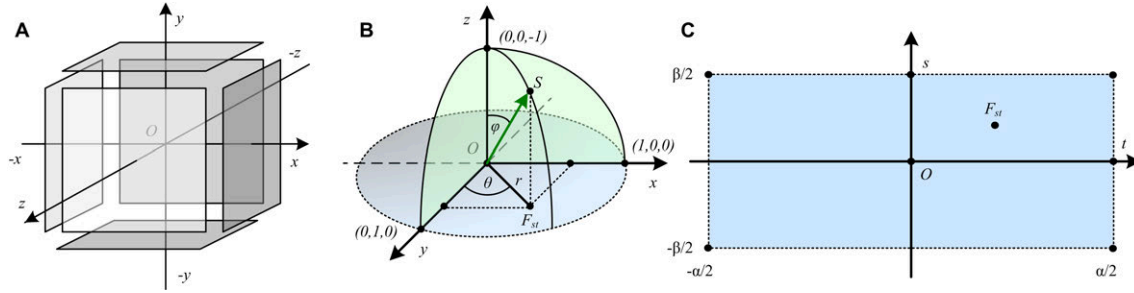
**Dynamic Environment Map:** Dynamic environment mapping [116] enables the image-based representation of the complete 3D GeoVE around the virtual camera. Figure 6.3 shows different types of environment maps. For this approach, a cube map texture representation is chosen because it is free of distortions, provides an optimal texture utilization, and is a fully hardware-accelerated feature. A dynamic cube map texture can be created by using single or multi-pass rendering (see Section 6.4).

**Projection Canvas:** A *projection canvas* (PC) is a parametrized geometry that represents the planar area on which a particular projection is rendered to. This can be a screen-aligned quad that covers the complete viewport or a quad that is placed within the 3D scene. Exemplary parametrizations of the projection canvas are shown in Figure 6.4.B and 6.4.C.

**Shader Functionality:** The presented rendering technique relies on fragment shader functionality to encapsulate the projection math by implementing a so-called *projection function* that is evaluated for each point on the projection canvas. A projection function is a mapping of a 2D point on the parametrized PC to a 3D cube map sampling vector. During rendering, the hardware rasterizer interpolates the parameters of the PC, which are subsequently evaluated by the shader program that implements a specific projection function.



**Figure 6.3:** Comparison between different types of environment maps. A: Cube map texture that consists of six equal-sized 2D textures. B: A sphere map that utilizes only 78% of the available texture space and contains distorted areas (front). C: Back-paraboloid environment map. D: Pyramidal environment map as a seldomly used type that fully utilizes the available 2D texture space.



**Figure 6.4:** Overview of the utilized coordinate systems for creating non-planar projections. A: Coordinate system and orientation of a cube map texture. B: Polar coordinates for computing spherical projections. C: Parametrization of the projection canvas that is suitable for cylindrical projections.

**Projection Functions** To compute a non-planar projection of the geovirtual environment represented by the cube map, a sampling vector  $S = (x, y, z) \in \mathbb{D}^3$  for each fragment  $F_{st} = (s, t) \in \mathbb{D}^2$  on the rasterized projection canvas is determined. Where  $\mathbb{D} = [-1, 1] \subset \mathbb{R}$  is the normalized coordinate space for the canvas. Figure 6.4 shows the parametrization of the cube map (A) and the projection canvas (B and C). A projection function  $\delta_{\mathcal{P}}(F_{st}) = S$  for a projection  $\mathcal{P}$  can be defined as:

$$\delta_{\mathcal{P}} : \mathbb{D}^2 \longrightarrow \mathbb{D}^3 \quad (s, t) \longmapsto (x, y, z)$$

Figure 6.5.A shows a horizontal cylindrical projection  $\mathcal{C}$  that can be formulated as instance  $\delta_{\mathcal{C}}(F_{st}, \alpha, \beta) = S$  with an horizontal FOV of  $2 \cdot \alpha$  and a vertical FOV of  $2 \cdot \beta$ :

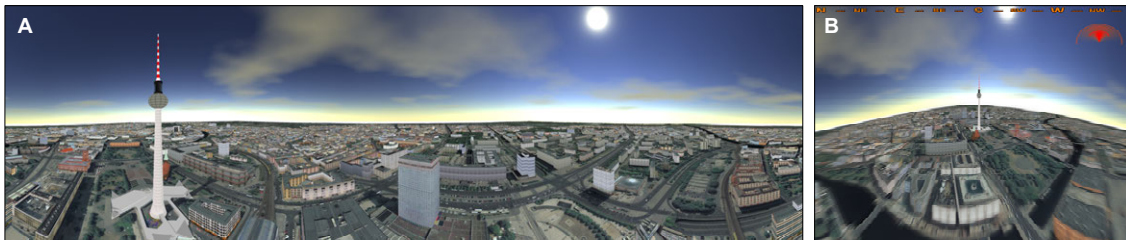
$$x = \cos(s \cdot \alpha) \quad y = t \cdot \tan(\beta) \quad z = \sin(s \cdot \alpha)$$

Figure 6.4.C shows a possible parametrization of the projection canvas. Further, a spherical projection  $\mathcal{S}$  with an FOV of  $\gamma$  can be expressed as  $\delta_{\mathcal{S}}(F_{st}, \gamma) = S$  with:

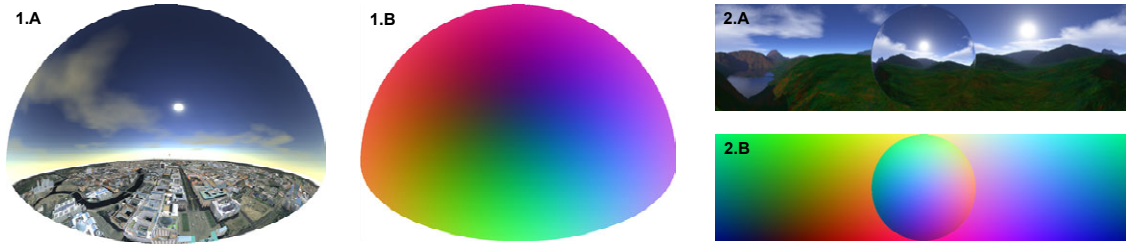
$$\begin{aligned} x &= \sin(\theta) \cdot \cos(\phi), & \phi &= \arctan(t, s) \\ y &= \sin(\theta) \cdot \sin(\phi), & \theta &= r \cdot \gamma / 2 \\ z &= \cos(\theta), & r &= \sqrt{s^2 + t^2} \end{aligned}$$

Figure 6.5.B shows an resulting example of this projection while Figure 6.4.B displays the mapping between a point on the projection canvas ( $F_{st}$ ) and cube map sampling vector  $S$ . The ideal hemisphere differs from an approximation for fisheye lenses [184] with  $\theta = 1.411269r - 0.094389r^3 + 0.25674r^5$ .

**Adjusting the Orientation of the Projection Camera** The above procedure assumes a cube-map camera orientation toward the negative  $z$ -axis. To decouple the orientation of the cube map and the projection, the sampling normal  $S$  has to be adjusted before sampling the cube map. If the cube map texture is created in the standard orientation (Fig. 6.4.A),  $S$  have to be corrected by transforming it with respect to the current parameters of the projection camera orientation. Let  $\mathbf{C}$  be an orthonormal base constructed from the current look-to vector  $L_T = (x_T, y_T, z_T) \in \mathbb{D}^3$ , look-up vector  $L_U = (x_U, y_U, z_U) \in \mathbb{D}^3$  and the cross product  $L_C = (x_C, y_C, z_C) = L_T \times L_U$ .



**Figure 6.5:** Exemplary applications of the presented rendering technique. A: Panoramic (cylindrical) projection with  $360^\circ$  horizontal FOV. B: Spherical projection with a FOV of  $180^\circ$  and applied motion blur.



**Figure 6.6:** Examples of the normal-map optimization technique. Figure 1.B and 2.B show the normal map for the respective projection 2.A and 2.A. Sub-figure 2 demonstrates the result of blending two normal maps (cylindrical and spherical) to obtain a mixed projection.

Subsequently, for a projection  $\mathcal{P}$ , a fragment  $F_{st}$ , and orientation of a projection camera  $\mathbf{C}$ , the final sampling vector  $V$  is computed by:

$$V = (\mathbf{C} \cdot \delta_{\mathcal{P}}(F_{st} \cdot s)) - O \quad \mathbf{C} = \begin{bmatrix} x_T & y_T & z_T \\ x_U & y_U & z_U \\ x_C & y_C & z_C \end{bmatrix}$$

The vector  $O \in \mathbb{D}^3$  is denoted as off-axis vector. Figure 6.2.C demonstrates an example for an off-axis projection. The scalar term  $s \in \mathbb{D}$  can be interpreted as a zooming parameter.

**Normal-map Optimization** If the parameters of a projection are constant, a simple optimization method can be applied. To avoid redundant shader computations at runtime, the sampling vector  $S$  can be stored into a high-precision 32Bit IEEE conformal floating-point texture map [176]. This reduces the shader execution costs for computing a projection function to two texture look-up operations. For full screen projections, the resolution of the texture map should match the viewport resolution. Figure 6.6 shows examples of projections (A) and their associated normal maps (B). The different value domains of normals and color are compensated for viewing. In contrast to tangent-space normal-maps, which store a normal regardless of its relative position in the tangent texture-space, the proposed technique uses unit-space normal maps. This is necessary for the correct sampling of the cube map texture later. The cube map sampling vector in unit-space  $N_{st}^U$  can be derived from a tangent-space normal  $N_{st}^T$  for each fragment  $F_{st}$  by:

$$S = \delta_{\mathcal{N}}(F_{st}) = N_{st}^U = \|N_{st}^T + (s, t, 0)\|$$

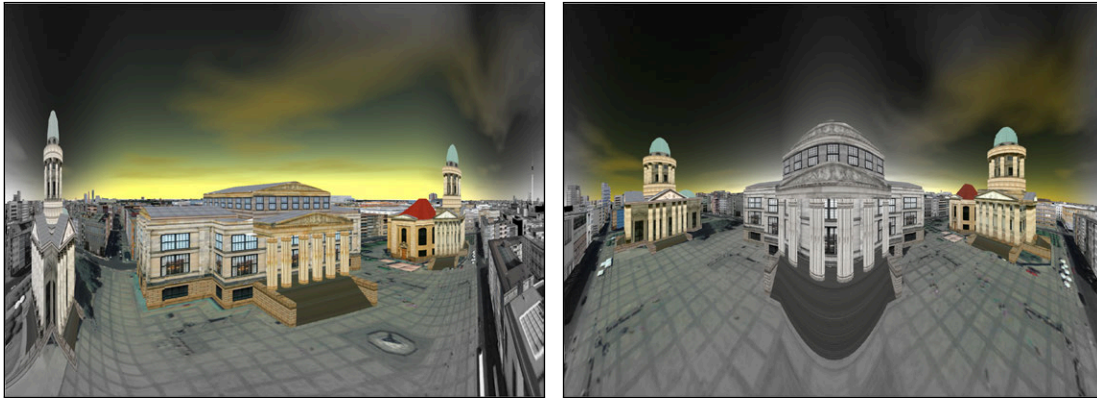
This optimization technique adds an additional parametrization to the approach: normal-maps can be stored using floating-point images formats, and blended to combine different normal maps to achieve mixed projections (Fig. 6.6.B). This approach enables the creation and combination of more complex projections using *projection tiles*.

### 6.3 Generalization of Single-center Projections

The projection tile method combines planar projections as well as different non-planar derivatives, and it facilitates the creation of custom projections, which would be hard to specify analytically. Projection tiles are a generalization of the approach described in Section 6.2. Since non-planar projections can be represented as normal maps, the question arises how these textures can be created and manipulated in an efficient way. For this purpose, projection tiles provide an additional parametrization of the projection canvas. The generalization is implemented by assigning a specific view direction to each fragment of the projection canvas. This approach is based on three components that match the programming model of current polygonal hardware rasterizers:

**Tile Feature:** In general, a *tile feature* (TF) describes a viewing direction for a specific point on the projection canvas. Additionally, a tile feature can be attributed with custom, user-defined attributes such as scaling and blending factors.





(a) Composition of a planar projection in the center and cylindrical projections left and right.

(b) The same scene with two planar projections for each cathedral.

**Figure 6.7:** Examples for combining planar and non-planar projections using projection tile screens by the example of the virtual 3D city model of Berlin. The images show the virtual 3D model of the Gendarmenmarkt with the German Church and the French Cathedral. The saturation fall-off is controlled by the respective tile features.

**Projection Tile:** A projection tile (PT) is a container that consists of a finite number of tile features. A projection tile controls the interpolation of the viewing directions and the custom attributes of a tile feature.

**Projection Tile Screen:** A projection tile screen (PTS) is a container for projection tiles. Together with projection tiles and tile features, it represents a parametrization of the projection canvas. At runtime, PTSs can be changed and updated. During rendering, tile screens will be transformed into normal maps in order to integrate in the existing framework.

This principle is analog to polygonal data representations: here, a tile feature could be interpreted as an attributed vertex, a projection tile as a primitive, and a projection tile screen as a polygonal mesh. This design matches polygonal rendering hardware and thus, is easy to implement. It further facilitates the integration into existing digital content creation (DCC) pipelines by reading and writing 3D meshes. Thereby, custom feature attributes can be encoded as texture coordinate values.

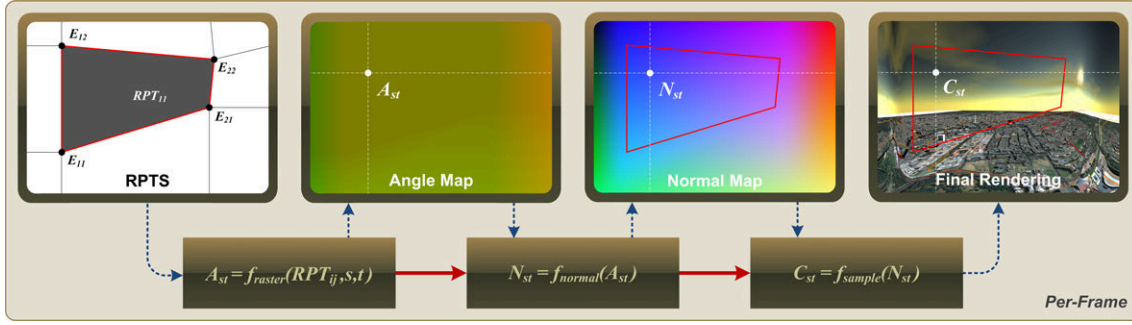
**Tile Features** To enable an intuitive way to describe a TF, polar coordinates  $\phi$  and  $\theta$  are chosen to express the view direction instead of using a normalized direction vector. A tile feature  $E$  is defined as a 6-tupel:

$$E = (x, y, \phi, \theta, s, f) \quad x, y, s, f \in [0, 1] \subset \mathbb{R} \quad \phi, \theta \in [-360, 360] \subset \mathbb{R}$$

that contains mainly the 2D feature position  $(x, y)$  and the particular horizontal and vertical view angles  $(\phi, \theta)$ . The parameter  $s$  is a scaling factor while the variable  $f$  can be used to bind custom parameters to each feature. For example, Figure 6.7 demonstrates this by adjusting the image saturation according to the value of  $f$ . The universe of all tile features is denoted as  $\mathcal{E}$ .

**Projection Tiles and Projection Tile Screens** A projection tile defines the area of a specific projection in relation to the projection canvas and consists of a number of tile features  $E$ . Projection tiles are organized in a projection tile screen (PTS), which is represented by a specific tile set  $\mathcal{T}$ . Two different approaches of grouping tile features with different properties and beneficial features are proposed: rectangular and triangular-shaped projection tiles.

A rectangular projection tile (RPT) consists of four tile features that are organized in a regularly structured rectangular projection tile screen (RPTS)  $\mathcal{T}_{mn}^{RPT}$  with a resolution of  $n$  rows and  $m$  columns:



**Figure 6.8:** Elaborated conceptual pipeline to render combinations of planar and non-planar projections defined by rectangular projection tiles. A projection tile screen is transformed into a normal map that contains the cube map sampling vectors to create the final rendering.

$$RPT_{kl} = (E_{(k,l)}, E_{(k+1,l)}, E_{(k+1,l+1)}, E_{(k,l+1)}) \quad \mathcal{T}_{mn}^{RPT} = \begin{bmatrix} E_{0n} & \cdots & E_{mn} \\ \vdots & \ddots & \vdots \\ E_{m0} & \cdots & E_{m0} \end{bmatrix}$$

Using rectangular tiles introduces a number of disadvantages. First of all, not every tile shape can be represented with four tile features. The structure of the rectangular projection tile screen  $\mathcal{T}_{mn}^{RPT}$  does not enable non-continuous transitions between projection tiles. The regular structure of RPTS and its associated RPT enable an easy direct manipulation. The concept of *triangular projection tiles* (TPT) can be used to overcome these drawbacks of the RPTs:

$$TPT_i = (E_0, E_1, E_2), \quad E_i \in \mathcal{E}, \quad \mathcal{T}_n^{TPT} = TPT_0, \dots, TPT_n$$

With TPTs, a user is able to control the triangulation of the projection canvas directly. This degree of freedom enables the usage of 2D lenses as shown on Figure 6.17.B in Section 6.8. TPTs also introduce non-continuous transitions between the tiles but are difficult to manipulate directly by the user.

**Mapping Projection Tile Screens to Normal-maps** The concept of projection tiles can be integrated in the existing framework by using the normal-map optimization method by mapping a PTS to a normal map. Figure 6.8 gives an overview of the complete process. A PTS (RPTS or TPTS) is transformed into a normal map by rendering all of its respective projection tiles (RPT or TPT) into an off-screen 2D texture-target using *render-to-texture* [273]. Thereby, the feature components  $x$  and  $y$  are interpreted as the 2D vertices of a quad or triangle in a standard orthographic projection [271].

The angles and user defined attributes are encoded into per-vertex texture coordinates. This avoids any domain-specific encoding of these values. The hyperbolic interpolation [19] between these values is performed by the graphics hardware. A fragment shader converts the horizontal and vertical angles  $A_{st} = (\phi_{st}, \theta_{st})$ , obtained for each fragment  $F_{st}$ , into a respective normal  $N_{st}$  and outputs it into the texture target. The normal vector can be computed by:

$$N_{st} = \delta_{\mathcal{PT}}(F_{st}) = \left\| \mathbf{R}_x(\theta_{st}) \cdot \left( \mathbf{R}_y(\phi_{st}) \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) + \begin{bmatrix} s \\ t \\ 0 \end{bmatrix} \right\|$$

Where  $\mathbf{R}_x$  and  $\mathbf{R}_y$  denote the 3D rotation matrices which transform the base normal  $(0, 0, 1)$  around the respective  $x$  and  $y$  axis. For each  $N_{st}$  in the resulting normal map a sampling vector  $S$  can be computed by setting  $\delta_{\mathcal{PT}}(F_{st} \cdot s_{st}) = N_{st}$ .

## 6.4 Interactive Rendering Process

The implementation of the rendering process is designed to meet the requirements for real-time visualization of geometrical complex scenes. An prototypical implementation was done by using OpenGL [230] in combination with the OpenGL shading language (GLSL) [144]. It uses framebuffer objects, floating point textures, and mip-mapped cube maps for dynamic texturing [113]. For interactive rendering, a two-phase process is applied: (1) create or update the dynamic cube map and (2) rendering of the projection canvas using the respective projection.

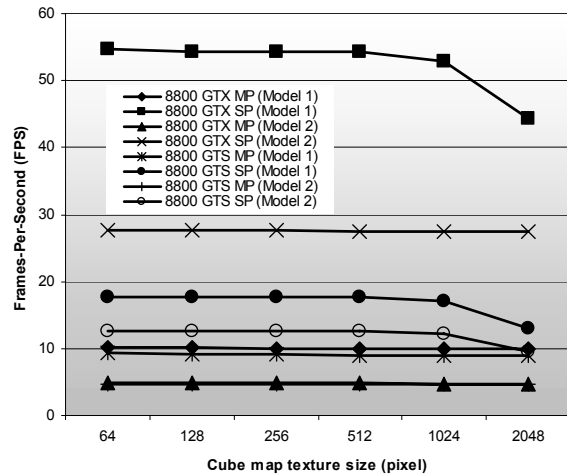
### Hardware-accelerated Cube Map Creation

The synthesis and update of dynamic cube map textures [102] is an essential process that can be implemented using single-pass or multi-pass rendering techniques. On modern graphics hardware [230], it is possible to create cube map textures within a single-pass rendering by utilizing *geometry shaders* and *layered-rendering* [264]. This so called *render-to-cube-map* technique duplicates each input triangle six times and applies a separate model-view transformation for each face of the cube map texture. Each of the duplicated triangle is directed to the respective layer of a layered render target [20]. On the other hand, a cube map texture can be created using multi-pass rendering in combination with *render-to-texture*. Given a reference camera position, the six local virtual cameras with a FOV of  $90^\circ$ , and an aspect ratio of 1 can be derived, which are used to render the 3D scene into the respective cube map texture targets. Both approaches differs with respect to runtime performance and integration costs. Figure 6.9 shows a runtime comparison of both approaches for two models of different geometric complexity (Model 1: 41,032, Model 2: 46,060 vertices).

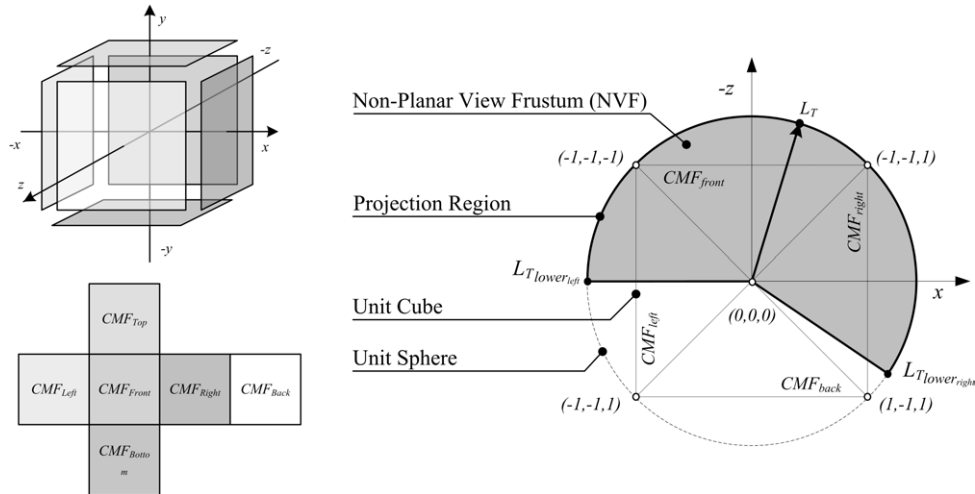
The test application does not utilize the second CPU core. The comparison shows the advantage of the single-pass cube map creation compared to the multi-pass approach. Rendering within a single pass has a main advantage: The scene has be traversed only once. This results in fewer state changes per frame. So, fast cube map creation enables the utilization of more than one dynamically created cube map.

**Optimization Techniques** Depending on the scene complexity, the multi-pass creation of a cube map texture can become costly. Usually, the creation is fill-limited due to the increased rasterization effort as well as bandwidth-limited due to the number of rendering passes. For general 3D scenes, it is often not possible to use proxy geometry to speed up the synthesis process or to distribute the rendering of each cube map face to different frames. There are two main optimization techniques that are able to counterbalance the performance problem:

A *Cube-Map Face Optimization* (CMF Optimization) omits the update of cube map faces (CMF) that are not visible in the generated projection. To determine which cube map faces have to be updated, the spherical coordinates for each corner vertex of the unit cube and the current look-to vector  $L_T$  are computed. Then, a non-planar view frustum (NVF) is defined by offsetting the spherical coordinates of  $L_T$  with the horizontal and vertical angles of the current projection. If one of the face vertices is inside the projection region, the associated face will be updated. Figure 6.10 demonstrates this by considering the lower vertices of the unit cube only. For the current look-to vector  $L_T$  the update of  $CMF_{back}$  is omitted. Further, a *Look-To Optimization* can be used to omit updates of the cube map texture if the camera position has not changed.



**Figure 6.9:** Run-time performance for multi-pass (MP) and single-pass (SP) cube map creation techniques.



**Figure 6.10:** Parameter space for the cube-map face optimization (CMF Optimization). The cube map is displayed from the positive  $y$ -axis. The gray area represents the non-planar view frustum (NVF) of a non-planar projection.

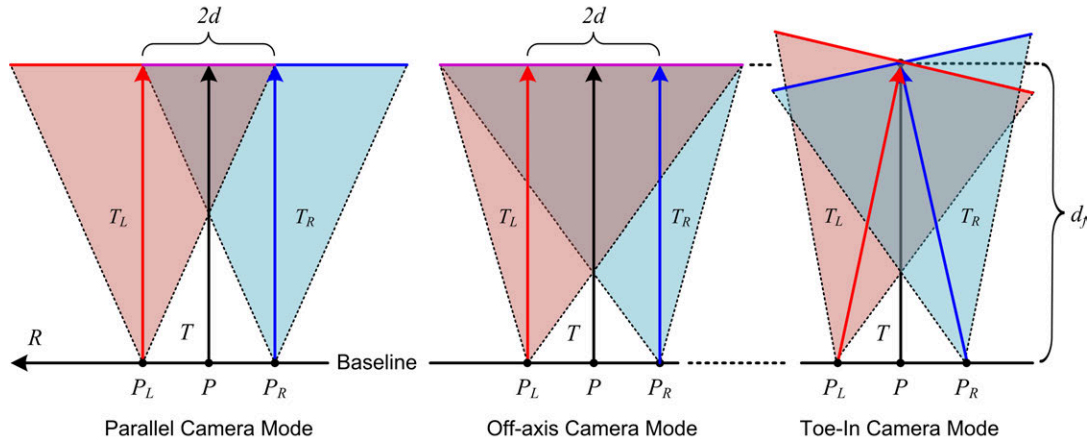
**Rendering Projection Functions** Given a generated cube map, a particular projection is applied in a post-processing pass subsequent to the cube map creation pass(es). For rendering a full-screen non-planar projection the following three steps are performed: (1) a standard 2D orthographic projection is setup. The camera is set to the standard orientation with a look-to vector  $L_T = (0, 0, -1)$ . (2) a specific fragment program that implements the mathematical concepts as described in Sections 6.2 and 6.3 is activated. The shader program performs cube map texture look-ups or outputs the computed normal vectors for later re-use. Finally, (3) a screen-aligned quad (projection canvas) with standard texture coordinates is rendered that covers the entire viewport.

## 6.5 Stereoscopic Rendering Non-planar Projections

Stereoscopy is the phenomenon of simultaneous vision with two eyes, producing a perception of the relative distances between objects in space, is an emerging field in cinematography and gaming. Today, this feature is a requirement in 3D immersive digital environments. Stereo vision, as an additional visual cue for humans, is an important method to increase the immersion into 3D virtual environments. Stereoscopic effects can be created by using a stereo image pair displayed with active or passive stereo viewing concepts, which enable the experience of the stereo sensation. Creating stereo image pairs is straightforward for standard planar projections that can be accelerated by graphics hardware. The renderer only needs to create an image pair, one image for the left eye and one for the right eye.

Most of today's computer games and visualization frameworks offer a 3D stereo mode for standard planar projections. Enabling interactive stereo rendering for non-planar projections is not a trivial problem. This is especially true for rendering on polygonal consumer graphics hardware without the support of parallel or distributed systems. The optimal solution for this problem enfoldes the following attributes: It should enable rendering at interactive frame rates for large-scale 3D models, such as virtual 3D city models or landscapes on current consumer hardware. Further, the approach should be applicable to multiple variants of single-center projections, support omni-directional stereo, deliver high-quality images, and should be easy to implement and integrate into existing rendering frameworks.

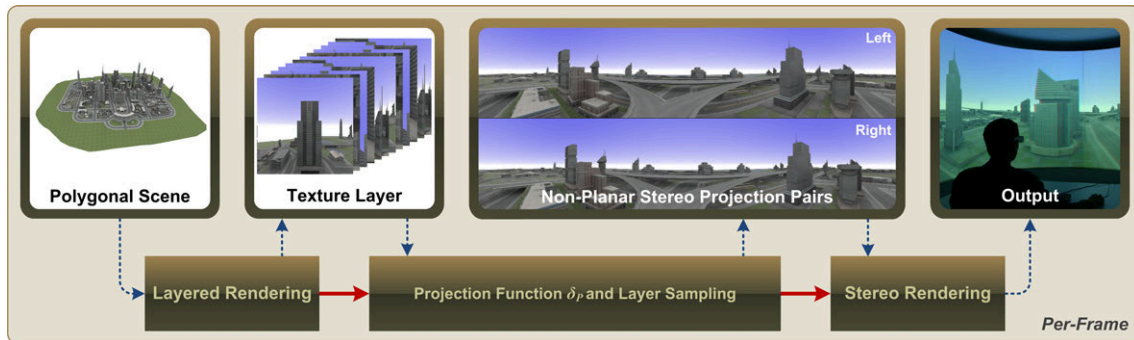
While generating stereo images is well known for standard projections, the implementation of stereoscopic viewing for interactive non-planar single-center projections, such as cylindrical and spherical projections, is a challenge. This section presents the results of adapting an existing image-based approach for generating interactive stereoscopic non-planar projections for polygonal scenes on consumer graphics hardware [TLD09]. The extended rendering technique creates stereo images within a single scene rendering pass.



**Figure 6.11:** Comparison of the parametrization for the parallel, off-axis, and toe-in camera orientation modes for generating stereo pairs.

**Stereoscopic Viewing** To achieve angle and depth disparity which create the stereo effect when viewing, the scene has to be rendered from two virtual eye positions. The eye separation is a distance  $d \in \mathbb{R}_+$  in world space. The average distance between the two eyes is  $2 \cdot d = 6.5\text{cm}$ . Given a virtual reference camera orientation  $C = (P, T, U)$ , the camera settings for the left  $C_L = (P_L, T_L, U_L)$  and right eye  $C_R = (P_R, T_R, U_R)$  are computed. Current applications use three different methods [29] to obtain these orientations (Fig. 6.11) by maintaining the up-vector  $U = U_R = U_L$ : In *Parallel Camera Mode*, the camera position is shifted along the baseline by maintaining the look-to direction, thus  $T = T_L = T_R$ . The direction of the baseline corresponds to the right vector  $R = U \times L$ . This results in the camera orientations:  $C_L = (P - d \cdot R, T_R, U_R)$  and  $C_R = (P + d \cdot R, T_L, U_L)$ . In addition to the shift of the camera position, the *Toe-In Camera Mode* adjusts the look-to vectors with respect to a focal distance  $d_f \in \mathbb{R}_+$ . The respective look-to vectors are defined as follows:  $T_L = \|(P + T \cdot d_f) - P_L\|$  and  $T_R = \|(P + T \cdot d_f) - P_R\|$ . The *Off-Axis Camera Mode* is similar to the parallel camera mode but uses a non-symmetric camera frustum as described in [29] (Fig. 6.11.B).

**Creating Raster Representations for Stereoscopic Rendering** Regardless of the rendering techniques used for creating non-planar projections, the creation of stereoscopic views comprises the following two basic steps (Fig. 6.12): First, the NPP for the left and right images are rendered by using image-based rendering techniques. Both require at least a single scene rendering pass. Second, the resulting stereo pairs are combined into a single framebuffer (passive stereo) or rendered into two framebuffers (active stereo) by using post-processing compositing passes. The extension requires a single scene rendering pass and two additional full-screen post-processing passes for each projection.



**Figure 6.12:** Overview of the implementation concept for image-based stereo rendering for non-planar projections. Layered rendering is used to create image representations of the input geometry. These images are combined into stereo pairs of non-planar projections, which are displayed subsequently.

```

1  uniform mat4 VPM[12]; //View projection matrices
2
3  bool cullViewFrustum(in vec4 P[3]) { // view-frustum culling
4      const vec4 plane = vec4(-1.0, -1.0, 1.0, 1.0); vec4 T[3];
5      T[0] = clamp(P[0].xyxy * plane - P[0].w, 0.0, 1.0);
6      T[1] = clamp(P[1].xyxy * plane - P[1].w, 0.0, 1.0);
7      T[2] = clamp(P[2].xyxy * plane - P[2].w, 0.0, 1.0);
8      return !any(notEqual(T[0]*T[1]*T[2], vec4(0.0)));
9  }
10 bool cullBackFace(in vec4 P[3]) { // back-face culling
11     vec2 d0 = P[1].xy * P[0].w - P[0].xy * P[1].w;
12     vec2 d1 = P[2].xy * P[0].w - P[0].xy * P[2].w;
13     float w = min(min(P[0].w, P[1].w), P[2].w);
14     return d1.x * d0.y < d0.x * d1.y || w <= 0.0;
15 }
16 void main(void) {
17     for(int face = 0; face < 12; ++face) {
18         gl_Layer = face; // assign layer ID
19         vec4 P[3]; // compute screen coordinates
20         P[0] = VPM[face] * gl_PositionIn[0];
21         P[1] = VPM[face] * gl_PositionIn[1];
22         P[2] = VPM[face] * gl_PositionIn[2];
23         // perform culling algorithms
24         if(cullViewFrustum(P) && cullBackFace(P)) {
25             for (int i = 0; i < 3; i++) {
26                 gl_Position = P[i]; //Fill further interpolants
27                 EmitVertex();
28             } // endfor
29             EndPrimitive();
30         } // endif
31     } // endfor
32 }

```

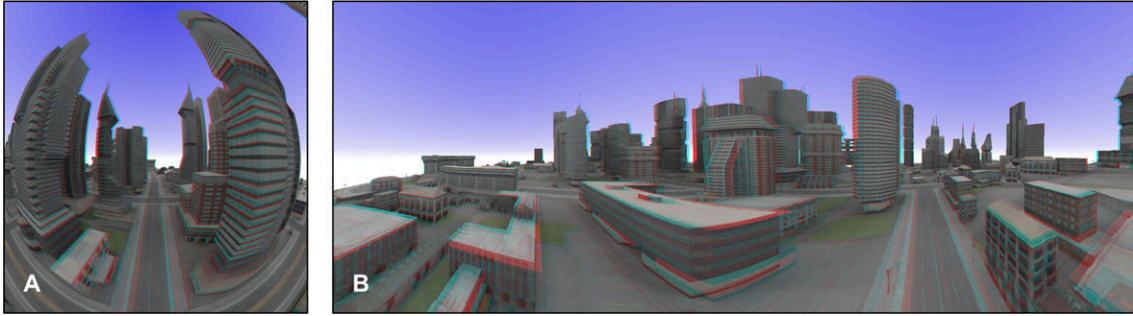
**Listing 6.1:** Implementation of single-pass render-to-cubemap using a geometry shader.

The present generation of raster-based polygonal rendering hardware [20] enables the application of *layered rendering*. Using the support of geometry amplification, the raster representation of the geovirtual environment is created in a single rendering pass. To implement a layered-rendering technique, geometry shader functionality is required. A geometry shader duplicates and emits triangles that are transformed into the camera-coordinate system of the respective cube map face and then projected. The view-projection transformation matrices are computed and bound as shader constants. A layer ID (0-11) is assigned to every emitted triangle. It defines the target layer of the framebuffer object. The geometry shader shown in Listing 6.1 implements the main logic of the concept. It uses a conservative view-frustum culling (`cullViewFrustum`) and back-face culling (`cullBackFace`) [194].

## 6.6 Extensions for Stereoscopic Rendering

After the NPPs for the left and right eye are created, the rendering technique has to perform the image synthesis for stereo viewing. Basically, there are three different rendering modes: (1) rendering for *passive* stereo (e.g., anaglyph and chroma-depth), (2) rendering for *active* stereo displays or glasses, and (3) rendering for *auto-stereoscopic* displays. This section describes the implementation of passive and active stereo rendering and do not discuss the support of auto-stereoscopic displays.

**Rendering for Passive and Active Stereo** Passive stereo viewing is independent of the refresh rates of the output device and can be achieved by using mainly two methods: anaglyph or polarized rendering. Anaglyph images (Fig. 6.13, next page) provide a stereoscopic 3D effect when viewed with two colored glasses, each with a chromatically opposite color (usually red and cyan). The picture contains two differently filtered color images, one for each eye. This can be implemented by computing two projections and performing a full-screen compositing pass. The image-based approach can be optimized by computing only a single projection function and perform texture sampling from the left and right raster representations. Anaglyph glasses may introduce problems with reproducing correct colors. This can be implemented by converting the color images to gray scale stereo pairs before compositing and the application of a color optimization scheme as described



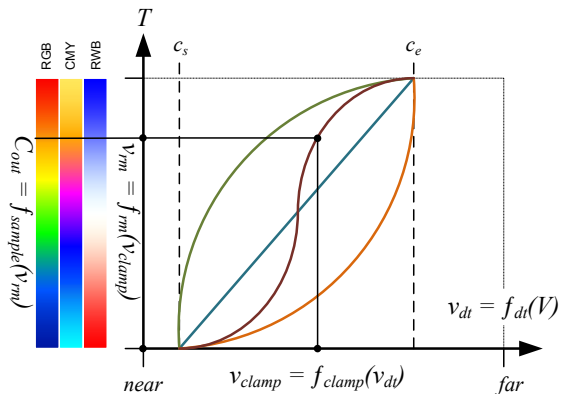
**Figure 6.13:** Examples of passive anaglyph stereo rendering. A: a spherical projection; B: a cylindrical projection.

in [282]. Another possibility is the use of polarized screens (3D monitor) or projector filters in combination with polarized glasses. The image-based approach, this requires the generation of two projections for the respective buffers.

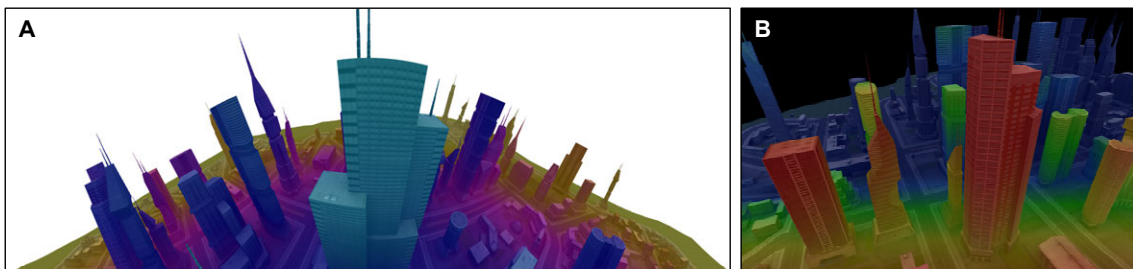
Frame-sequential, active stereo can be achieved by using shutter glasses that are synchronized with the graphics hardware. Here, alternate left and right images are displayed on the screen, multiplexed in time. The user wears LCD shutter glasses which alternately show and hide the left right views. Active stereo can be implemented using OpenGL and a quad buffer [3]. This feature requires a professional graphics card with hardware stereo support. Similar to polarized rendering, it requires the evaluation of the projection function twice.

#### Rendering for Chroma-depth Stereo

Chroma stereoscopy [239], or color stereoscopy, is a three-dimensional viewing approach that does not rely on binocular parallax and convergence. This chroma-depth technology is an inexpensive way to achieve a 3D impression of images that is compatible across different media such as paper or monitor displays [TLJD12]. The user needs to wear a chroma-depth glass that performs a color shift for one-eye. Unlike anaglyph or polarized stereo viewing approaches, the depth perception using chroma-depth glasses relies on color coding the scene with respect to the current camera settings (Fig. 6.14). A mapping from the coordinates of a scene point  $V = (x, y, z)$  to a normalized color value  $C_{out}$  of the respective chroma palette has to be computed. Figure 6.15 shows an overview of the components and participating functions for such color mapping. The structure of such a function depends on the representation of the color palette and the mapping of the point coordinates into a palette index. Similar to [11], color tables are represented as



**Figure 6.15:** Components and functions for the chroma-depth color mapping.



**Figure 6.14:** Examples of chroma-depth stereoscopy rendering. A: A CMY chroma-depth rendering of a horizontal cylinder projection with  $360^\circ$  horizontal and  $90^\circ$  vertical field of view. B: A RGB chroma-depth rendering of standard planar projection with  $60^\circ$  field-of-view.

**Table 6.1:** Performance comparison of image-based rendering for creating anaglyph stereoscopic views. The measurements (FPS) are taken for a 360° cylindrical projection with anaglyph passive stereo and a viewport resolution of 2048 × 768 pixels.

Triangles	Passes	8800 GTS	GTX 280
34,596	2	6.01	20.93
	1	6.37	29.23
236,276	2	0.80	7.82
	1	0.84	8.95
540,655	2	0.57	3.57
	1	0.39	4.13
3,210,162	2	0.11	0.45
	1	0.09	0.60

1D texture maps. This enables the flexibility to change the color models, e.g., red-green-blue (RGB), red-white-blue (RWB), or cyan-magenta-yellow (CMY), during runtime. Standardized color tables can be obtained from the respective vendors of the chroma-glasses. Given a color  $C_{in}$  obtained by shading and texturing the input scene, the color palette  $P$  represented as 1D texture, a mixing scalar  $m$ , and the eye-space coordinates of a point  $V$ , the output color is obtained by  $C_{out} = f_{lerp}(C_{in}, f_{adjust}(f_{sample}(P, T), m))$ . The sampling function  $f_{sample}$  performs a texture look-up in  $P$  using the generated texture coordinates  $T$ . The  $f_{lerp}$  function performs a linear interpolation between  $C_{in}$  and  $C_C$  with respect to  $m$ . The texture coordinates are obtained by the concatenation of the following 1D functions:  $T = f_{rm}(f_{clamp}, f_{dt}(V))$ . The distance-transform function  $f_{dt}$  returns a linear scalar that is clamped and re-normalized using  $f_{clamp}$ , and finally re-mapped by  $f_{rm}$ . The distance-transfer function is parameterized with respect to the projection that is used. For a standard planar projection, the output equals the value of the  $z$  coordinate of  $V$  in eye space. For image-based non-planar projections  $z = |V_0 - V|$ , i.e., the distance of  $V$  to the camera position  $V_0$ .

**Performance Evaluation for Stereoscopic Non-Planar Projections** The performance evaluation is conducted on two different platforms: NVIDIA GeForce 8800 GTS GPU with 640MB video RAM on an AthlonTM64 X2 Dual Core 4200+ with 2.21 GHz, 2 GB of main memory, as well as NVIDIA GeForce GTX 280 with 1024 MB video RAM on a Intel Core2 Duo, 3 GHz 3,25 GB of main memory. Table 6.2 shows a comparison of the two different cube map creation alternatives described in Section 6.5 with respect to the number of input triangles. Each test comprises the creation of two cube maps or one texture array with 1024<sup>2</sup> texture resolution and the rendering of a horizontal 360° cylindrical projection with a viewport size of 2048 × 768 pixels using anaglyph stereo viewing. No cube map face culling techniques were used.

## 6.7 Comparison of Image-based and Geometry-based Approaches

This section compares the presented image-based approach (IBA) with a geometry-based approach (GBA) described in [168]. This GBA implementation projects all mesh vertices non-planarly and rasterizes the primitives immediately [237]. The inadequate linear interpolation during rasterization requires highly tessellated meshes for artifact-free renderings. Dynamic mesh tessellation based on instancing [28, 247], geometry shaders [167], or hardware tessellation units [246] can ensure this property for arbitrary meshes. An alternative approach is tessellating the non-planar projection into smaller and simpler projections. Both approaches are compared w.r.t. the following criteria:

**Stereo Functionality** The image-based approach is limited to generating directional cylindrical projections because the raster representations are created with a fixed base line for each camera



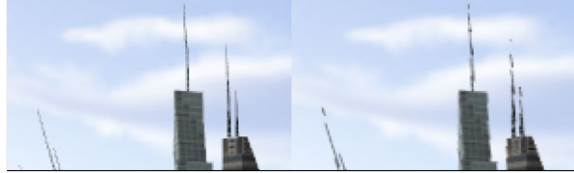
**Table 6.2:** Performance comparison between the image-based and geometry-based approach for generating stereo images pairs. The measurements (FPS) are taken for a  $180^\circ$  cylindrical projection with anaglyph passive stereo and a viewport resolution of  $1280 \times 1024$  pixels.

	IBA		GBA	
Triangles	8800 GTS	GTX 280	8800 GTS	GTX 280
34,596	20.66	42.55	31.32	52.15
236,276	6.04	24.51	12.42	35.77
540,655	2.58	9.40	3.49	9.11
3,210,162	0.41	2.83	0.93	4.14

orientation. consequently, the angle disparity is zero for views along the base-line and the user observes only depth disparity. In contrast thereto, the geometry-based approach is able to create full  $360^\circ$  omni-directional stereo cylindrical projections. Further, the GBA is capable of supporting all three camera modes described in Section 6.5 without artifacts. The IBA is limited to the parallel camera mode to avoid artifacts in the stereo pairs. Thus, the GBA has a clear advantage over the IBA.

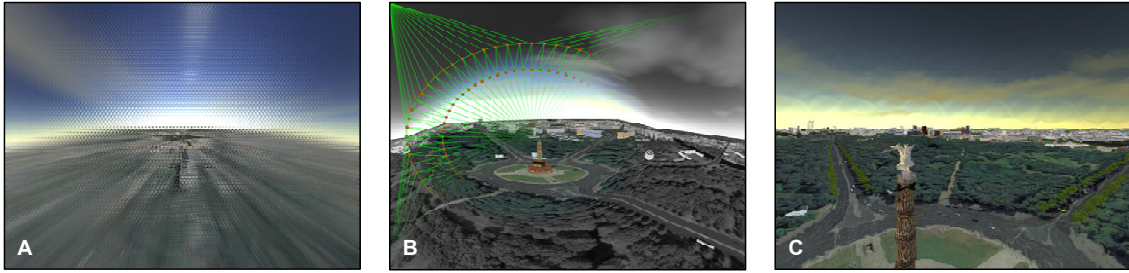
**Rendering Performance** Table 6.2 shows a comparison between IBA and GBA with respect to the number of input primitives. Both rendering techniques require only a single scene rendering pass for passive stereo viewing of a single cylindrical projection with a horizontal FOV of  $180^\circ$  and a vertical FOV of  $90^\circ$ . The IBA uses cube map face culling to render only required faces. The measurement shows that the GBA performs better than the IBA for low to medium model complexity. For a higher model complexity, both approaches obtain similar non-real-time performance but GBA performs better.

**Image Quality** The main drawback of the IBA is image-quality. In contrast to GBA, sampling artifacts are introduced while creating the projections. This is especially problematic for wire-frame renderings or NPR techniques such as hatching [200] or similar ones. Figure 6.16 shows the different image quality of both approaches. The images are taken from screen shots that are created using a cube map texture of  $2048^2$  texel on a target resolution of  $1600 \times 1200$  pixels. The quality of the displayed image also depends on the projector systems. However, the GBA is superior over IBA.



**Figure 6.16:** Comparison of the image quality between the geometry-based (left) and image-based approach (right).

**Memory Footprint** A further criteria considers the memory footprint for data related to the rendering technique, e.g., texture size and geometry. The footprint of the IBA can be considered constant. It depends on the texture resolution  $s$ , the precision per color channel  $b$ , the number of color channels  $c$ , and the number of raster layers  $l$ . The footprint can be approximated by:  $O_{IBA}(l, s, b, c) = 2 \cdot l \cdot c \cdot b \cdot s^2$  byte without mip-maps. This parametrization enables the user to balance the trade-off between image quality and space as well as runtime complexity. The memory footprint of the GBA is view-dependent and scales linearly with the number of input triangles  $t$ . Further, memory footprint depends on the average rate of primitive amplification  $r$  (for a  $180^\circ$  cylindrical projection  $r = 1.5 - 2$ ), and the size of each triangle in an intermediate data structure  $i = 16$ . The amount of additional memory can be approximated by:  $O_{GBA}(t, r, i) = t \cdot r \cdot i$ . Subsequently, the space complexity of the GBA is independent of rendering a single NPP or a stereo pair of NPPs. For a geometrical complex model (3,210,162 triangles) the additional memory requirement for a  $180^\circ$  cylindrical projection is  $O_{GBA} \approx 69$  MB. This corresponds to four RGBA raster layers with  $1024^2$  texels resolution. Thus, for a higher FOV:  $O_{GBA} < O_{IBA}$  is valid in any case.



**Figure 6.17:** Application examples of image-based deformations. *A:* Shows a compound-eye projection tiles screen with approximately 30,000 triangular projection tiles. *B:* A screen-aligned 2D magnification lens which is embedded within a fisheye projection with  $180^\circ$  FOV. *C:* Image-distortion based on a normal map.

**Implementation Complexity** Both approaches rely on programmable GPU (geometry shader functionality). The GBA uses transform feedback that is available on current graphics hardware [20]. IBA can be considered as easy to implement and integrate into existing rendering frameworks but needs to apply custom cube-map sampling. Since the IBA is independent of the scene geometry, the application of LOD approaches is unproblematic. Further, the application of more complex projections is easy to integrate. Furthermore, the multi-pass cube map texture creation of the image-based approach provides an implicit fall-back solution for older hardware.

## 6.8 Results and Discussion

The presented rendering technique enables a broad range of applications for the interactive visualization of large scale datasets, such as 3D virtual city and landscape models. It can be used to create standard non-planar projections (Fig. 6.5, Section 6.2) and its variations. Further, projection tiles facilitate different combinations of planar and non-planar projections (Fig. 6.7, Section 6.3) and thus, enable distortion-based focus+context visualization techniques for geovirtual environments. Triangulated projection tiles support the embedding of 2D screen aligned lenses into non-planar projections. Figure 6.17.B shows an example of a zooming lens. The lens position can be changed at runtime while the PTS is adapted per frame. To avoid possible under-sampling artifacts, a sufficient resolution of the cube map must be available. Figure 6.1 (Section 6.1) and Figure 6.17.A shows an application that utilize projection tiles to create a complex visualization that approximates a compound-eye like view on a 3D geovirtual environment. The concept also supports image-warping for planar and non-planar projections based on user defined normal maps. It enables post processing effects water on lenses or the simulation of shower door effects (Fig. 6.17.C).

To summarize, this chapter presents the concept of projection tile screens, a generalization of single-center projections and image distortions that is applicable in real-time, especially for complex geometric scenes. It enables the efficient creation and combination of planar as well as non-planar single center projections, 2D screen-aligned lenses with arbitrary shapes [237, 42, 15], image warping and image distortions. The concept is based on dynamic cube maps and programmable hardware features. It further presents a comparative performance evaluation for creating dynamic cube maps using single-pass and multi-pass approaches as well as a comparison between the image-based and geometry-based rendering technique. One drawback of the presented concept is a possible lack of image quality compared to geometry-based approaches. This is mainly caused by texture undersampling and oversampling artifacts that can results in blurred output images. To avoid such artifacts, the optimal resolution of the cube map texture must be adapted to the resolution of the viewport and the used projection or projection tile screen. For large horizontal and vertical FOV, a cube map resolution of  $1024^2$  pixels is sufficient within a viewport size ( $1600 \times 1200$ ). Further, the quality of the output images for projection tiles screen depend on the resolution of the PTS. If the resolution is too low, the feature interpolation can cause artifacts for tiles with acute angles. Possible future work can focus on possible applications of projection tiles screens, especially the interactive modification of projection tile screens using in-space authoring tools. Particularly interesting is the reproduction of other SCOP projections described in [83].

## Chapter 7

# Multi-perspective Views for 3D Geovirtual Environments

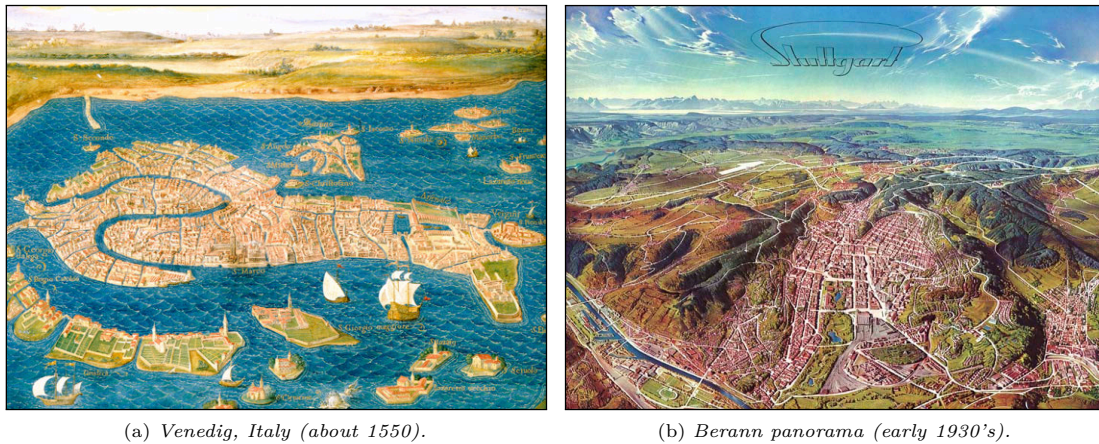
This chapter discusses how 3D geovirtual environments can be visualized using multi-perspective views [169, 182], based on view-dependent global deformations. Multi-perspective views or projections enable 3D visualization similar to panoramic maps by increasing overview and information density in the depictions of 3D GeoVEs. To make multi-perspective views an effective medium, they must adjust to the orientation of the virtual camera and be constrained by the geovirtual environment. However, changing multi-perspective camera configurations during interaction tasks typically require the user to manually adapt the global deformation — an error-prone, non-intuitive, and often time-consuming task [37]. The main contribution of this chapter comprises a concept for the automatic and view-dependent interpolation of multi-perspective viewing configurations. Concerning this, the approach includes the application of global deformations [7], based on 2D parametrized curves, as well as the option of assigning different geometric representations to sections of these curves. This enables the embedding of focus+context paradigms via the application of geometric variances. Further, a hardware-accelerated, multi-pass rendering technique is presented that enables real-time rendering of multi-perspective views.

## 7.1 Multi-perspective Views

Virtual spatial environments based on 3D landscape and city models are common tools for an increasing number of commercial and scientific applications for planning, simulation, and visualization tasks. Whereas the efficient rendering based on level-of-detail techniques and multi-resolution models represent a key requirement, the effective presentation of the environment and its contents (e.g., by providing detailed views for important areas while giving a coarse overview of their spatial context) is essential as well. While a single-perspective view depicts a scene from a single view-point, "a multi-perspective rendering combines what is seen from several viewpoints into a single image." [277]. In this way occlusions become resolvable, scales at which objects are depicted are adjustable, and spatial context information can be included in a single image.

With these techniques, multi-perspective views can visually emphasize or clarify an area-of-interest while maintaining or extending the depiction of its surrounding area, achieving an effective information transfer [142]. Furthermore, they utilize the available screen real estate (the amount of space available on a display for an application to provide output) to a higher degree than standard 3D perspectives [136]. Their characteristics make multi-perspective views a tool for focus+context visualization.

**Multi-Perspective Views for Maps** Multi-perspective views have been developed particularly in landscape depiction and Cartography. Chinese landscape painters used multi-perspective views in the 11<sup>th</sup> century already [258]. Another example, a 360° panorama view of the London skyline consisting of six separate paintings, was created in the late 18<sup>th</sup> century. The incorporation of cartographic information yields panorama maps. Figure 7.1(a) shows an early example of Venedig, Italy (about 1550). H.C. Berann, an Austrian artist and panorama maker, pioneered one particular kind of panorama map. Beginning in the early 1930's he created a deformation and painting style,



**Figure 7.1:** *Historic examples of hand-crafted multi-perspective views.*

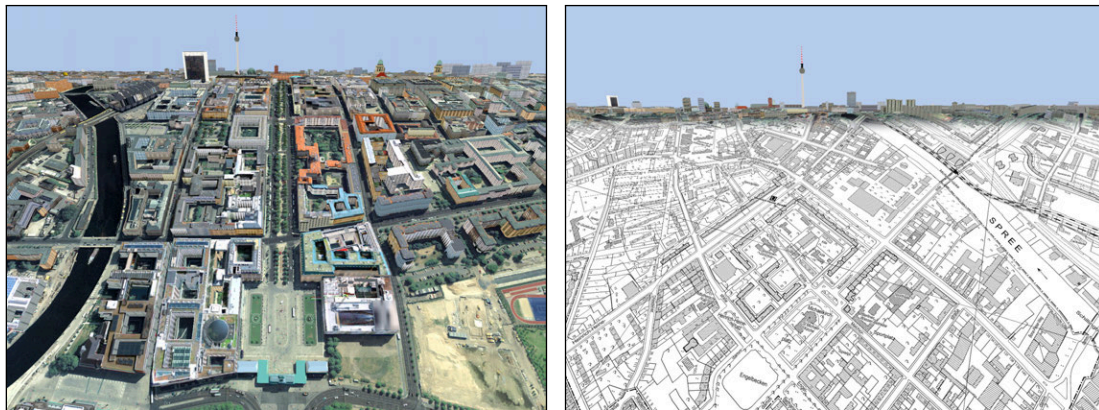
known as "Berann panorama" (Fig. 7.1(b)), which became the de-facto standard for tourist maps in recreational areas. This style seamlessly combines a highly detailed image of the area-of-interest with a depiction of the horizon including major landmarks. The area-of-interest is shown in the foreground from a high viewpoint, whereas the horizon is shown in the background from a low perspective. The environment is depicted with "natural realism" [192] and key information (e.g., trails or slopes) is superimposed in an abstracted, illustrated fashion. As a result of the high viewpoint the foreground shows key information top-down, i.e., free from obstructions and clearly visible. At the same time, the map user can easily orient the map using the horizon, which is visible due to the changed perspective, as reference without the need for a compass. For these reasons, panorama maps are useful specifically to unskilled map readers.

In general, the creation of panorama maps is time consuming and requires a skilled artist. It includes proper viewpoint selection, partial landscape generalization, identification of landmarks, their integration into the map with recognizable shapes, and a smooth transition between the foreground and background perspective [192]. Even with the support of digital tools and digital 3D geodata, panorama creation still remains a tedious manual process [201]. Despite their effectiveness, panorama maps are rarely created, and the creation techniques can hardly be transferred to interactive systems where the user manipulates the viewpoint.

**Multi-Perspective Views for Spatial 3D Environments** Multi-perspective views can be used to visualize 3D landscape models, e.g., mountainous regions with the mountain peaks providing a distinctly recognizable background for orientation purposes. Similarly, they can visualize 3D city models, using the skyline of the city as background. In today's applications, interactive visualization is required to support the user in exploring and analyzing the geovirtual 3D environment. With respect to the usability of such applications, the navigation and orientation aids represent key issues because users frequently "get lost in space" without guidance [37]. Here, the inclusion of a fixed horizon or skyline similar to a Berann panorama offers an additional orientation cue in the sense of a focus+context visualization. This chapter focuses on the projection as major tool for orientation and neglect artistic aspects such as landmark depiction and selective generalization.

Computer graphics knows three approaches to achieve a panorama effect: multi-perspective images, deformations, and reflections on non-planar surfaces [258]. Multi-perspective images either use non-linear, non-uniform projections or combine multiple images from different viewpoints to create the final rendering. Deformations distort the landscape before rendering the final image using a standard projection, which implies recomputation of all geometric data for every image. Finally, reflections on non-planar surfaces use standard projections showing an intermediate object that in turn reflects the landscape.

**Interactive Multi-Perspective Views** Techniques implementing multi-perspective views can be classified as multi-pass or single-pass. Multi-pass techniques create several intermediate images that

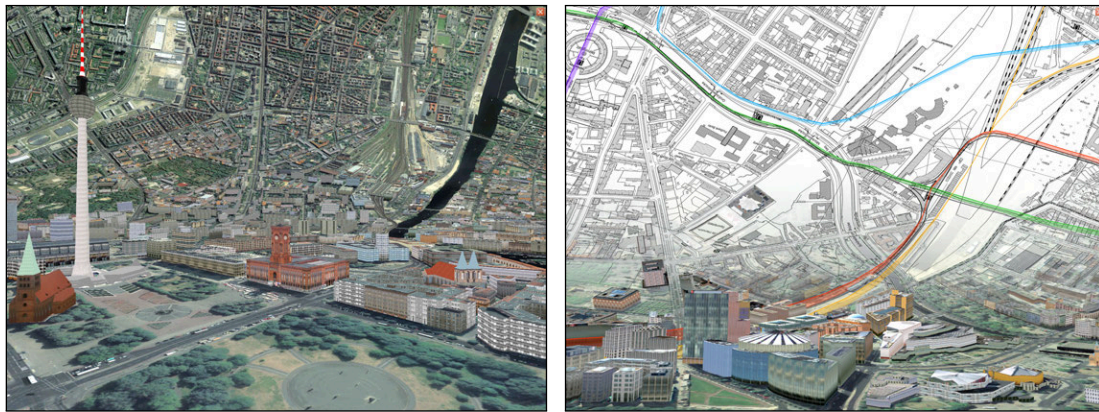
(a) *Progressive perspective with homogeneous styling.*(b) *Progressive perspective combining aerial and topological map data.***Figure 7.2:** *Examples of viewing deformation that implement progressive perspectives.*

are blended in a final compositing step. Each intermediate image requires separate data processing, which is rather expensive when it comes to complex spatial 3D environments. Specifically, out-of-core algorithms can incur additional penalties because rendering of intermediate images often significantly reduces caching efficiency. Additionally, image quality suffers due to resampling in the compositing step. Single-pass techniques do not exhibit these disadvantages, yet they require customization of the rendering process. This chapter demonstrates a technique that implements a dynamic global deformation [7] and shifts this task to the GPU. This approach exploits best the optimization of current graphics hardware for standard projections both in terms of image quality and speed.

## 7.2 Effective Presentation of 3D Geovirtual Environments

Multi-perspective views facilitate the implementation of effective presentation of spatial 3D environments. They can add valuable cues by seamlessly integrating multiple perspectives in the resulting images and, therefore, make efficient use of the image space. In the following, two main deformations are presented that partially implement multi-perspective views [136]. In general, these deformations need to ensure the user's location awareness during navigation and interaction. Even experienced users get disoriented if the current perspective does not contain sufficient points of reference or if the image sequence does not provide spatio-temporal coherence. For these reasons, all presented deformations provide a seamless combination of different views in a single image and achieve interactive frame rates.

**Progressive Perspective** Similar to Berann's panorama maps, this deformation is based on a depiction of the area-of-interest using a bird's eye view, which would not permit a visible sky, a view of the horizon and sky, and a smooth transition between both perspectives. As a result, the landscape appears to be separated into two planar sections connected by a curved transition zone with the focus lying on the bird's eye view part in the foreground (the lower image part, Fig. 7.2). Nevertheless, the area-of-interest is not strictly separated from the transition zone but often reaches into the curved section. For a painted panorama map, the map designer decides on relevant parameters such as the view points and the transition in between. In an interactive application the user can move the virtual camera. The fixed horizon provides strong temporal coherence and eases orientation tracking during navigation, whereas the ever-changing shape of the landscape does not lead to distraction. In general, painted panoramas exhibit a horizontal horizon. In contrast, an interactive application can permit rolling of the camera. In this case, the horizon should provide feedback about the roll angle. In the following descriptions, no camera rolling is assumed.



(a) *Degressive perspective with homogeneous styling.*

(b) *Degressive perspective combining aerial and topological map data.*

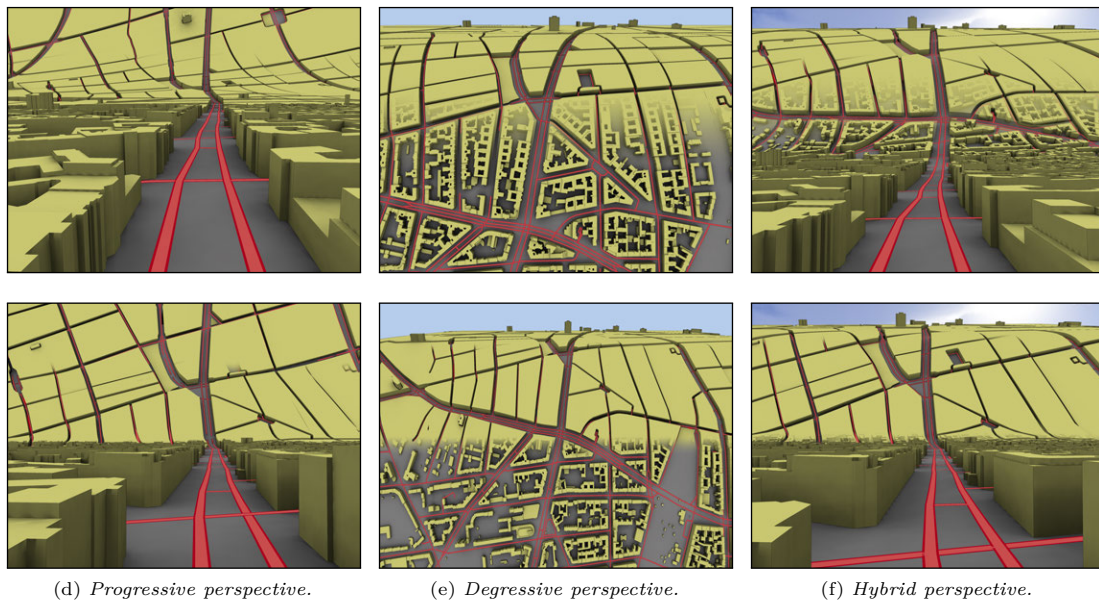
**Figure 7.3:** *Examples of viewing deformations that implement degressive perspectives.*

**Degressive Perspective** The bird's eye view deformation supports answering questions such as "Which direction am I looking to?" without the need for a virtual compass. For pedestrian's views, which occur in walk-through scenarios, the question changes to "Where am I going to?", e.g., if users want to look ahead the path along they are currently walking. Due to the low viewing angle, however, users can generally not obtain an effective overview without changing the perspective or navigation mode because large parts of the 3D geovirtual environment are occluded. To counter this effect, the viewing deformation bends upwards distant parts of the virtual 3D city model (Fig. 7.3). Compared to the technique proposed in [258], which deforms a reference plane to fit the inside of a cylinder, the presented deformation has the advantage of using a planar and, hence, clear and undistorted view of distant regions in the background. In terms of focus+context, the prominent sky in a degressive perspective, which provides only little information, is replaced by a top view of the region ahead, resulting in a more efficient use of screen space.

In addition, hybrid variants [182] combine three aspect of progressive and degressive perspectives: detailed views in the foreground, a bird's eye overview of the distant parts, and the view of the horizon and sky (Fig. 7.4(c) and 7.4(f)).

**Graphical Representation of Focus and Context** The deformations smoothly and seamlessly combine focus and context areas. However, the user might loose distinction between geometrically correct information in the focus area and deformed information in the context during navigation. This might lead to misinterpretations, lost of orientation, or erroneous navigation [280]. Specifically, the degressive perspective, permitting views without visible focus area and, hence, without navigation reference, is prone to such effects. Solutions require visual cues (e.g., iconic navigation aids), distinct rendering styles (style variances) for focus and context [135], as well as combinations of different levels-of-abstraction [93] (geometry variances).

Besides the more effective use of screen space, in focus+context visualization the two constituents can serve different purposes and thus are to display different information dimensions beyond change of rendering style [142, 241]. Whereas the focus gives core information, the context shows supporting information. Panorama maps use this principle by adding thematic information such as trails to the focus area while the landscape depiction style is constant for the whole image. The presented approaches showing a map with 3D landmarks in the focus area, while the context area remains a complete and photo-realistic depiction since the skyline is required to be recognizable. The degressive perspective permits displaying more important information in the context. In fact, the focus area is limited to serve as navigation reference and location marker within the spatial 3D environment whereas the context generally receives the larger screen space and exhibits less occlusion. According to this observation, the sample visualization (compare Fig. 7.3(b)) shows



**Figure 7.4:** Exemplary results of a visualization system that enables the view-dependent interpolation between different multi-perspective views combining different generalization levels of a virtual 3D city model of Berlin. The top row show smooth perspective transitions with wide transition zones, while the bottom row shows hard transition with relatively small transition zones. The latter uses B-Spline curves with six control points to enable hard transitions between three planar regions.

a photo-realistic style in the focus area and a topographic map in the context area for visual distinction.

However, a major drawback of 3D GeoVEs are multiple geometric scales [136] that are caused by the perspective projection. Consequently, the depiction of most objects occupy only a small image space (e.g., only one pixel) and cannot be distinguished by a viewer anymore (pixel noise). To overcome this problem in the domain of paper maps, cartographers apply generalization techniques to minimize visual complexity and to improve comprehension [110]. A similar concept is used for the presented multi-perspective visualizations. In addition to a photo-realistic style, a map-based style is applied to regions of small scales. The user can define multiple geometric representations (e.g., obtained from cell-based generalization [95]), to sections of the deformation (Fig. 7.4). These explicit geometric representations enable more design freedom than automatically derived style variations such as in [169]. Jobst and Döllner [136] suggest to subdivide the visualization into distinct zones where a constant scaling and thus a constant generalization is applied on a per-zone basis (Fig. 7.4, bottom row).

### 7.3 View-dependent Multi-perspective Views

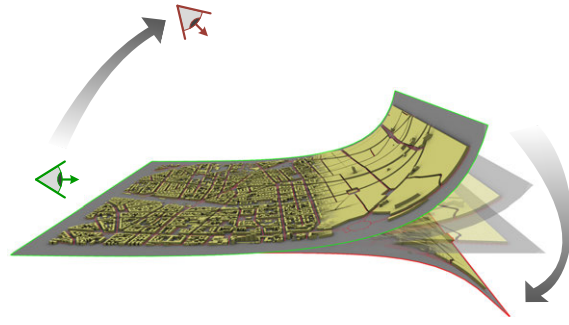
In the context of interactive global deformations and multi-perspective views, the presented perspectives most effective for specific settings of a virtual camera, i.e., the virtual camera must be near ground (pedestrian view) or at a certain height (birds-eye view), in order to exploit their full potential. In a 3D GeoVE, the user usually wants to interact and navigate freely. This would require the manual adaptation of the visualization parameters during interaction and navigation. In general, this task is complex, error-prone, and time-consuming. This section provides a concept that renders suitable multi-perspective images for a virtual camera setting using automatic view-dependent interpolation of global deformations (Fig. 7.4). The presented approach comprises two main steps: First, during rigging of visualization presets, the user prepares discrete visualization presets. A preset includes a deformation curve, the assignment of geometric representations to curve sections (tagging), and the definition of a viewing angle interval, in which the preset is valid. Second, during real-time rendering, these presets are interpolated based on the parameters of the virtual camera.

**Preliminaries and Deformation Curves** For the multi-perspective visualization, the geometric representation of a 3D GeoVE is assumed to be approximated by a 3D reference plane  $R = (N, O) \in \mathbb{R}^3 \times \mathbb{R}^3$  defined by a normal vector  $N$  and a position vector  $O$ . Thus, and because of the isotropy of the global deformation variants described in this section, a view setting for a virtual camera can be described by an viewing angle  $\varphi = \cos(90 - C_D \cdot N)$  (Fig. 7.6). Möser et al. [182] generalize the concept introduced in [169] by using Hermite curves for the parametrization of global deformations. These can be easily manipulated by the user. However, the application of standard parametrized curves additional geometric distortions that can be compensated these by using an arc-length parametrization [193]. In [136] it is argued that a smaller transition zone and linear segments would benefit the comprehension of such a visualization. This specific configuration is hard to implement using a single Hermite curve, but can be easily achieved using B-Splines curves with six control points, by setting two consecutive control points to the same position. The presented approach uses cubic B-Splines curves[208] with four or six control points, to implement progressive, degressive, or hybrid perspectives.

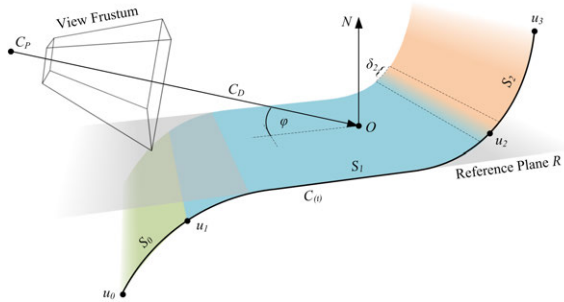
To implement deformations of a 3D scene, a global space deformation [7] based on B-Spline curves  $C(t) = \sum_{i=1}^{n+1} B_i \cdot N_{i,k}(t)$  is applied is applied on a per-vertex basis. The parametrization comprises control vectors  $B_i$  of the  $n - 1$  control points and the normalized B-Spline basis functions  $N_{i,k}(t)$ . Assuming that  $B_i$  and  $N_{i,k}(t)$  are fixed for a specific B-Spline, the resulting position vector  $C(t)$  only depends on the parameter  $t$ . For the deformation, a mapping of an input vertex  $V = (x, y, z, w) \in \mathbb{R}^4$  to  $t$  is required. Given this,  $V$  is first aligned along the viewing direction  $C_D$  of the virtual camera by computing  $V' = V \cdot \mathbf{R}_A$ , whereas  $\mathbf{R}_A$  rotates  $V$  with  $\varphi$  degrees around the axis defined by  $O$  and  $N$ . This is necessary, because the mapping of the vertex's depth to  $t$  must be independent of the camera orientation. To compute  $t$ , the depth of  $V'$  is linearized w.r.t. user defined scalars for the start  $s$  and end  $e$  of the curve in camera space. Finally, the deformed vertex  $V''$  is computed as follows:

$$V'' = \begin{cases} V' \cdot \mathbf{M}_S & t < 0 \\ V' \cdot \mathbf{M}_E & t > L \\ V' \cdot \mathbf{M}_{C(t)} & \text{otherwise} \end{cases} \quad t = \frac{z'_V - s}{e - s} \cdot \frac{1}{L}$$

The the curve evaluation, the parameter  $t$  must be in a well defined interval (e.g.,  $[0, 1]$ ). Therefore, it is normalized according to the curve length  $L$ . The deformation matrix  $\mathbf{M}_{C(t)}$  consists of two transformation:  $\mathbf{T}_{C(t)}$  translates the vertex according to its position on the curve and  $\mathbf{D}_{C(t)}$ ; translates the vertex along the normal of the curve. According to  $t$ , the position vector  $C(t)$  as well as the corresponding tangent  $C'(t)$  are computed. Since the input vertex must be translated along the tangent orthogonal to the bi-normal  $B_x$  and tangent  $C'(t)$ , the normal vector  $N(t) = C'(t) \times B_x$  is computed. Both vectors are used to setup the translations  $\mathbf{T}_{C(t)}$  and  $\mathbf{D}_{C(t)}$  to define  $\mathbf{M}_{C(t)}$  as follows:



**Figure 7.5:** *Conceptual sketch for the view-dependent interpolation of global deformations with different geometric representations based on the viewing angle of the virtual camera.*



**Figure 7.6:** *Exemplary parametrization of a deformation curve preset using four tag points ( $u_i$ ).*



$$\mathbf{M}_{C(t)} = \mathbf{D}_{C(t)} \cdot \mathbf{T}_{C(t)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & C(t)_y + d \cdot N(t)_y \\ 0 & 0 & 1 & C(t)_x + d \cdot N(t)_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here,  $d$  denotes the distance of  $V'$  to its projection onto  $R$ . To handle the cases of  $t \notin [0, 1]$ , the deformation matrices  $\mathbf{M}_S$  and  $\mathbf{M}_E$  are applied accordingly to transform  $V'$  on an extension plane at the beginning or the end of the curve-based deformation.

**Arc-Length Parametrization of B-Spline Curves** Depending on the distribution of the control vertices and the knot vector of a B-Spline curve, a sampling with equidistant values ( $t_1$ ,  $t_2$ , and  $t_3$ ) may not yield to an equidistant distribution of points ( $P(t_1)$ ,  $P(t_2)$ , and  $P(t_3)$ ), because a B-Spline curve is not arc-length preserving. This will lead to scaling errors introduced by an straining or stretching of the geometric representation.

To achieve a deformation behavior with minimal scaling artifacts, the curves must be re-parametrized. The approaches presented in [213] and [195] are not suited for a real-time rendering techniques, because they either globally distribute the scaling error or are computational expensive. Rogers approach [213] suggest a modification of the knot vector, to span equidistant ranges and as a result the scaling error would be uniformly distributed. Peterson [195] presents a technique that approximates a given B-Spline curve  $C(t)$  using a second arc-length-based B-Spline curve  $P(t)$ . it is not suited for real-time rendering, because the approximation must be recomputed every time the control points changing. Further, an computational overhead is introduced, because two curves or a curve of higher degree must be evaluated.

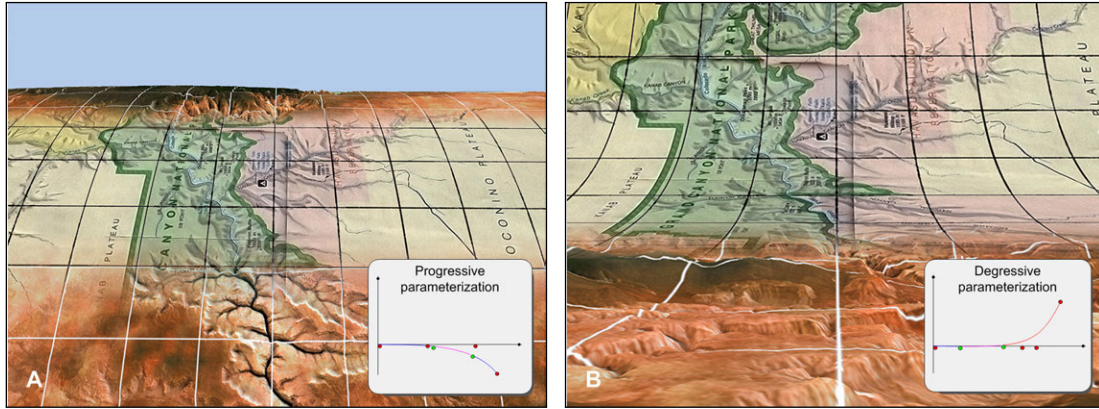
Instead, the parameter  $t$  is re-parametrize similar to the method described in [193]. The B-Spline curve is sampled in equidistant intervals and the arc-length of these segments are computed. Based on the sampled length  $L$  and the according parameter  $t$ , the arc-length preserving parameter  $t'$  is computed by linear interpolation and stored in a look-up table. First, a look-up table between  $t$  and the arc-length  $L$  is created by subdividing the B-Spline curve in  $p$  line segments. This is done by sampling  $C(t)$  in equidistant intervals  $t_i - t_{i-1} = 1/p$ . The arc length of the curve for a given  $t_j$  is computed by summing up the length of the lines approximating the curve segment defined by the interval  $[t_0, t_j]$ . Then,  $L$  is sampled in equidistant intervals  $L/p$ , and the prior established table is searched to find the interval  $[t_i, t_{i+1}]$ , such that  $L_{t_i} \leq L < L_{t_{i+1}}$ . The curve parameter  $t'$  corresponding to  $L$  is computed by linear interpolation between  $t_i$  and  $t_{i+1}$ :

$$t' = t_i + \frac{L - L_{t_i}}{L_{t_{i+1}} - L_{t_i}} \cdot (t_{i+1} - t_i)$$

**Visualization Presets and Tagging of Deformation Curves** A *visualization preset* denotes a single configuration of a multi-perspective view (e.g., degressive or progressive). A preset  $P$  consists of the following components:  $P = (C(t), \mathcal{T}, \mathcal{G}, \varphi, \tau, s, e, a, b)$ . The set of all presets is denoted as  $\mathcal{P}$ , with  $|\mathcal{P}| = m$ . In addition to the B-Spline deformation curve  $C(t)$ , a preset contains an ordered list of tag points  $\mathcal{T}$ , a list of geometric representations  $\mathcal{G}$ , and the following scalar parameters (Fig. 7.6): a camera angle ( $\varphi$ ), an angle interval ( $\tau$ ), the start ( $s$ ) and end ( $e$ ) of the deformation in eye-space, and finally  $a, b \in [0, 1]$  defining the start and end of the geometry interpolation.

The approach enables the association of different geometric representations to curve sections to implement geometry variances for focus and context areas. This is useful to increase or decrease the visual complexity of part of the virtual 3D model. In [169], this was implied by blending between different type of textures within the transition zone and by omitting unimportant buildings. This idea can be extended by blending between 3D geometry assigned to consecutive sections of a deformation curve. In the presented examples (Fig. 7.4) different levels-of-abstraction (LoA) are used, which are automatically derived from the virtual 3D city model of Berlin [95].

Therefore, a deformation curve  $C(t)$  can be partitioned into a number  $l \geq 2$  of consecutive *styling sections*  $S_i = (T_i, T_{i+1}, G)$  as elements of a global set of styling sections  $\mathcal{S}$ . Here,  $i = 0, \dots, l - 1$  represents an index into the list of *tag-points*  $\mathcal{T} = T_0, \dots, T_l$  assigned to every preset  $P$ . The



(a) Example of a tagged progressive visualization. The focus area shows a topological map embedded in an aerial image.

(b) Example of a tagged degressive visualization. The focus area shows the aerial image while the context area shows a topological map.

**Figure 7.7:** Styling section of a deformation curve with different geometric representations of the Grand Canyon. The inset shows the associated tag point and sections of the curve: The control points are depicted in red and the tag points are depicted in green.

geometric representation for a styling section is denoted as  $G$ . A single tag point is defined by  $T_i = (u, \delta) \in \mathcal{T}$ , whereas  $i = 0, \dots, l$  and  $u, \delta \in [0, 1]$ . The parameter  $u$  controls the position of the tag point on the curve. The parameter  $\delta$  describes the length of the transition zone between two consecutive sections and is used for blending between consecutive sections. Further, fixed start and end tag points are assumed at the curve start ( $T_0 = (0, 0)$ ) and at the curve end ( $T_l = (1, 0)$ ). Figure 7.7 shows different variants of a virtual 3D landscape model of the Grand Canyon and the associated curve preset (inset).

**View-Dependent Curve Interpolation** The view-dependent curve interpolation based on the camera angle  $\varphi$  comprises of two steps: (1) the *preset selection* and (2) the *preset interpolation*. Given the viewing angle  $\varphi_a$  and the set of all presets  $\mathcal{P}$ , a selection function  $s(\mathcal{P}, \varphi_a) = (P_S, P_T)$  delivers two presets as follows:

$$s(\mathcal{P}, \varphi_a) = (P_S, P_T) = \begin{cases} (P_i, P_{i+1}) & \varphi_a \geq \varphi_i \wedge \varphi_a < \varphi_{i+1} \\ (P_1, P_2) & \varphi_a \leq \varphi_1 \\ (P_{m-1}, P_m) & \varphi_a > \varphi_m \end{cases}$$

This requires an ascending ordering of  $\mathcal{P}$  by  $\varphi$  performed at the end of the rigging process. Given the viewing angle  $\varphi_a$  and the resulting two presets  $P_S$  and  $P_T$ , a weighting factor  $\sigma \in [0, 1]$  is computed as follows:  $\sigma = \text{clamp}((\varphi_a - \varphi_S)/(\varphi_T - \varphi_S), 0, 1)$ . Given this factor, the interpolation  $P_I = p(P_S, P_T, \sigma)$  between the source preset  $P_S$  and target preset  $P_T$ , is performed by a linear interpolation of all control points:  $B_{i,I} = B_{i,P_S} + \sigma \cdot (B_{i,P_T} - B_{i,P_S})$  and the respective tag points:  $T_{i,I} = T_{i,P_S} + \sigma \cdot (T_{i,P_T} - T_{i,P_S})$ . In addition thereto, the geometric representations must also be interpolated. First, the geometric representations of  $P_S$  and  $P_T$  are rendered into two texture-arrays that are blended according a factor  $\beta \in [0, 1] = \text{clamp}((\sigma - a_{P_S})/(b_{P_S} - a_{P_S}), 0, 1)$ . The interval  $[a_{P_S}, b_{P_S}]$  defines in which section of the curve the specific geometric representations is visible.

## 7.4 Multi-scale Rendering

To enable focus+context visualization with geometric variances for focus and context areas, multi-scale rendering applied [122] that uses different geometric representations of a 3DGeoVE. For example, the rendering of Figure 7.7 uses two virtual 3D landscape models of the Grand Canyon, and Figure 7.4 is composed of three levels-of-abstraction of the virtual 3D city model of Berlin, which are generated using an cell-based approach [93]. Multi-scale rendering requires the visualization prototype to be implemented using multi-pass rendering in combination with render-to-texture. During multi-pass rendering, for each styling section the global space deformation is applied using

vertex shader. Each deformed geometric representation is written to an off-screen buffer, using RTT. Finally the textures are composed. An overview of the process is depicted in Figure 7.8, details are given in Algorithm 5.

---

**Algorithm 5** Scene Rendering
 

---

```

1:  $\varphi = \text{computeAngle}(R, C)$  [Determine angle]
2:  $(P_S, P_T) = s(\mathcal{P}, \varphi)$  [Select start and target preset]
3:  $P_I = p(P_S, P_T, \varphi)$  [Compute interpolated preset]
4: for all  $T_i \in \mathcal{T} \in P_I$  do
5:    $\text{setup}(P_I, T_i)$  [Setup rendering for tag point]
6:    $\text{Texture}_i \leftarrow \text{renderToTexture}(S_{S_i}, P_S)$  [Render styling section of source preset]
7:    $\text{Textures} \cup \text{Texture}_i$  [Add source preset rendering results]
8: end for
9:  $l = |\text{Textures}|$  [Number of rendered tag points]
10: for all  $T_i \in \mathcal{T} \in P_I$  do
11:    $\text{setup}(P_I, T_i)$  [Setup rendering for tag point]
12:    $\text{Texture}_{i+(l-1)} \leftarrow \text{renderToTexture}(S_{T_i}, P_T)$  [Render styling section of target preset]
13:    $\text{Textures} \cup \text{Texture}_{i+(l-1)}$  [Add target preset rendering results]
14: end for
15: for  $\text{Texture}_i \in \text{Textures}, i = 0, \dots, l-1$  do
16:    $\text{applyStylization}(\text{Texture}_i)$  [Apply stylization to source preset render results]
17:    $\text{applyStylization}(\text{Texture}_{i+(l-1)})$  [Apply stylization to target preset render results]
18:    $\text{blend}(\text{Texture}_i, \text{Texture}_{i+(l-1)})$  [Blend target into source texture]
19:    $\text{blendToFrameBuffer}(\text{Texture}_i)$  [Compositing of source textures into frame buffer]
20: end for

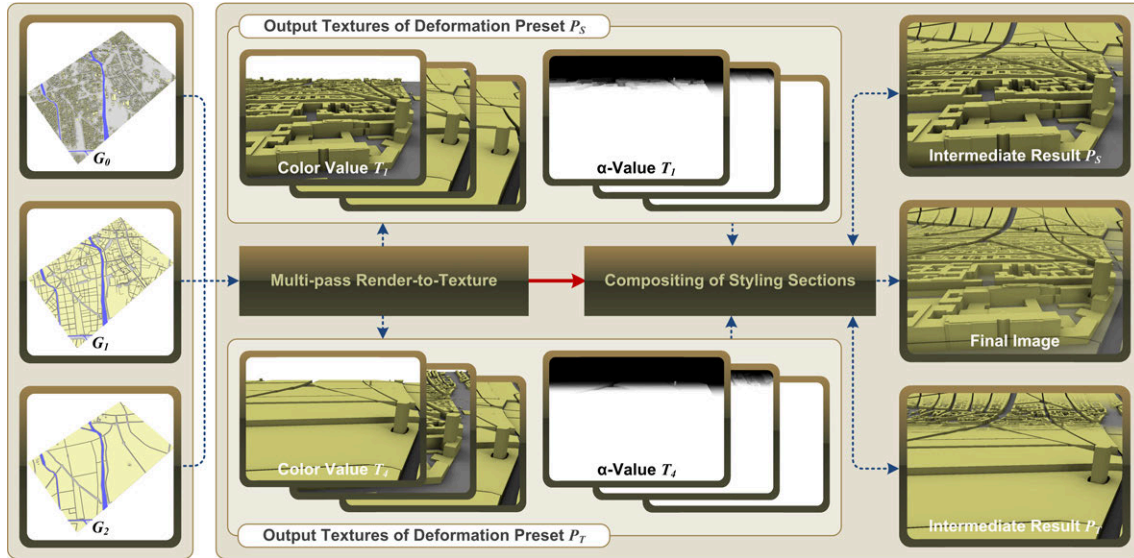
```

---

**Computing Global Deformations using Vertex Shader** As described in the previous section, the deformation is applied in a two step process. First, an incoming vertex  $V$  is aligned parallel to the camera viewing angle  $\varphi_a$ . To achieve this, the viewing angle is recomputed on a per frame basis and the resulting rotation matrix  $\mathbf{R}_A$  is passed to the vertex shader. Multiplying  $V$  with  $\mathbf{R}_A$  yields  $V'$ , which is afterwards projected onto the reference plane  $R$ . Its initial distance  $d$  to the reference plane is stored. Second, the control point and tangent vector of the B-Spline curve is evaluated on a per-vertex basis, to setup  $\mathbf{M}_{C(t)}$ . One possibility is to evaluate the B-Spline in the vertex shader on a per-vertex basis. This implies, that the specific formulas to evaluate the parametric curves are known at shader compilation time and are fixed in the vertex shader source code. A change of the parametric curve would lead to a change of the shader code. Instead, the position and tangent vectors of the B-Spline curve are computed off-line on the CPU, prior to rendering. Thus, the B-Spline curve must only be evaluated on a per-frame instead of per-vertex basis.

The look-up table for the arc-length parametrized B-Spline is encoded using a 32 bit luminance texture and passed to a vertex shader implementing the deformation. The texture look-up is performed using the parameter  $t$ , yielding the arc-length corrected values. The quality of the arc-length approximation depends on the number of precomputed samples. The bi-linear interpolation during texture sampling and filtering provides a second parameter interpolation. Experiments have shown that 2000 samples are sufficient for an arc-length preserving parametrization.

During the arc-length parametrization, the corrected position and tangent vectors of the B-Spline curve are pre-computed on the CPU. These values are stored in a 1D texture, which represents the look-up table accessed in the vertex shader. The 2D vectors  $C(t)$  and  $C'(t)$  are encoded in a 32Bit RGBA texture. The look-up table must be recomputed every time the setup of the parametric curve (e.g., the number or the position of the control points) changes. Thus, for a static curve setup (e.g., the user does not change the viewing angle of the virtual camera) no overhead is introduced. During view-dependent preset interpolation, the look-up table is updated once per frame.



**Figure 7.8:** Exemplary compositing of different styling sections into a final rendering. The middle section shows the output textures of the presets  $P_S$  and  $P_T$ , which are each blended into two intermediate results (right side). Finally, the blending of the intermediate results produces the final image.

**Compositing of Styling Sections** As shown in Figure 7.8 the composition consists of two steps: (1) Multipass RTT and (2) image-based composition in the fragment shader. To compose the potentially different geometric representations of  $P_S$  and  $P_T$ , an image-based compositing method is applied. Therefore, every styling section of the presets is rendered into separate textures using RTT. Each texture contains RGBA information at viewport resolution. During rendering, a fragment shader adjust the  $\alpha$ -value of a fragment according to the styling section boundaries defined by  $T_i$  and  $T_{i+1}$ , so that:

$$\alpha = \begin{cases} 1 & u_{T_i} + \delta_{T_i} \leq t \leq u_{T_{i+1}} - \delta_{T_{i+1}} \\ \frac{(u_{T_{i+1}} + \delta_{T_{i+1}}) - t}{2 \cdot \delta_{T_{i+1}}} & u_{T_{i+1}} - \delta_{T_{i+1}} < t \leq u_{T_{i+1}} + \delta_{T_{i+1}} \\ 0 & \text{otherwise} \end{cases}$$

After RTT is performed, the  $2 \cdot (l - 1)$  textures (i.e.,  $l - 1$  textures per preset) are alpha-blended [3] into the frame buffer. The blending is performed as follows: The first  $l - 1$  textures encoding  $P_S$ , are blended based on  $\alpha$ , starting with the most distant styling section (back to front compositing). The resulting fragment color is temporally stored. This procedure is repeated for the styling sections of  $P_E$ . Finally, the two color values are blended based on  $\beta$ . In addition thereto, stylization algorithms are applied. In a preprocessing step, light maps (ambient occlusion term only [61]) are computed for the complete model. At runtime during the compositing step, edge-detection based on the normal and depth information of a fragment [186] are computed. Further, unsharp-masking the depth buffer [170] is computed to improve the perception of complex scenes by introducing additional depth cues.

**Performance Evaluation** The performance tests are conducted using a NVIDIA GeForce GTX 285 GPU with 2048 MB video RAM on a Intel Xeon CPU with 2.33 GHz and 3 GB of main memory. The tests are performed at a viewport resolution of  $1600 \times 1200$  pixels. Table 7.1 shows the results of the performance evaluation. All models are rendered using in-core rendering techniques with  $8 \times$  anti-aliasing.

**Table 7.1:** Comparative performance evaluation for different test scenes (in frames-per-second). The abbreviation LoA  $i$  names the configuration of a preset using different geometric representation  $G_i$  (see Fig. 7.8).

Preset	#Vertex	#Face	FPS
LoA 0/1	1,219,884	477,437	21
LoA 1/2	380,689	364,500	39
LoA 0/1/2	1,443,895	720,587	17

The performance mainly depends on the number of tag sections, hence the number of required rendering passes, and the geometrical complexity of the scenes attached to them. Due to the heavy usage of render-to-texture in the compositing steps, the performance also depends on the size of viewport. Here, the additional amount of graphics memory  $O(l)$  required for a number of global styling section  $l$  can be estimated by:  $O(l) = 2 \cdot l \cdot w \cdot h \cdot 4 \cdot p$  Bytes. The prototypical implementation uses a precision  $p = 2$  Byte per channel, which is sufficient for post-processing stylization.

## 7.5 Summary and Discussion

This chapter presents a concept and interactive rendering technique for view-dependent global deformations of spatial 3D environments. It is inspired by the well-known panorama maps and aim to increase the effectiveness of interactive applications by using the principle of focus+context visualization. It presents an approach for a view-dependent parametrization and real-time interpolation of global deformations based on B-Spline curves. The application of parametrized curves offers the possibility to customize or extend traditional perspectives to cartography-oriented visualization (e.g., by using degressive or progressive perspectives), in an intuitive and flexible way. Further, the definition of camera-dependent presets and their automatic interpolation overcomes the restriction of existing multi-perspective visualization. In addition thereto, a concept for assigning different geometric representations to specific sections of a B-Spline curve is provided, which offers a high degree of design flexibility. This features enables the application of focus+context visualization principles to multi-perspective views by supporting the usage of style and geometry variances. It permits the seamless combination of different graphical representations for focus and context areas. Furthermore, a prototypical implementation is presented that enables hardware-accelerated, real-time image synthesis as discussed in the performance evaluation.

However, the presented approach implies some conceptual limitations. First, the number of control and tag points must be the same for each preset in a visualization. Second, the visual quality of the approach relies on a sufficient vertex density of the geometric representations. To counter this, the functionality of the tessellation shader unit can be applied to ensure this property for general scene geometry. This would also improve the visual quality in the transition zone. Furthermore, the rendering concept can be optimized w.r.t. the required off-screen rendering passes. The current implementation requires an off-screen rendering pass for each styling sections. If two or more styling sections contain the same geometric representation they can be treated as one single styling section. This reduces the number of rendering passes. The same applies for the geometry interpolation. The number of vertices can be further reduced by a culling algorithm based on the boundaries of the styling sections.

Since the arc-length parametrization and shader setup does only depend on parametric curves, other curve types than B-Splines could be evaluated. The described concept here is not limited to view-dependent interpolation. One can think of the automatic derivation of presents based on semantical or thematic properties of a 3D geovirtual environment, e.g., to ensure the visibility of certain landmarks or features. While this contribution describes the underlying technology, user studies about the effectiveness and/or expressiveness of the visualization approach, different rendering style combinations, and navigation in a deformed 3D landscape model remain future work. Parts of the implementation an concept are transferred to mobile devices and evaluated w.r.t. to the perception of navigation routes (Section 9.3).



## Chapter 8

# Highlighting Techniques for 3D Geovirtual Environments

Highlighting is a fundamental visualization principle and can be considered as the most naive variant of a focus+context technique. It describes the method of applying visual effects to objects-of-interest in order to make them more visible or prominent to a user or that they can be considered separately from the whole to direct attention. Based on object-level separation methods, this section presents object-based and image-based rendering techniques for the highlighting of multiple scene objects that are located on or off-screen. The remainder of this chapter is structured as follows: Section 8.1 introduces challenges and applications examples for interactive highlighting techniques in 3D geovirtual environments. Section 8.2 presents highlighting techniques, such as an object-based approach for adaptive landmark scaling and an image-based highlighting framework. Section 8.3 describes how 3D location that are outside the view frustum can be visualized using proxy objects. Finally, Section 8.4 summarize this chapter and present ideas for future work.

## 8.1 Applications and Challenges of Object Highlighting

Highlighting of objects as a form of visual feedback can be considered as an instance of higher-order visualization [17], similar to the principle of focus+context [46]. Usually, different object states are communicated by using different visual styles or appearances for rendering. These visual distinguishable styles, such as color overlay or using outlines, supports pre-attentive cognition. Despite changes of color or texture, also changes of geometrical properties such as geometric and cartographic scales can be used to highlight important objects [95].

Object highlighting has a number of applications not limited to 3D geovirtual environments. The *preview of a user's selection* can be considered as the classical use case for object highlighting. Further, in the field of visual analysis, objects to highlight can be the result of a computations. Instead of showing the results in a list, the *visualization of computational results* the application of object-highlighting has the advantage that a human can spatially cluster the highlighted result immediately. Furthermore, highlighting techniques can also be applied as *navigation aid* to landmarks, points-of-interest (POI), routes, as well as to navigation way points to guide the users attention.

Besides the high geometrical complexity of 3D GeoVE, an interactive highlighting technique has to deal with the following characteristics of a 3D virtual geoenvironment: First, *object sizes and shapes* can differ enormously. The size of an object can vary from large structures (e.g., places or buildings) to very small items (e.g., city furniture). Some of the shapes can have extreme spatial extent along a major axis and maybe not completely visible on the viewport. Second, depending on the application, a highlighting technique must be able to *highlight a number of different objects simultaneously* and maintain interactive frame rates. For instance, this case often occurs when editing virtual 3D city models. Finally, if unconstrained navigation and interaction metaphors are used, a highlighting technique has to handle different perspectives and orientations of the virtual camera, which have an influence on object occlusion and the objects size on the viewport.

An important aspect of highlighting is the appropriated visualization of landmarks or point-of-interest- These elements of geovirtual 3D environments are of outstanding importance for user orientation and navigation. Traveling in the real world depends on salient structures and objects, e.g., regarding their height, color, structure, or usage. These objects or structures are considered as landmarks, used by the human brain to create a mental map and remember the right way [267]. Especially, they are able to facilitate navigation and exploration within virtual 3D city models.

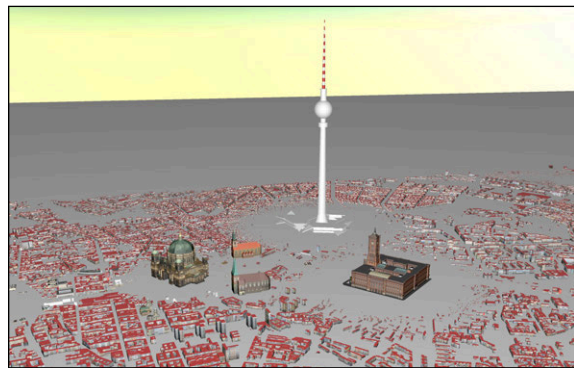
Highlighting of such objects concerns basically two major cases: (1) an object is visible within the current viewport (*on-screen*) or (2) it is not visible, because it is located outside the current view-frustum (*off-screen*). In case of 2D visualization on-screen points-of-interest are often displayed as small icons overlaying a map. The problem of visualizing 2D off-screen points-of-interest can be handled using *partially-out-of-the frame* approaches [9, 105].

## 8.2 On-screen Highlighting Techniques

The interactive rendering techniques for on-screen object-highlighting presented in this section can be classified into *object-based* and *image-based* approaches. Object-based approaches highlight objects by modifying their properties, such as the geometrical scale, *prior* to rasterization. Image-based approaches perform highlighting by modifying the image-based representations (G-Buffer [222, 74]) of respective focus objects or the context. This section introduces the basics of a highlighting framework for 3D GeoVE as well as hardware accelerated implementations for raster-based graphics. Applications and case studies of these techniques are presented in Section 9.1 and 9.3.

### Object-based Approach for Adaptive Landmark Scaling

This section presents a concept for the real-time depiction of landmarks that effectively emphasizes 3D landmark objects by improving their visibility with respect to their surrounding areas and the current 3D viewing settings [GTD07]. It is based on scaling landmark geometry according to an importance function while simultaneously adjusting the corresponding surrounding region. In order to be adaptive, the amplification of landmarks takes the parameters of the virtual camera into account. To reduce visual artifacts caused by such a multi-scale presentation (e.g., geometry intersections), the surrounding of each landmark is adapted according to a deformation field, which encodes displacement and scaling transformations. An individual weight coefficient can be defined that denotes the landmark's importance. To render a collection of weighted landmarks within a virtual 3D city model, the technique accumulates their associated, weighted deformation fields in a view-dependent way. The concept provides a flexible approach for the importance-driven enhancement of objects within interactive 3D GeoVE and aims at improving the perceptual and cognitive quality of their depiction.

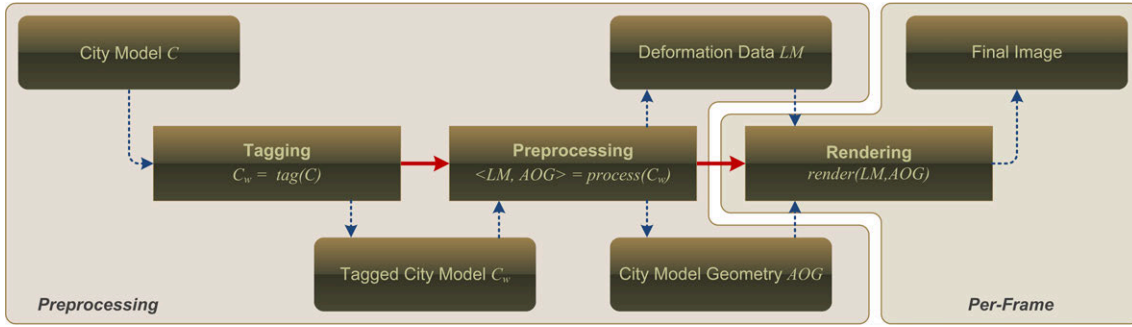


**Figure 8.1:** Enhancement of multiple landmarks in a virtual 3D city model.

Interactive 3D geovirtual environments can be used to provide more than just a photo-realistic rendering of reality: they give users a high degree of freedom for exploring complex geospatial and georeferenced information. In contrast to 2D maps or top-down views, 3D GeoVE exhibit the problem of occluding distant objects by objects close to the virtual camera. Thus, to facilitate effective navigation with 3D GeoVE, a user must be able to be aware of important landmark objects, even if they are occluded or the screen size of a projected object is too small [202]. In 2D maps, this problem is partially solved by displaying landmarks using different visual styles to reflect their importance. Depending on the geometric and cartographic scale, they (1) can be depicted in a larger size than their neighborhood [112, 126], (2) can be highlighted by different colors or drawing styles, and (2) exposed by clearing their immediate surrounding.

The concept presented in this section resolves the visibility problem by scaling landmarks to a





**Figure 8.2:** Components and processing stages of the adaptive landmark visualization concept.

sufficient size, i.e., a size that enables a user to identify them properly on the screen (Fig. 8.1). The weighted uniform scaling depends on the distance of the object to the virtual camera and therefore is dynamically adapted when the user explores the 3D GeoVE. The scaling transformation is performed dynamically in real-time and considers landmark objects that displace both, each other and their surrounding buildings.

Figure 8.2 shows the processing pipeline for the adaptive landmark scaling approach. The input data is a virtual 3D city model that is manually augmented with landmark weights during the data gathering stage (**tagging**). The tagged city model serves as input for the **preprocessing** stage: an automatic process that derives both city model geometry and deformation data required for the real-time deformation. At runtime, during **rendering**, the deformation data is evaluated and the objects of the city model are deformed at interactive frame rates. The user can explore and navigate the city model, leading to permanent updates of the deformation model and, hence, the scaling of the landmark objects.

**Tagging and Preprocessing** During the tagging and preprocessing stage, the city model data is augmented and transformed prior to rendering. In the tagging stages, weights are associated to a set of city model objects  $C = \{c_1, \dots, c_n\}$ , by defining an *importance function*  $w : C \rightarrow \mathbb{R}^+$ . Thus, the importance function defines a partition into two sets of city objects: landmark objects and non-landmark objects. The results of the tagging can be expressed as a mapping  $tag(C) = C_w = \{(c_1, w(c_1)), \dots, (c_n, w(c_n))\}$ . During the preprocessing stage, the city objects are analyzed automatically and a number of vertex attributes (a unique object identifier  $id$  and axis-aligned bounding box  $bb$ ) are computed and stored (Alg. 6).

---

**Algorithm 6** Pseudo-code for the geometric processing of a city model.

---

```

procedure process ( $C_w$ )
1: for all  $(c_i, w(c_i)) \in C_w$  do
2:    $id \leftarrow id(c_i)$  [assign unique identifier]
3:    $bb \leftarrow boundingBox(c_i)$  [compute axis-aligned bounding box]
4:   if  $w(c_i) > 1$  then
5:      $isLandmark(c_i) \leftarrow true$  [classify as landmark]
6:      $s_{c_i} \leftarrow scalingFunction(c_i)$  [compute scaling function]
7:      $LM \leftarrow LM \cup \{(id, s_{c_i})\}$  [store results]
8:   else
9:      $isLandmark(c_i) \leftarrow false$  [classify as non-landmark object]
10:  end if
11:   $AOG \leftarrow AOG \cup \{(generateGeometry(c_i), id, bb, isLandmark(c_i))\}$  [store results]
12: end for
13: return  $AOG$ 
14: return  $LM$ 

```

---

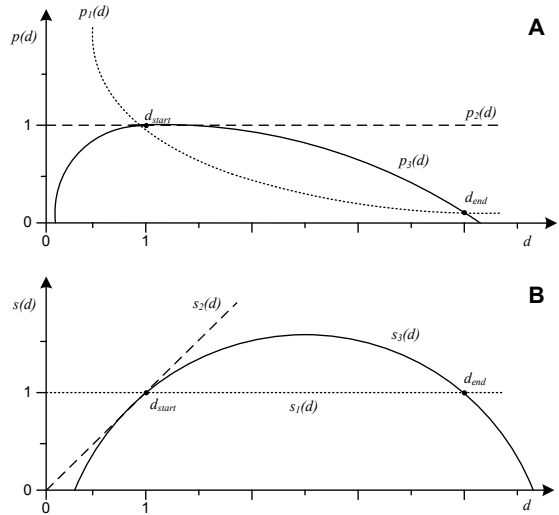
If the city object is a landmark object, a scaling function  $s$  is derived from the respective weighting

factor  $c_i$ . The scaling function could be for example a linear or quadratic function of the distance of which the coefficients are stored.

The next paragraphs describe how city model objects are scaled and displaced to achieve the desired landmark visualization. During the rendering of each frame the landmark objects are evaluated according to the deformation model, their bounding box, current position, and weight. Landmark objects are rated more important than non-landmark objects, hence non-landmark objects are scaled down or even omitted at all to achieve visibility of landmark objects. Sacrificing completeness and accuracy of the depiction can be reasoned with the superior significance of landmarks for navigation and is done similarly during cartographic generalization processes [112].

**Landmark Scaling** The motivation of the presented landmark visualization is to achieve visibility of certain important objects in a 3D environment. To accomplish this, the technique scales these objects if their appearance would be too small in the projected image. The scaling effect occurs only within a distance interval derived from the weight, defining a starting and ending distance  $I = [d_{start}, d_{end}]$ .

For clarification, two cases are described: using the standard perspective projection [3], an object with extent  $x = 1$  is projected on the viewing plane to an extent  $x'$ . The projected size  $p_1(d)$  of an object on the projection plane is inverse proportional to the distance  $d$  between the object and the virtual camera:  $p_1(d) = 1/d$ . Figure 8.3 shows the behavior  $p_1(d)$  (Fig. 8.3.A)



**Figure 8.3:** Projected size of a landmark object (A) and the scale factor (B).

If an object should keep a constant projected size (stippled line), its size (in the 3D scene) have to be scaled by the current distance prior to projection  $p_2(d) = 1 \cdot d/d = 1$ . However, the effect shall be locally limited depending on the landmarks weight, e.g., a small church is only an important landmark in its neighborhood but not relative to the entire city. Therefore, when zooming out from a landmark object that has its projected size kept constant, eventually it has to lose this property and return to its usual shape to avoid "crowding" the scene.

For these two cases, the objects are scaled before the projection by a scaling function as depicted in Figure 8.3.B:  $s_1(d) = 1$  and  $s_2(d) = d$ . To accomplish a smooth return to the original shape (i.e., scale = 1), a quadratic function  $s_3(d)$  is used that has a slope of 1 at a certain starting distance  $d_{start}$  for the exaggeration of the landmark object. The exaggeration effect is limited to ending distance  $d_{end}$ , where  $s_3(d)$  falls below 1. Figure 8.3 shows an example of  $s_3(d)$  for  $d_{start} = 1, d_{end} = 4$ . Further,  $d_{start}$  and  $d_{end}$  are derived from the single weight parameter  $w(c_i)$ . As they only depend on the previously defined weight of the landmark, the function  $s_3(d) = ad^2 + bd + c$  is precomputed using the distance interval and its coefficients  $a, b, c$  are stored for each landmark object.

**Displacement of Landmark and Non-Landmark Buildings** Scaling landmark objects implies that surrounding objects (e.g., other landmark objects) have to be displaced to avoid self-intersection artifacts. Approaches for this problem have been researched in simulated annealing [146, 268], least squares adjustment [234], and spring mass models [22]. Similar to the latter one, the problem is approached using a naive spring model without mass: overlapping objects create a repelling force that shifts the objects apart. This model is applied iteratively until no shifting occurs or a maximum of iterations is reached. In spite of its simplicity, the model yields acceptable results and sufficient performance, which is a requirement since this operation is performed on a per-frame basis. While this spring model can be computed sufficiently fast for a small number of objects, it is hardly applicable for the entirety of city objects. However, this is not necessary since common city

objects usually will be too small to reason a high computational effort for their correct individual positioning. Instead, a radial distortion is applied to all non-landmark objects in the environment of a landmark object as an application of distortion lenses presented by Carpendale et al. [43]. These lenses using a drop-off function that define how features in the vicinity of the lens are displaced and scaled.

To limit the distortion effect locally, a distortion zone is defined twice the size of the scaled landmark's extent. This zone compresses and offsets the environment of the landmark to fit it into the distortion zone. The translation function shifting elements at distance  $x$  from the landmark center within the distortion zone is defined:

$$t_{nonLM}(x) = s \cdot e + (x - e) \cdot \frac{s}{2s - 1}$$

with  $s$  being the landmark's scaling and  $e$  its half extent. Just offsetting the neighboring buildings would yield self-intersections; therefore an additional scaling function is defined that shrinks buildings within the distortion zone:

$$s_{nonLM}(x) = \frac{x}{2s \cdot e} \quad \text{with} \quad 0 \leq x \leq 2s \cdot e$$

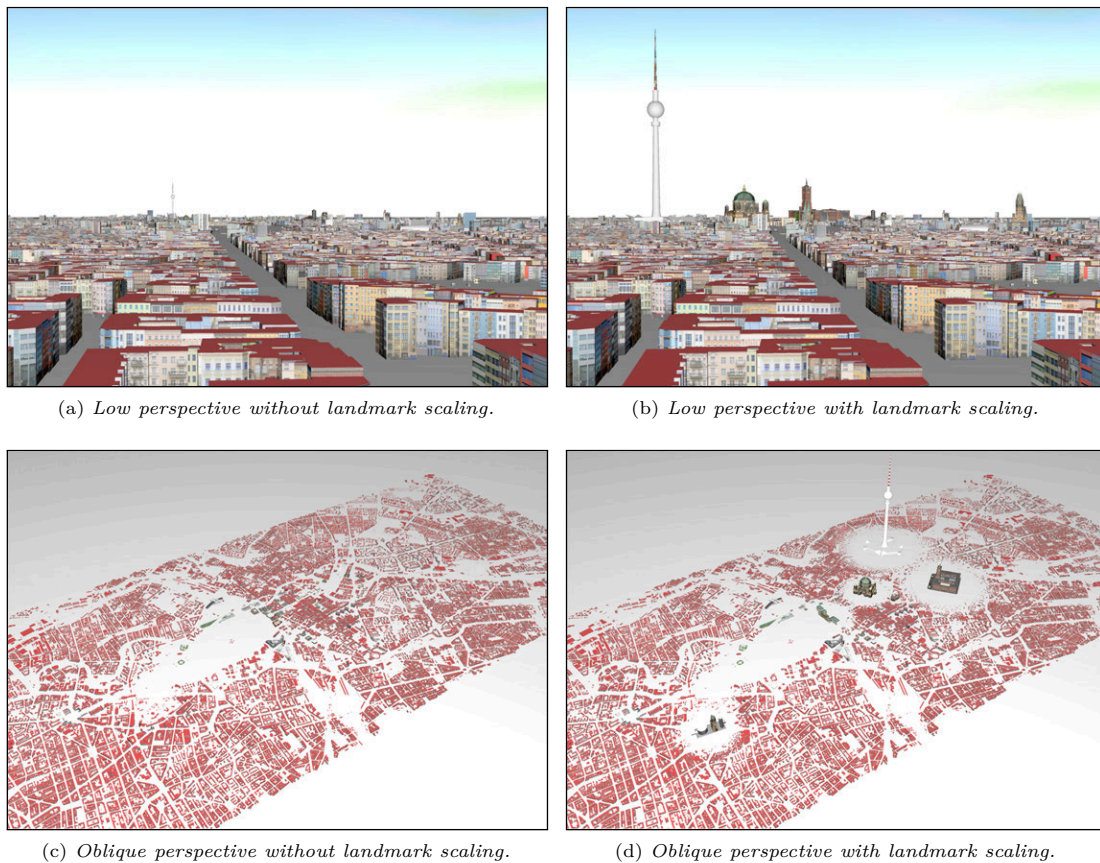
Objects near the landmark are smaller in size and become linearly larger until they reach their original size. Thus, the lens effect integrates smoothly in the city model while it exposes the landmark object.

**Interactive Rendering Technique** The rendering process comprises two passes performed on a per-frame basis: a *pre-traversal pass* and a *rendering pass*. The pre-traversal pass traverses a scene graph that represents the virtual 3D city model and collects all attribute nodes that represent landmarks. The resulting set can be optional culled against the current view-frustum [3] to reduce computation complexity. Subsequently, the deformation model determines the deformation parameters that are encoded for the subsequent rendering pass by using global shader constants [144]. During the rendering pass, a vertex shader program is activated that deforms every vertex of the building geometry according to these constants. Additional vertex attributes, such as object identity (*id*) and the buildings bounding box (*bb*), which are computed or set during the preprocessing of the scene-geometry, enable the distinction between landmark and non-landmark geometry. The shader program uniformly scales and displaces the landmark geometry or applies the deformation parameters to clear space for the landmarks.

This approach is efficient in terms of rendering complexity because the complete scene geometry is rendered only once per frame. Thus, the rendering performance is limited only by the number of landmarks and the geometric complexity of the virtual 3D city model. Despite the limited physical resources such as main and graphics memory this concept is mainly limited by hardware related issues: With an increasing number of visible landmarks, the computation cost of the deformation parameters on CPU side can stall the GPU. Further, the encoding of the deformation parameters into an adequate assignment of constant registers can exceed the limitations of shader programs.

**Summary and Discussion** This section presents a concept and rendering technique for the real-time depiction and highlighting of 3D landmarks. These 3D objects are emphasized by improving their visibility with respect to their surrounding areas and the current 3D viewing settings. Figure 8.4 compares the scaling approach with a standard renderings from the same viewpoint. The enhancement by scaling improves the perceptual and cognitive quality of the landmark display. Consequently, it facilitates the task of identifying landmarks that can have an impact on navigation and exploration of virtual 3D city models. Further, this visualization technique facilitates an overview of the main landmarks. The choice of an adequate scaling function is important for the interactive application of this concept. The recurrence of a landmark to its original size on close and far distances is necessary to enable a smooth integration into standard 3D navigation techniques as describes in [37].

The presented concept has limitations and drawbacks. The spring model applied to control the mutual landmarks displacement cannot guarantee a stable frame-to-frame coherence, i.e., jumps or



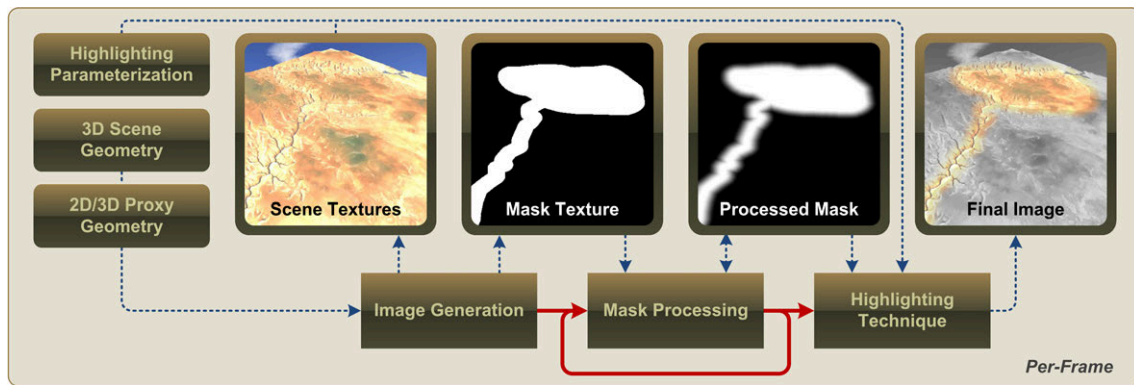
**Figure 8.4:** Comparison between standard (left column) and enhanced landmark rendering (right column) using different perspectives.

popping artifacts can occur. Further, this model is not able to properly visualize a high number of landmarks having equal or similar weights. In addition, the current deformation model cannot fully avoid self-intersections for non-landmark buildings. Furthermore, it seems to be useful to research the impact of shape-preserving (uniform) deformation vs. per-vertex (non-uniform) deformation to the viewer's perception.

However, the concept delivers an adequate approach for the landmark enhancement problem. Nevertheless, several features should be improved and further investigated. Despite handling only points-of-interests, the approach can be generalized for the application to regions of interest [202]. In addition, the research of other displacement models or alternative scaling function can be of interest. The usage of object-oriented bounding boxes as well as the consideration of the buildings perimeter could achieve a more precise displacement of the surrounding area. This approach can also be extended for the non-uniform displacement and scaling of terrain and line data such as streets or railway lines, as well as other surface objects such as geometric representations of land use data. Further, a landmark could be enhanced by rotating it towards to camera, which enables a priority-based view on the landmark. The necessary information for this enhancement could also be processed during the tagging step.

### Image-based On-Screen Highlighting

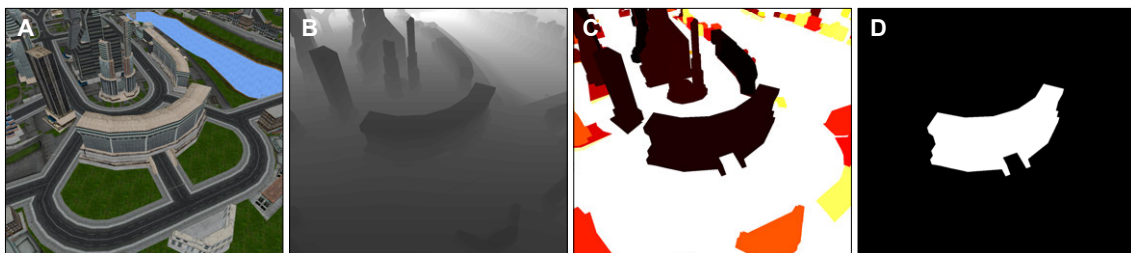
The previous section presents an object-space approach that creates geometric scale variances for the point-of-interest and object-of-interest focus types. This section presents a real-time rendering technique that performs highlighting for region-of-interest focus types using style variances. Fig. 8.5 shows an overview of the rendering pipeline used for the interactive image synthesis. Basically, it comprises of the following three stages: (1) an *image generation* step forms the basis for the continuous fragment-level separation between object- or regions-of-interest to highlight and the



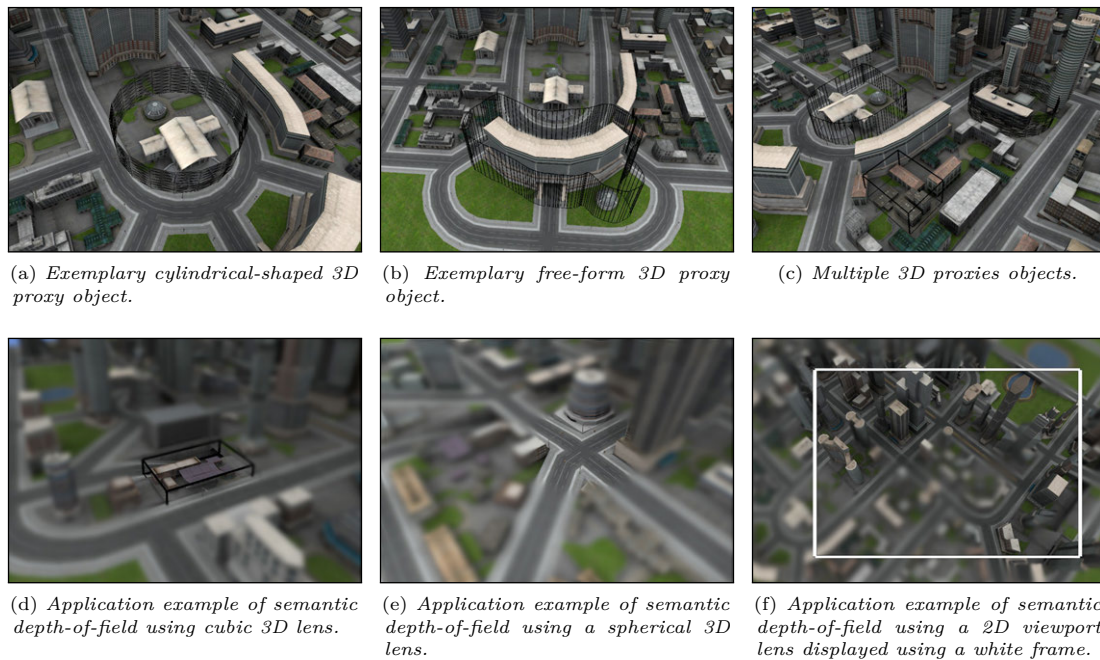
**Figure 8.5:** Overview of the rendering pipeline for applying on-screen highlighting techniques to 3D GeoVE in real-time. The data flow is depicted using stippled blue and the control flow uses solid red lines.

remaining 3D scene. Here, Figure 8.6 shows the resulting data (scene textures) that is required for the remaining two subsequent steps: (2) to enable smooth transitions between focus and context regions, a *mask processing* step applies image-based post-processing methods (e.g., jump-flooding [214] or convolution filtering required for glow effects [191]) to the mask texture; and (3) in a final step, style variance, outline, or glyph-based highlighting techniques are applied to the scene texture. For the final visualization, the result of this step is written into the frame buffer. The pipeline takes textured polygonal geometry as input. It requires a unique identifier per scene object. To increase the rendering performance, each numerical object ID is encoded as a per-vertex attribute of the specific input mesh in a preprocessing step. This procedure enables geometry batching [3] or geometric streaming without modifying the proposed rendering pipeline. Therefore, the presented approach is suitable especially for real-time rendering of geometrical complex 3D scenes, since the geometry is rasterized only once.

**Synthesis of Image-based Object Representations** Image generation represents the first stage in the highlighting pipeline and enables the separation of focus and context regions. During a single rendering pass, image-based representations (scene textures) of the 3D geometry and a mask texture are created at viewport resolution (Fig. 8.6). Therefore, off-screen rendering in combination with fragment shaders and multiple rendering targets are used, which enables writing into multiple raster buffers simultaneously. At first, the generated mask texture (Fig. 8.6.D) contains a discrete image-based focus representation. Two methods for representing the input for the mask generation step are distinguished: *scene objects* and *proxy objects*. If complete scene objects are selected for highlighting, their fragments are taken as input for generating the mask texture (Fig. 8.6.D). Their respective object identifier are encoded using an ID texture (Fig. 8.6.C). This approach works only for highlighting complete objects.



**Figure 8.6:** Image-based representations (scene textures of Fig. 8.5) of a 3D scene that are required by the highlighting pipeline: color (A), depth (B), false colored object identifiers (C) and mask (D) values are derived within a single rendering pass.

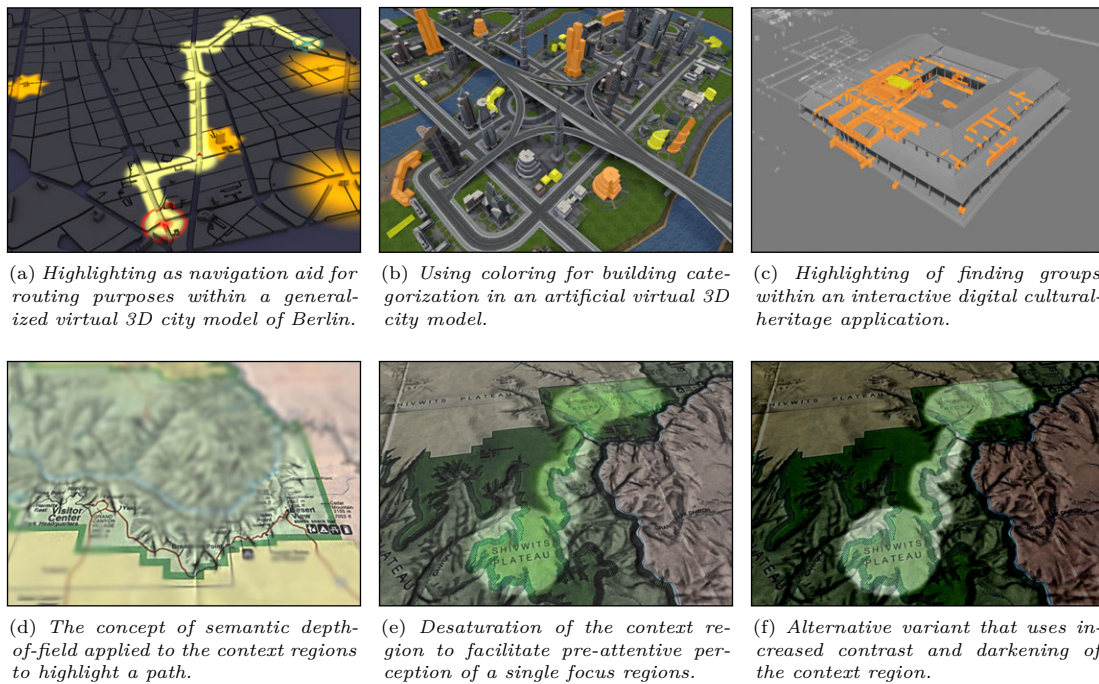


**Figure 8.7:** Application of proxy-geometry for representing discrete 2D and 3D focus types.

If it is required to highlight only parts of a single or multiple objects, as well as highlighting focus regions of a 3D scene that have no geometrical representation, so called proxy objects can be used. These are presented using additional polygonal geometry that is located in the 3D scene or on the viewport. Another important use case for proxy objects is the partially highlighting of objects or the highlighting of regions affecting multiple objects (e.g., in the vicinity of a route). Here, the proxy geometry is usually created manually by the user (Fig. 8.7.A-C), e.g., by directly painting the proxy shapes on the viewport. The resulting line segments are converted to a polygonal representation using buffering and extruding operations and used as input for the mask generation step. Later, the user can modify the position, scale, rotation of a proxy interactively. Note that the color and depth values of proxy objects are not written into the color and respective depth texture. Despite approximations of small objects, proxy shapes can be used for implementing Magic Lenses [15]. The concept distinguishes between 2D and 3D lenses. 2D lenses are located in screen space and move with the virtual camera. They can be used to implement auto-focus features as described in [119, 120] (Fig. 8.7.F). The proxy geometry of 3D lenses is placed in the 3D scene and does not align with the virtual camera (Fig. 8.7.D and E).

**Mask Processing and Highlighting Operations** After the mask texture is rendered, a mask processing step is performed. It enables the creation of smooth transition between the focus and context regions within the mask texture. The mask processing is implemented using RTT in combination with multi-pass rendering. Basically, two different processing algorithms are applied: jump-flooding and convolution filtering. A jump-flooding algorithm [215] is used to perform distance transforms between focus and context regions. If only a small dilation of the input mask is required, e.g., for creating the glow outline, convolution filtering can be applied. The final value of the mask can be controlled by a global drop-off function. It weakens or exaggerates the previous results from jump flooding or convolution filtering.

After processing the mask texture, the final pipeline phase applies style variance or outline highlighting techniques at a per-fragment level, using fragment shader. Therefore, the color values in the scene texture (Fig. 8.6.A) are used to apply color overlay or vignetting based on the values of the mask texture. Given the object identifier (Fig. 8.6.C), each object can be highlighted using a different technique. In general, the intensity of an effect, (e.g., blur or opaqueness of a highlighting color) is controlled by the values of the mask texture. For example, Figure 8.7 (bottom row) shows



**Figure 8.8:** Application examples of different highlighting techniques applied to virtual 3D city models (top row) and a virtual 3D landscape model of the Grand Canyon (bottom row).

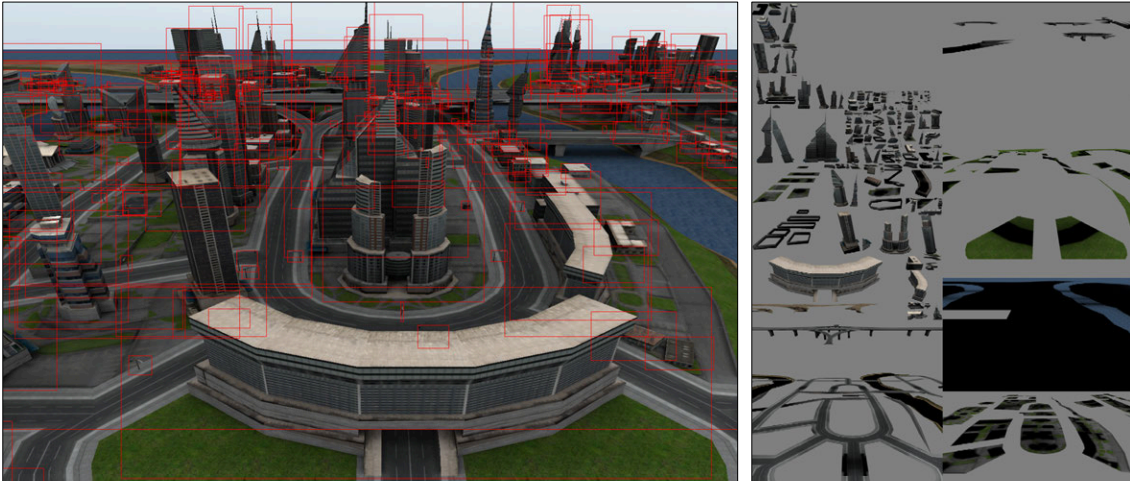
an implementation of semantic depth-of-field (SDOF) [149]. It applies convolution filtering with different kernels in combination with multi-pass rendering. The trade-off between rendering speed and output quality can be controlled by the choice of the filter kernel. Gaussian blur requires additional rendering passes than a box filter, but delivers a better blur quality.

Subsequently, the results of the previous step is applied to a screen-aligned quad that is rendered into the frame buffer. The stored depth values (Fig. 8.6.B) are also written into the frame buffer. Finally, glyph-based highlighting techniques (e.g., bounding boxes or arrows) are applied to the frame buffer using forward rendering.

**Application Examples for Image-based Highlighting Techniques** Figure 8.8 shows application examples for the presented concept. Besides the highlighting of single objects, the approach enables route highlighting and the visualization of multiple objects. Figure 8.8(a) shows an example for route and landmark highlighting using style variances applied to focus and context regions embedded in a generalized version of a virtual 3D city model of Berlin [95]. A vignetting technique is used to highlight a street and color-highlighting is applied to the start and end position of a route. In addition, nearby landmarks are highlighted.

Figure 8.8(b) shows the application of color overlays to categorize a number of buildings. The colors, which are blended with the facade textures, can be used to encode specific data values. Figure 8.8(c) shows the application of object highlighting in the domain of digital cultural heritage. The findings of a basement are highlighted in yellow to delimit them from the remaining artifacts, which base color is orange (cf. Section 9.1). The bottom row of Figure 8.8 shows different post-processing approaches applied to the context instead of the focus regions.

**Rendering Performance** The performance evaluation of the rendering technique uses data sets of different geometrical complexity: the generalized model of Berlin comprised 1,036,322 rendering primitives, the model of the Grand Canyon 1,048,560 primitives, and the artificial 3D city model comprises 34,596 primitives. The performance tests are conducted using a NVIDIA GeForce GTX 285 GPU with 2048 MB video RAM on a Intel Xeon CPU with 2.33 GHz and 3 GB of main memory. The 3D scene geometry was not batched and no view frustum or occlusion culling is applied. All depicted scenes in real-time are rendered within the range of 12-32 frames-per-second.



**Figure 8.9:** Example of a perspective texture atlas for a virtual 3D city model. The left image shows the transformed bounding boxes of each visible scene object and the left image shows the resulting perspective texture atlas.

The performance of the presented image-based approach is geometry-bounds by the geometrical complexity of the 3D scene and fill-limited by the number of mask-processing passes, which have to be performed at the viewport resolution.

### Interactive View-dependent Texture Atlases

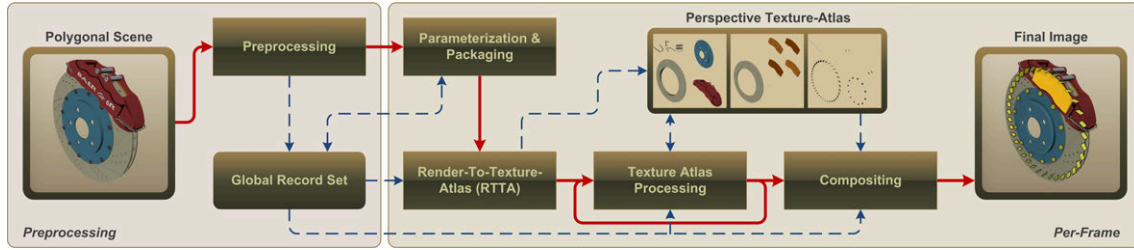
View-dependent texture atlases (Fig. 8.9) are an extension of the mask generation step in the highlighting pipeline. They are especially suitable for highlighting a high number possibly overlapped or occluded objects (Fig. 8.10). The image-based representation of geometry is a well known concept in computer graphics [222]. Due to z-buffering, the derivation of such representations using RTT delivers only information of the closest fragments with respect to the virtual camera. Often, transparency-based visualization techniques (e.g., ghosted views) also require information of occluded fragments. These can be captured using multi-pass rendering techniques such as depth-peeling [10] or stencil-routed A-buffers [183] on a per-fragment basis. This section presents an additional rendering technique that enables the derivation of image-based representations on a per-object level within a single rendering pass [TD10]. The approach uses a dynamic 3D texture atlas that is parametrized on a per-frame basis. Prior to rasterization, rendering primitives are transformed to their respective positions within the texture atlas using vertex-displacement in screen space [237].



**Figure 8.10:** Example for multi-layered object-highlighting using view-dependent texture atlases.

**G-Buffer Synthesis for Overlapping and Occluded Objects** The concept of image-based representation of 3D shapes [222] has numerous applications in computer graphics. Despite static and dynamic impostors [53], they represent the basis for advanced rendering effects performed in post-processing (e.g., edge detection, screen-space ambient occlusion, deferred shading). For the purpose of image-based occlusion management [78] (ghosted views), object highlighting, or the enhancement of depth perception (halos), it is necessary to efficiently generate such representations for all, or a subset of scene objects within a 3D geovirtual environment.





**Figure 8.11:** Components, control and data flow for the generation and rendering of view-dependent textures-atlases applied to a technical 3D model of the automotive domain.

An application that uses RTT capabilities of current rendering hardware to derive these representations on a per-object basis encounters two major challenges: (1) only fragments with the minimal depth value (with respect to the virtual camera) are captured; and (2) either the complete 3D scene or a single scene object can be captured occlusion-free during a single off-screen rendering pass. The first challenge is coped efficiently using existing rendering techniques such as depth-peeling [163] or stencil-routed A-buffer [183]. These techniques operate at fragment level and usually require multiple rendering passes. The second challenge can be handled using multiple rendering passes in combination with multiple render-targets (textures). However, such an approach results in multiple, sparse texture layers, which require additional management and, if at high viewport resolution, yielding to high GPU memory consumptions.

For applications that require only a single texture layer representation of an object, this section presents render-to-texture atlas (RTTA): an interactive and scalable rendering technique that enables dynamic generation of occlusion-free, image-based representations for multiple scene objects using graphics hardware (Fig. 8.9). It extends RTT by using a single 3D texture-atlas as render target for all scenes objects. In contrast to the original concept of texture-atlases [269], it computes the texture-atlas parametrization and packaging per rendering frame with respect to the projected boundary approximation (e.g., axis-aligned bounding box) of each object. During off-screen rendering, it uses screen-space vertex displacement to transform each object into its respective texture-atlas region. This is achieved using an additional 2D transformation that is applied to each object prior to rasterization. The approach enables the usage of optimized (batched) scene geometry, which reduces state changes during rendering. Further, it can be easily integrated into existing rendering frameworks and systems. To summarize, this section present a concept for view-dependent parametrization and generation of a single texture atlas containing all image-base representations of projected scene objects. Further, it describes the concept of screen-space vertex displacement and its application for generating view-dependent texture-atlases within a single rendering pass. Furthermore, it briefly describes a hardware-accelerated rendering technique that implements this concept and discuss its performance and limitations.

**Concept of Render-To-Texture-Atlas** The concept mainly consists of two phases that are performed per frame (Fig. 8.11): (1) the view-dependend computation of the texture-atlas parametrization and subsequently (2) the rendering of the scene geometry into a texture atlas. A texture atlas  $TA = (t_w, t_h, t_d) \in \mathbb{N}^3$ , denotes a number of  $t_d$  layers of 2D textures, each with a fixed width  $t_w$  and height  $t_h$ . This data structure can be effectively represented on graphics hardware using 3D textures or 2D texture arrays. It is assumed that the orientation and projection transformation of the virtual camera can be described by a matrix  $\mathbf{VPM}$ , and that the scene is rendered to a viewport given by  $VP = (x, y, w, h) \in \mathbb{N}^4$ . At runtime, the concept requires global information about the objects of a 3D scene. Such record can be computed off-line for static meshes or dynamically for animated scenes. For each object, a record  $R_{ID}$  of the following structure is stored in a *global record set*  $\mathcal{R} \in \mathcal{R}$  with:  $R_{ID} = (B_{world}, B_{viewport}, B_{atlas}, l, \mathbf{T})$ .

To identify an object at runtime, an unique object identifier  $ID \in \mathbb{N}$  is required. This identifier must be encoded as a per-vertex attribute to enable geometry batching and arbitrary scene partitions for rendering. To approximate the area a 3D object occupies in a 2D texture atlas, its 3D boundary representation  $B_{world}$  is computed in world-space coordinates. The presented approach uses 3D

**Algorithm 7** Texture-Atlas Parametrization and Packaging Algorithm

---

```

1: for all  $R_{ID} \in \mathcal{R}$  do
2:   if viewFrustumCulling( $B_{world}, \mathbf{VPM}$ ) then
3:      $B_{projected} = \text{project}(B_{world}, \mathbf{VPM})$  [Compute projected bounding box]
4:      $B_{clipped} = \text{clip}(B_{projected})$  [Clip bounding box against viewport]
5:      $B_{viewport} = \text{scale}(B_{clipped}, VP)$  [Scale to viewport]
6:      $B_{viewport} = \text{addBorder}(B_{viewport}, b)$  [Dilate scale bounding box]
7:      $R_{ID} \leftarrow B_{viewport}$  [Store 2D bounding approximation]
8:   end if
9: end for
10:  $(TA, T_d) \leftarrow \text{atlasPackaging}(\mathcal{R}, T_{w_{max}}, T_{h_{max}})$  [Perform atlas packing]
11: for all  $R_{ID} \in \mathcal{R}$  do
12:    $\mathbf{T} = \text{computeTransform}(B_{viewport}, B_{atlas}, TA, VP)$  [Compute displacement transformation]
13:    $R_{ID} \leftarrow \mathbf{T}$  [Store transformation]
14: end for

```

---

axis-aligned bounding boxes (AABB) as boundary representation. The 2D rectangular boundary  $B_{viewport} \in VP$  denotes the clipped on-screen area of  $B_{world}$  and  $B_{atlas} \in TA$  the occupied area within the texture atlas. An affine 2D transformation matrix  $\mathbf{T}$  describes the transformation of  $B_{viewport}$  to  $B_{atlas}$  in normalized device coordinates (NDC). Further,  $l = 0, \dots, t_d$  denotes the texture layer each object is rasterized to.

Prior to RTTA, the texture-atlas parameterization needs to be determined, i.e., the mapping of a 3D boundary representation ( $B_{world}$ ) into a 2D texture domain ( $B_{atlas}$ ) for all records  $R_{ID}$ . Algorithm 7 shows the pseudo code for computing this mapping. In the first step, the boundary representation  $B_{world}$  is conservatively culled against the view frustum defined by  $\mathbf{VPM}$ . On success, it is projected into normalized device coordinates ( $B_{projected}$ ) and clipped against the area  $[-1, -1] \times [1, 1]$ . The resulting 2D boundary ( $B_{clipped}$ ) is scaled ( $B_{viewport}$ ) with respect to the viewport  $VP$ . To enable artifact-free convolution filtering during texture-atlas post-processing and compositing (Fig. 8.11), a border of  $b \in \mathbb{N}$  pixels can be added. Next, texture-atlas packaging computes  $B_{atlas}$  for each  $B_{viewport}$ , starting a maximal texture-atlas resolution of  $T_{w_{max}}$  width and  $T_{h_{max}}$  height. After completion, it also delivers the resolution and the required number of texture layers  $T_d$ . For example, an implementation can use a rectangular atlas packaging approach [124], which has a runtime complexity of  $O(n) = n \log n$ . Finally, based on the packing results, a displacement transformation  $\mathbf{T}$  is computed that basically consists of a translation and a scaling transformation.

This displacement transformation is further denoted as screen-space vertex displacement (SSVD). An early type was introduced for rendering *camera textures* [237]. It can be applied in image- or object-space by transforming vertices using a translation vector stored in a 2D texture. For the purpose of RTTA, this concept is extended to arbitrary affine 2D transformations to transform a rendered primitive into its respective atlas region prior to rasterization. Every vertex  $V = (x, y, z, w)$  is transformed into its designated texture-atlas area  $B_{atlas}$  by displacing it parallel to view plane using  $\mathbf{T}$ . The new vertex position can be obtained by:  $V' = \mathbf{T} \cdot V$ .

**Real-time Implementation using Geometry Shaders** The prototypical implementation is based on OpenGL [230] in combination with GLSL [144]. To perform SSVD using the geometry shader stage the global data record  $\mathcal{R}$  is encoded into a suitable GPU data structure. Therefore, the transformation matrix  $\mathbf{T}$ , the target layer  $l$ , and the atlas region  $B_{atlas}$  of each record  $R_{ID}$  are stored linearly in a single texture-buffer object, denoted as *record buffer*. At runtime, the respective object  $ID$  is used to index this buffer. The buffer is encoded per-frame and shared between RTTA, successive texture atlas post-processing, and compositing steps.

The rendering setup for RTTA is similar to standard RTT applications. First, the framebuffer objects and render textures are set up according to  $TA$ , and second the viewing and projection

```

1  uniform samplerBuffer recordBuffer; // global data
2  in int ID[3]; // per-vertex attribute: object ID
3  // fetch transformation and layer for object ID
4  void fetchRecord(in ID, inout mat4 T, inout int layer);
5  ...
6  void main(void) {
7      mat4 T; int layer; fetchRecord(ID[0], T, layer);
8      gl_Layer = layer; // set texture-target layer
9      for(int i = 0; i < 3; i++) // set every vertex
10     {
11         vec4 v = gl_ProjectionMatrix * gl_PositionIn[i];
12         gl_Position = (T * (v / v.w)) * v.w; // screen-space vertex displacement
13         gl_ClipVertex = gl_PositionIn[i]; // set clip vertex to original position
14         // set additional attributes...
15         EmitVertex();
16     } //endfor
17     EndPrimitive(); // close and emit primitive
18     return;
19 }

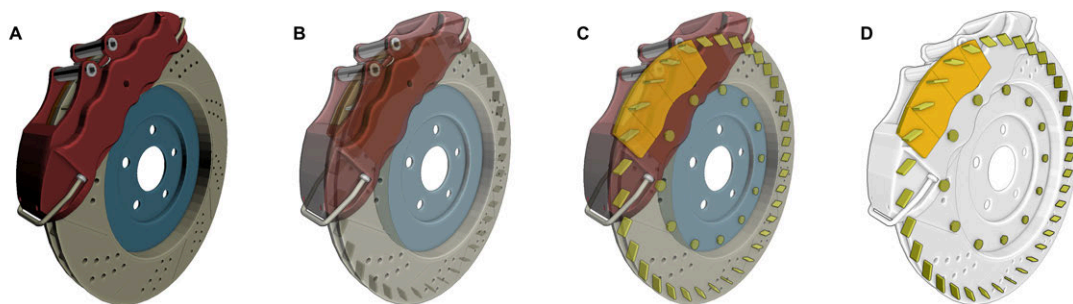
```

**Listing 8.1:** Geometry shader implementation of RTTA.

transformation  $\mathbf{VPM}$  is applied and viewport dimensions are set to  $T_w$  and  $T_h$ . After binding the record buffer (`recordBuffer`) a shader program (Listing 8.1) is enabled before rendering the scene geometry. The shader performs SSVD for each vertex and assigns the respective layer of the texture atlas to each output primitive.

After RTTA is performed, an application-specific processing of the texture-atlas contents (e.g., edge-detection, color quantization, glow) can be applied. The final compositing is performed on per-object level. Figure 8.12 shows exemplary results for per-object compositing using frame-buffer blending. This is performed by generating and rendering 2D sprites [3] for each object using the point-sprite expansion functionality of the geometry shader. Therefore,  $n = |\mathcal{R}|$  point primitives with their respective object  $ID$  are rendered and converted into screen-aligned quads. Given the viewport setting  $VP$ , the four corner points are set according to  $B_{atlas}$  and transformed to  $B_{viewport}$  using the inverse transformation matrix  $\mathbf{T}^{-1}$ .

**Performance Evaluation and Discussion** The performance tests are conducted using a NVIDIA GeForce GTX 285 GPU with 2048 MB video RAM on a Intel Xeon CPU with 2.33 GHz and 3 GB of main memory. Table 8.1 shows the results of a comparative evaluation. The tests are performed at a viewport and texture atlas resolution of  $1024^2$  pixels without view-frustum culling enabled. The performance mainly depends on the number of scene objects and is bound by the performance of the geometry-shader stage. To summarize, this section presents the concept of view-dependent textures-atlases. It enables the generation and management of occlusion-free, image-based representations for multiple overlapping objects in complex 3D scenes. It further describes a real-time, hardware accelerated implementation that generates these textures-atlases within a single rendering pass



**Figure 8.12:** Compositing variants derived from a view-dependent texture-atlas containing color and depth values per pixel. A: reconstruction of the scene by rendering depth-sprites; B: transparent rendering by ignoring the fragments depth (requires depth sorting); C: ghosted-view visualization showing a brake pad and screws highlighted; D: ghosted-view visualization that uses screen-space ambient occlusion and edge-enhancement during compositing for important scene objects.

**Table 8.1:** Comparative performance evaluation for different test scenes between plain rendering (STD), standard render-to-texture (RTT), and render-to-texture atlas (RTTA) (in frames-per-second).

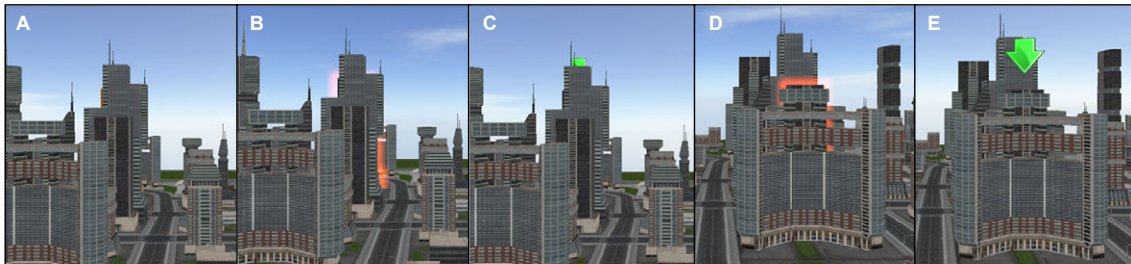
#Vertex	#Primitives	#Objects	STD	RTT	RTTA
2,191	4,236	5	2066	1015	940
32,081	21,246	21	1066	328	239
56,654	34,596	580	65	14	19
1,040,503	346,835	269	39	21	35

and demonstrate its applications for interactive ghosted views. One conceptual problem of the implementation concerns the usage of 2D rectangular boundary approximations for representing  $B_{viewport}$  and  $B_{atlas}$ . This can lead to under-utilization of the texture atlas, especially for objects with non-convex shapes. For representing objects that cover the entire screen, the utilization can be improved by choosing  $T_w$  and  $T_h$  as multiples of the viewport size. A further problem represents the dynamic allocation of texture memory if adapting the texture-atlas size on a per-frame basis: driver stalls during the removal of layers from the texture array can be observed. This is counterbalanced by performing lazy-updates of the texture array at (1) rendering idle time or (2) if the number of layers have not changed during a number of passes. Another hardware limitation represents the maximal number of texture layers  $T_d$ , as well as its resolution  $T_w$  and  $T_h$ . Thus, the maximal number of objects that can be captured within a single pass depends on this resolution and the texture atlas utilization. For future work, one can research the application of more complex types of boundary representations, such as (hierarchical) object-oriented bounding volumes, to achieve a better texture-atlas utilization.

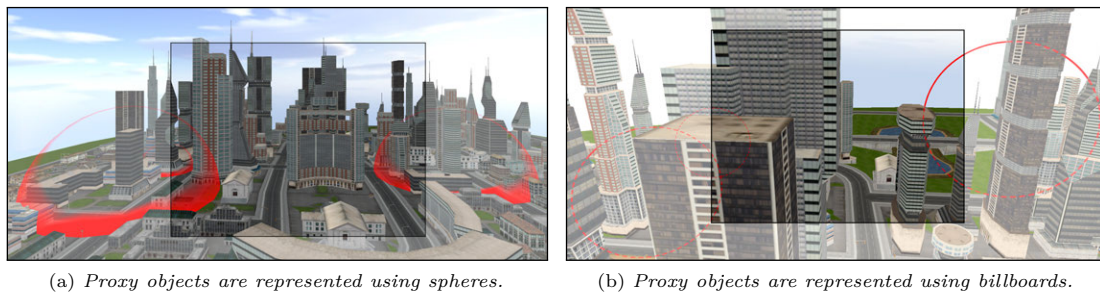
### Comparison of On-screen Highlighting Techniques

Fig. 8.13 shows a comparison of style-variance, outline, and glyph-based highlighting techniques with respect to object occlusion and depth-cues. Although color-based and glyph-based highlighting techniques are well established techniques for directing the focus-of-attention to an object, they suffer from disadvantages: in the case of color highlighting, the appearance of the building is altered because the facade is dyed. As a result, important information, such as building color or texture details, become hardly recognizable. Further, in the area of visual analytics, the color attribute is often used for thematic mappings. Hence, color highlighting can lead to unfeasible interpretations. Another problem in virtual 3D geoenvironments is occlusion. If the object-of-interest is covered by other objects, the viewer may hardly perceive any visual hints to guide his or her attention (Fig. 8.13.A).

In contrast to that, glyphs neither change texture information nor the buildings appearance. Instead, an additional geometric feature (glyph) can be attached to an object. The size and the position of a glyph can be adapted dynamically to avoid occlusion and ensure its visibility (Fig. 8.13.C). One disadvantage of this method is a lack of depth cues due to missing scale hints. If the object is occluded by additional geometry and the scene is displayed in a near ground perspective, the user may hardly distinguish to which object the glyph is attached to (Fig. 8.13.E).



**Figure 8.13:** Comparison of style-variance, outline, and glyph-based highlighting techniques with respect to occlusion (A-C) and the objects depth cue (D and E).



**Figure 8.14:** Mock-ups to clarify the terms off-screen and partially-out-of-the frame: Proxy-objects (red) are displayed partially-out-of-frame to enable a user to approximate point-of-interest positions that are located off-screen. The saturated areas show the viewport visible to the user and the desaturated areas show the surrounding scene including the positions.

Using an context-based or outline highlighting technique as a method of object emphasis seems to be a promising approach. First, no relevant appearance parameters of the objects are altered. Second, even if objects are partly or completely occluded, an outline or glow can still be recognized to a certain degree (Fig. 8.13.B). Problems of thematics mappings are reduced because the outline can be seen as a propagated building silhouette and is, therefore, view invariant and supports an acceptable depth cue. (Fig. 8.13.D). The application of SDOF (Fig. 8.7 (bottom row) and Fig. 8.8(f)) to 3D GeoVE exhibits a number of problems. If applied to virtual 3D city models, the user is often distracted while trying to focus the blurred areas. In the case of 3D landscape models, viewed from a birds-eye perspective, this effect appears less stronger. However, semantic depth-of-field fails if the screen size of a highlighted object is too small.

### 8.3 Off-screen Highlighting Techniques

3D geovirtual environments are increasingly used as general-purpose medium for communicating spatial information. In particular, virtual 3D city models have numerous applications such as car navigation, city marketing, tourism, and gaming. In these applications, points-of-interest (POI) play a major role since they typically represent features relevant for specific user tasks and facilitate effective user orientation and navigation through the 3D GeoVE. This section presents strategies and approaches that aim for the effective visualization of points-of-interest in a 3D geovirtual environment. A major problem of 3D GeoVE is the so-called "keyhole" situation, i.e., users can perceive only a small part of the geovirtual environment due to the limited field-of-view and image resolution. Thus, it is probable that a number of point-of-interest are located outside the view-frustum. This section presents approaches for the effective visualization of points-of-interest that are located out-of-frame or off-screen (Fig. 8.14) by introducing additional visual cues.

Within a 2D GeoVE, a user can efficiently compare distances to multiple locations, such as the distances to multiple restaurants, to estimate which of these is the nearest. For a complete estimation, all relevant objects have to be within the viewport. Due to zooming or panning, some of these objects may disappear into off-screen space. Especially for small-screen devices, the user is forced to frequently zoom in and out, and pan to see the off-screen objects. Consequently, within a certain zoom level, it is hardly possible to see all relevant objects, but only a subset of them, which makes spatial routing tasks more complicated and time-consuming. Semantics of 3D virtual representations can be derived from real world objects by comparing them by means of reference points. These reference points reflect the main characteristics of the depicted objects and are needed to construct a relation between virtual object and real world object, according to the bilateral term of characters in the theory of semiotics [51]. If there are enough reference points, as usual for 3D representations, the interpretation can be handled in an efficient and effective way. Considering this as a crucial advantage of 3D space, the 2D Halo visualization technique [9] is supposed to be adapted within a 3D GeoVE.



(a) An object-space 3D Halo Circle visualization combined with occlusion management showing three landmarks. One landmark is on the left front side out-of-view. One landmark is farther away on the left side out-of-view and slightly behind the camera. One landmark is right in front but occluded by another building (which is therefore rendered as wire-frame).

(b) An screen-space 3D Halo rojection visualization. The landmark is on the left side out-of-view (arc). One landmark is marginally within the view (circle) but occluded by another building. One landmark is in the right half-space behind the camera (straight line).

**Figure 8.15:** Examples of visualizing point-of-interests on a mobile device.

For purposes of routing decisions and navigational tasks, this section introduces user awareness strategies (Fig. 8.15 and 8.14), to emphasize 3D objects in virtual 3D city models that are located off-screen and on-screen. These techniques based on 3D adaptations of the 2D halo visualization introduced by Baudisch and Rosenholtz [9]. The present work aims mainly at, but is not limited to, mobile devices as implementation platform. The introduced visualization concepts for navigation and exploration of 3D GeoVE can be applied in real-time. Further, simple occlusion management is applied to maintain the POI's visibility in 3D. The applications of such visualization are extensive. As GPS-enabled devices as well as software applications that use digital maps become more available, so too the applications for POI are also expanding.

### Approaches for the Visualization of 3D Halos

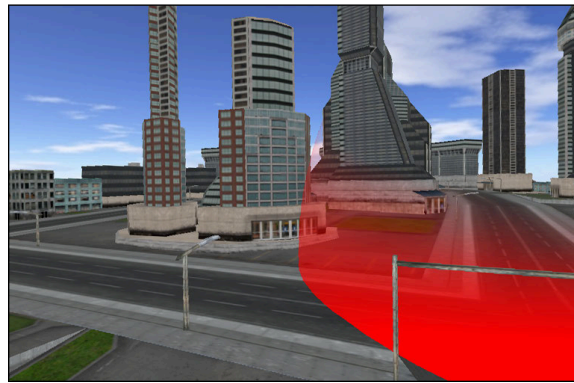
This section presents four different approaches to transfer the idea of 2D Halos [9] to 3D geovirtual environments in order to indicate the position or direction as well as the distances of point-of-interest, which is located outside the view-frustum, to the virtual camera. The basic idea comprises the placement of proxy shapes, which are partially of-out-frame [9, 105]. For 2D maps, the curvature of the visible part of a circular ring that surrounds an off-screen location contains all the information required for locating the ring center [9]. Gustafson et al. uses wedges instead of circles to address the problem of clutter that limits the effectiveness of the Halo approach when used with multiple instances [105].

The presented 3D approaches can be classified into *screen-space* and *object-space* approaches (Fig. 8.15). Screen-space approaches depict 2D proxy object on the view plane, while object-space approaches integrate these objects within the respective 3D scene. This remainder of this section describes three object-space approaches: *3D Halo Sphere*, *3D Halo Circle*, and *3D Halo Billboard*, as well as a single screen-space approach: *3D Halo Projection*. The first approach extends a circular 2D Halo to the third dimension by using a sphere. The circle approach simplifies a 3D sphere using only a 2D circle rendered in 3D object-space parallel a reference plane that approximates the digital terrain model locally. This decreases occlusions and the geometric complexity of the proxy objects. To add additional cues, this concept is extended by the billboard approach that adds an additional approximation of a sphere in the vertical plane. Finally, the 3D projective approach display a circle proxy on the view plane.

The following paragraphs discusses the visualization techniques with respect to the "Halo's usefulness in tasks involving spatial reasoning" [9]. The primary question is whether the efficiency for the 2D geovirtual environment is weakened due to adaptation for 3D GeoVEs.

**3D Halo Sphere Approach** This approach visualizes off-screen locations using sphere as proxy object (Fig. 8.16). It enables a user to estimate the direction and distance to an off-screen location by estimating the center of the sphere based on the partially visible sphere.

To place a sphere within a virtual 3D scene two parameters: the center and the radius are required. The center is determined by projecting the midpoint of the object's axis-aligned bounding box onto the reference plane that approximates the digital terrain model of the virtual 3D city model. Similar to the 2D Halo visualization technique [9], an intrusion border is defined to limit the screen-space occupied by a halo. For three dimensions, an intrusion frustum within the view-frustum is introduced. The radius is computed by the Euclidean distance the sphere center and the closest plane of the intrusion frustum. Using Halo Spheres, the problem of occlusion is expanded from intersecting lines [9] to areas. To reduce possible self-occlusions, a sphere is textured using an alpha-gradient image that is opaque having for the sphere portion near the ground and transparent for the upper portions. Hence, the outline of the Halo is visible, while background objects stay perceptible. 3D Halo Spheres have the advantage to provide intuitive depth cues since a 3D terrain intersection curve is provided. However, in most cases users do not interpret the spheres as proxies for off-screen objects, but try to be integrate them as scene objects. This results in disorientation of the user.



**Figure 8.16:** Example of the 3D Halo Sphere approach that approximates an off-screen location using sphere around the respective center point.

**3D Halo Circle Approach** The *3D Halo Circle* approach enables the user to determine the distance and direction of an off-screen location by extrapolation of a complete circle out of a partly visible circle that is rendered parallel to a reference plane, which approximates the virtual 3D city or landscape model (Fig. 8.17). The circle is rendered around the location's center point. The radius computation for the circle is similar to the 3D Halo Sphere approach.



**Figure 8.17:** Example of the 3D Circle approach that approximates an off-screen location using a horizontal circle around its vertical center axis.

An asset of 3D Halo Circle is the visualization of locations behind the virtual camera. In such case, the user has the impression of being within the circle, which is consistent with the visualization a user would naturally expect. Due to an increasing number of lines, the visualization of more than one off-screen location leads to visual clutter. The different circles will cause a certain level of distraction. Moreover, as the circle radius increases (i.e., the location is farther away), it becomes harder for a user to estimate the complete circle. Further, the circles can occlude the scene objects, which is necessary to support the user's mental distance estimation within a distorted view frustum. Subsequently, a user will probably not be able to mentally complete the radius in all cases, especially if too much of the circles area is occluded.

**3D Halo Billboard Approach** 3D Halo Billboards enables the user to determine the off-screen POI distance and direction by estimating of a complete sphere out of two partially visible circle fragment. The Halo Billboard combines a continuous, opaque circle for non-occluded regions with the dotted-lined, transparent circle in case of occluded regions. This approach extends the previous introduced 3D Circle Halo approach by using crossed billboards (Fig. 8.18) to improve visibility, reduce the lack of depth perception, and to communicate occluded parts more effectively. The approximation of a sphere using crossed billboards further reduces the visual cluttering of 3D Sphere Halo approach.

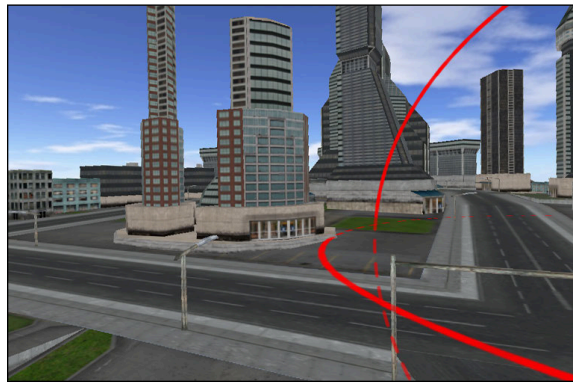
Billboards are used in various applications to depict complex geometry in real-time systems [3]. Using bill-boarding, a geometrical complex 3D object is approximated by an impostor texture. A classic example is the visualization of vegetation objects in 3D landscape models. The orientation of a billboard is adjusted in the way that it usually faces the camera position. To depict a Halo as a textured billboard, three parameters are required: center, radius and camera position. Center and radius are computed analogue to the 3D Halo Sphere approach. The camera position is necessary to align the billboard according to the user's position. To minimize occlusions between Halo Billboards, two different line styles are applied to the billboard. Regarding the portion of a Halo that is not occluded by other objects, the outline of the circle is continuously depicted. In case of occluded portions of the Halo, the circle's outline is depicted with a dotted, partly transparent line style.

Halo Billboards provide a sufficient perceptibility since they are interpreted as separate meta-objects to the scene. As the Halo Billboard only shows the outline of circle, it minimizes the occlusion problem. Due to intersections between lines, Halo Billboards are less complex and do not exhibit self-occlusions. The disadvantage of Halo Billboards is the imprecise depth cue, as the intersection of the circle's outline with the ground is only an intersection point. Together with the second intersection point of the circle that is constructed cognitively, there are only two points that are supposed to complete the terrain intersection area mentally. The impreciseness results from the loss of information as only an approximated line can be constructed from one given intersection point and one cognitively constructed point. In case a foreground object occludes the intersection point, this depth cue is also lost. Regarding Halo Billboards with footprint, the depth cue can be maintained, as the footprint is at least shown as a dotted circular outline in case of occlusions.

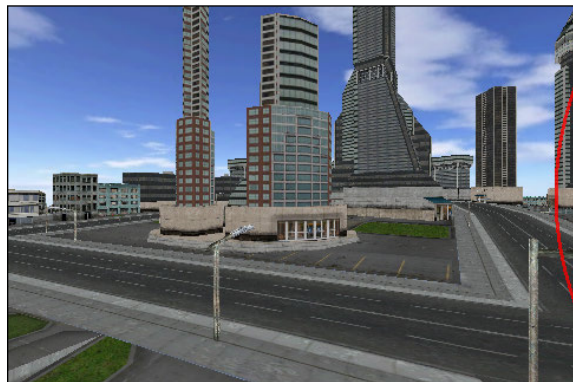
**3D Halo Projection Approach** The 3D Halo Projection approach visualizes off-screen locations with a circle around the projection of an off-screen location on the view plane (Fig. 8.19). Based on the partially visible arc, a user is able to estimate, out-of-view distance and direction (Fig. 8.19). The position of the off-screen location and is projected onto the view plane. In case the projected point is outside of a defined intrusion border, a circle is drawn with a radius large enough to tangent to this border.

Since the direction is not altered due to the projection, the user is directly aware of the location direction. However, it is not possible to map the circle radius to the actual off-screen distance. Instead, the radius communicates only how relatively far is the location out-of-view. In case a POI enters the viewport, a small circle is drawn to indicate the relation between POI and circle. Off-screen locations behind the virtual camera are visualized using a straight line on the viewport side of the respective half-space.

A drawback of this approach is the possible distraction of the user induced through the flipping of that line, caused by POI changing the half-spaces. The possible overlapping of lines, due to a high number of POI represents a further drawback of this approach. However, a major advantage of 3D Halo Projection is the limited viewport distraction, ensured by an intrusion area.



**Figure 8.18:** Example of the 3D Halo Billboard approach that approximates an off-screen location using two orthogonal circles.



**Figure 8.19:** Example of the 3D Halo Projection approach that approximates an off-screen location using a circle on the view plane.



## Interactive Rendering and Occlusion Handling

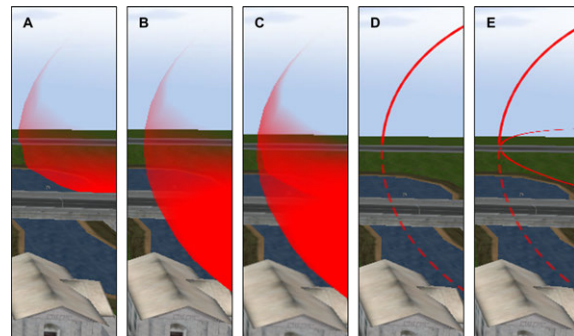
The interactive implementation of the proposed visualization and occlusion management approaches is based on a scene-graph system. Scene-graph traversal techniques process the scene objects accordingly. The computation and rendering of Halos is performed on a per-frame basis using the respective camera settings. The rendering requires two rendering passes. In addition to render the complete scene, the first pass is also used to compute the center and radius for each halo. Based on the computed parameters the proxy objects are added to the scene graph and rendered subsequently. The technique pre-traverses the scene graph and sets the corresponding flags for the removal or wire-frame rendering of an object according to a visibility test. These flags are considered in the subsequent rendering traversal and the scene objects are rasterized accordingly. The visibility test determines whether a scene object occludes a point-of-interest or not. It is based on a ray-intersection test [98] between the AABB, which is computed based on the input geometry, and the rays constructed from the center-of-projection to all visible point-of-interests.

Occlusion is a major problem when adapting the Halo concept to 3D-space, especially if a 3D scene is depicted from a low or pedestrian perspective. This section does not focus on object-object occlusions, which can be compensated using the *Virtual X-Ray Pattern* [257]. In the context of this work, there are basically two types of occlusions that are relevant for the presented object-space approaches: *Halo-Object* and *Halo-Halo* occlusions.

Halo-Object Occlusion addresses the following two cases: (1) A halo occludes other scene objects and (2) a halo is occluded by scene objects in front. The first case can be partially resolved using transparency. Thereby, the Halo is rendered with a transparency gradient varying between opaque for Halo outlines and transparent for the inside area. In case of a Halo Sphere, a semi-transparent appearance is chosen to reduce occlusions regarding objects behind the Halo. To obtain this transparency effect, the sphere is textured using a color texture with an additional transparency channel. For the latter case represents a limitation of the object-based approaches. Drawing the proxy object over the occluder would yield a loss of depth cues. However, the case where of proxy object is completely occluded by object is often the case for pedestrian perspectives and seldom for birds-eye views.

Halo-Halo Occlusions often occurs if a user takes a larger set of relevant POIs into account. As a result, intersections and overlaps of Halos increase. This can lead to visual clutter that makes the perception and interpretation of the Halos cues more difficult. Intersections between Halo Spheres occur between lateral areas of the sphere. In conclusion, they lead to occlusions of three dimensional complexity. To handle these occlusions, the Halo Spheres variant implements three different depth modes (Fig. 8.20, A-C). Using standard depth-testing, the halo's sphere is rendered whereas fragments having a depth smaller than the depth buffer's value are kept. Hence, the for this *normal depth mode*, the sphere appears partially visible since the lower half of the sphere is under the referencing plane.

The upper half can be additionally occluded by objects in the foreground, e.g., buildings in a city model (Fig. 8.20.A). Using the *foreground depth mode*, the sphere is rendered with writing to depth buffer enabled, but with depth-testing turned off. As a result, the sphere appears in foreground and loses depth cues (Fig. 8.20.B). The *mixed depth mode*, implements a combination of the previous both. It depicts the spheres that are rendered with normal depth mode as well as the spheres in foreground. Hence, the spheres in the foreground overlay the rendered sphere and a terrain intersection area becomes visible to give depth cues (Fig. 8.20.C).



**Figure 8.20:** Comparison of 3D Halo variants: Spherical Halos with normal depth mode (A), foreground depth mode (B), and mixed depth mode (C). Billboard Halos without footprint (D) and with footprint (E).

## 8.4 Summary and Future Work

This chapter presents concepts and interactive rendering techniques for the object highlighting in 3D geovirtual environments. All techniques aim at facilitating the pre-attentive cognition of important scene object within virtual 3D city or landscape models.

The object-based approach for scaling landmark object with respect to the position and distance of the virtual camera. It provides a flexible approach for the importance-driven enhancement of landmarks and aims at improving the perceptual and cognitive quality of their display. In particular, the concept can be applied to systems and applications as in the fields of car and pedestrian navigation, disaster management, and spatial data mining. However, this form of landmark highlighting distorts distances between landmarks and cannot guarantee the preservation of topology for multiple landmarks of high scaling factors.

The presented image-based framework enables the highlighting of objects by modifying the appearance of the focus objects or the context respectively by using well know post-processing effects. A GPU based implementation has been presented that enables the image synthesis for various highlighting techniques at interactive frame rates. It supports the integration of various coloring and outline effects for focus objects or regions as well as vignetting and semantic-depth-of-field effects for the context region. This approach is especially suitable 3D GeoVE of high geometrical complexity. Using view-dependend texture atlases, this concept can be extended to multiple overlapping or occluded focus objects and facilitates their individual stylization. Further, different highlighting-techniques have been compared with respect to 3D GeoVE. It has been shown that the capabilities of outline-based and context-based style variant techniques support the preemptive perception, while dealing with disadvantages of glyph-based 3D highlighting techniques. The presented framework can easily be integrated into existing software products, so it could be a promising addition to existing focus+context visualization.

For future work, the presented concepts can be extended by using NPR techniques for the stylization of 3D GeoVE [62]. Of particular interest is the question, if the visual differences between photo-realistic and non-photorealistic depictions are sufficient for the application as highlighting technique. Further, one can investigate the computation of specific highlighting color sets, i.e., given the image-based representation of a virtual 3D scene, what set of colors have the maximal visual differences, and how can they be computed automatically?

However, a user study is required to determine the effectiveness of the presented techniques with respect to certain user and visualization tasks. Based on this study, the proposed framework cab be extended by an automatic approach for the view-dependent selection of highlighting techniques, e.g., based on the distance between object and the virtual camera, or based on the angle with respect to the virtual terrain model.

Finally, the four approaches for the visualization of POIs that are located outside the view frustum are presented. The concept is based on extending existing partially out-of frame techniques to the 3<sup>rd</sup> dimension. A major task for future work is to compensate visual clutter introduced by the on-screen proxy objects. The 3D Circle Halo approach can be improved by using stacks of circles to enable a visual separation for multiple POIs. In such case, different stacking schemes, which define what circle is above the other, as well as priority-based or distance-based schemes are possible. Further, possible cluttering of the projection-based approach could be counterbalanced by a 3D adaptation of the Wedge technique [105]. Nevertheless, to validate the presented approaches, a user evaluation is required that includes a comparison to alternative 2D off-screen location visualization techniques or overview+detail techniques.

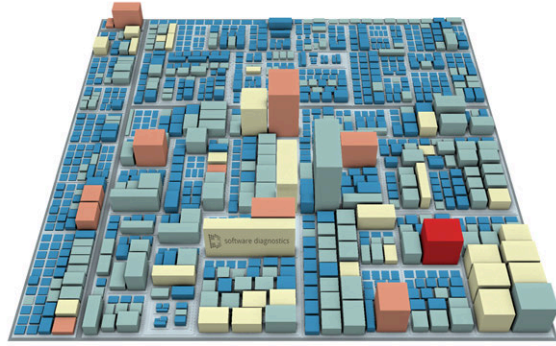
## Chapter 9

# Case Studies and Applications

This chapter presents projects, applications, and case studies of the rendering techniques as well as concepts introduced in the previous chapters. It comprises questions of software integration and the technology transfer of these rendering techniques into other domains or rendering systems. The applications emphasize the need of the presented technologies in various application contexts.

The implementations of the rendering techniques are integrated in a software framework based on a scene-graph based rendering middleware described in [63]. The C++ software library comprises approximately 115,000 lines of code organized in 958 files and 378 classes. Figure 9.1 shows a software map of the system.

The rendering techniques presented in this dissertation can be used in various application contexts. Table 9.1 summarizes applications beyond focus+context visualization. Besides 3D generalization lenses presented in Chapter 4, the concept of volumetric depth sprites applied in combination with the volumetric parity test can be used to implement generalized clipping for cut-away views and relief clipping planes used in the Colonia3D project (Section 9.1). This project also applies the image-based on-screen highlighting approach described in Section 8.2. Further, it uses the image-based deformation approach described in Chapter 6 to generate cylindrical projections. Image-based deformation can also be used to drive multi-projector system for non-planar projection surfaces as described in Section 9.2. Finally, this chapter presents the integration of multi-perspective views for mobile devices based on the Nokia rendering middle ware (Section 9.3). The multi-perspective views are combined with the adaptive landmark scaling described in Section 8.2. Based on this integration, a user evaluation and comparison with established 2D and 3D geovirtual environments are performed.



**Figure 9.1:** Software map of the framework for the rendering techniques for focus+context visualization of 3D geovirtual environments.

**Table 9.1:** Applications of the presented rendering techniques beyond focus+context visualization.

Visualization Technique	Focus+Context Applications	Other Applications
3D Generalization Lenses	3D Lenses	Generalized Clipping Interactive Cut-Away Views Relief Clipping Planes
Dynamic Mapping of Raster Data	2D Surface Lenses Region-of-Interest Highlighting	Dynamic Spatio-Temporal Data Glyph-based Annotations
Image-based Deformation	Multi-focal Fish-eye Lenses	Panoramic Image Generation Non-Planar Projection Surfaces
Image-based Highlighting	Object Highlighting	Ghosted Views

## 9.1 Communication of Digital Cultural Heritage in Public Spaces

The communication of cultural heritage in public spaces such as museums or exhibitions, gain more and more importance during the last years. The possibilities of interactive 3D applications open a new degree of freedom beyond the mere presentation of static visualizations, such as pre-produced video or image data. A user is able to directly interact with 3D geovirtual environments that enable the depiction and exploration of digital cultural heritage artifacts in real-time. However, such technology requires concepts and strategies for guiding a user throughout these scenarios, since varying levels of experiences within interactive media can be assumed. This section presents the application of a subset of the presented rendering techniques integrated



**Figure 9.2:** Permanent exhibition of Roman Cologne at the Roman-Germanic Museum in Cologne.

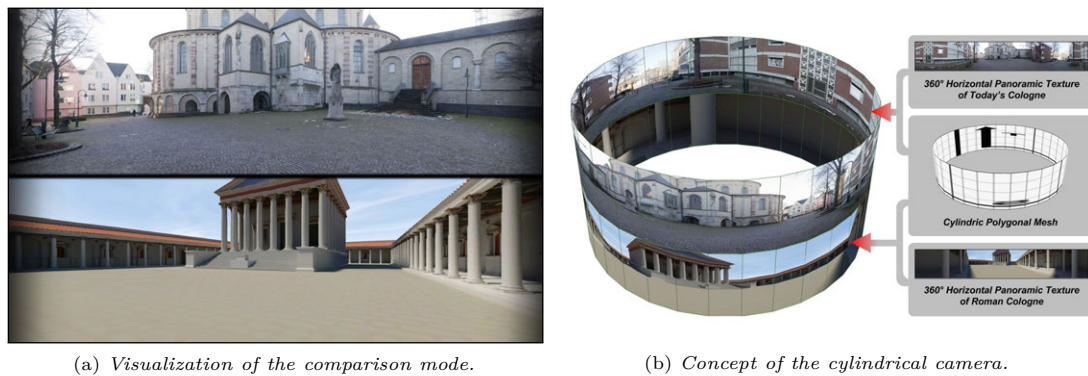
in a visualization framework for the communication of digital cultural heritage in public spaces by example of the project Roman Cologne ([www.colonia3d.de](http://www.colonia3d.de)). It partially describes the results achieved by an interdisciplinary team of archaeologists, designers, and computer graphics engineers with the aim to virtually reconstruct an interactive high-detail virtual 3D city model of the ancient Roman Cologne (Fig. 9.2). The visualization framework offers different *presentation modes* for the effective communication of 3D digital cultural heritage in interactive 3D geovirtual environments [MTK<sup>+</sup>08, TSP<sup>+</sup>10, TSD11, TSP<sup>+</sup>12]. The following three presentation modes of Colonia3D uses four of the rendering techniques presented in this thesis:

**Reconstruction Mode** The visualization of virtual 3D reconstructions can be considered as main purpose for a system that communicates digital cultural heritage. It forms the basis for the remaining two presentation modes. Basically, there are two possibilities for the rendering of this visualization mode: photo-realism vs. abstract visualization. For example, in the case of Roman Cologne people often wish to have more realism in texturing and lighting, but archaeologist concerns that this would imply a "finished" reconstruction to the user. Therefore, an abstract, non-photorealistic, and simple coloring schema is used to communicate that the visualized reconstruction is only one out of many realities.

**Comparison Mode** Based on the reconstruction mode, the comparison mode enables the comparison and dissemination of structural changes over time, i.e., between the reconstruction and today's state. There are several computer graphical approaches and techniques of different implementation complexity to enable the rendering for such a mode, e.g., 3D magic lenses [15] can be used to combine different geometries within a single view. For the visualization framework of Roman Cologne, an image-based approach is applied that enables a side-by-side comparison of locations between the modern Cologne and the ancient version represented by the virtual 3D reconstruction using cylindrical projections. This feature uses image-space deformations described in Chapter 6 to create such cylindrical projections of the virtual 3D reconstruction.

**Findings Mode** The purpose of this mode is the communication of the findings at their respective geolocations, which represent the foundations for the actual reconstructions. The purpose is to enable a user to understand the relation between artifacts and a proposed virtual 3D reconstruction, which was developed by archaeologists and designers. In this scenario, object-highlighting introduced in Section 8.2 was applied for the visualization of the virtual 3D finding objects.

Further the project Colonia3D served as test platform for advanced rendering techniques: *relief clipping planes* combine the concept for the dynamic mapping of raster data (Chapter 5) which



**Figure 9.3:** Visualization concept and components for the comparison mode of Colonia3D. Panoramic images are texture mapped onto a cylinder with a virtual camera placed at the center point.

volumetric depth sprites and the volumetric parity test (Section 4.3 and 4.3). Further, *multiple cut-away views* apply these techniques based on generalized clipping described in Section 4.3 to the data sets of Colonia3D. They can be used for the interactive inspection for expert users.

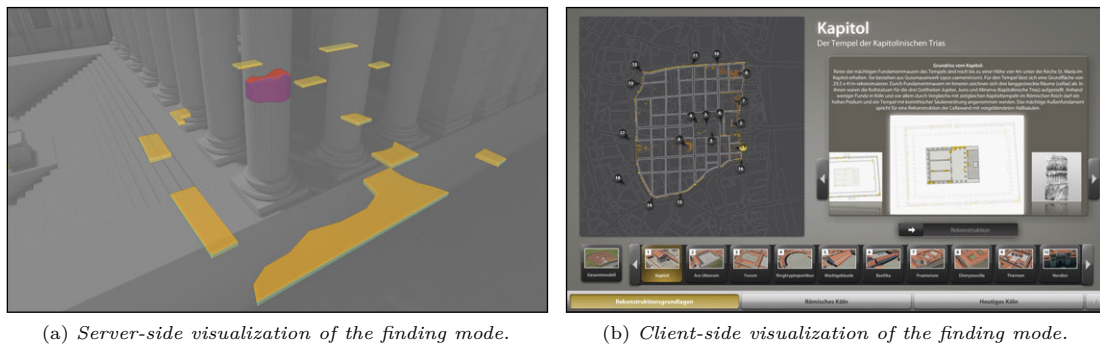
### Panoramic Image Generation and Rendering

This section describes the application of the image-based rendering technique described in Chapter 6 to the rendering framework of Colonia3D (Fig. 9.3(a)). Its integration ensures the functionality of the comparison mode by enabling designers to easily create and modify cylindrical projections on demand. It supports the same stylization of the reconstruction mode for the comparison images.

Instead of using planar images, 360° onmi-directional cylindrical projections are rendered that are mapped onto two cylinders, each rendered using a virtual camera (Fig. 9.3(b)). To navigate within this setup, the user can rotate both cylindrical cameras simultaneously. The application of the rendering technique for image-based deformations has two main reasons: (1) existing DDC tools (e.g., Autodesk 3dsMAX) are not able to render the cylindrical projections because the application cannot handle the geometric complexity of the virtual 3D reconstruction; and (2) to obtain the same visual quality for the comparison mode and the reconstruction mode. During runtime, a user can choose the file name, the resolution of the resulting cylindrical projection, the resolution of the cubemap, as well as the horizontal and vertical field-of-view (fixed to 360° and 90°). Subsequently, the application performs the rendering. Finally, the parameters (e.g., position and orientation) of the virtual camera are dumped to a file, ready to use within the framework.

A straight forward approach for rendering 360° cylindrical projections is to map their corresponding textures onto the surface of a cylinder, with the camera positioned in the middle and directed towards the vertical center of the projection plane. To render the cylindrical projections undistorted, the horizontal and vertical field-of-view (FOV) are adapted respectively. The vertical FOV is adapted to the proportions of the cylinder geometry, while the horizontal FOV is adapted to the aspect ratio of the canvas (screen) and the texture.

As the camera is positioned and directed towards the vertical center of the cylinder, it is expected that the imaginary horizon of the cylindrical projection lies on the same level. A problem occurs if the horizon of the cylindrical projection lies above or below the camera's center point. In this case, visually important lines that are parallel to the viewer's line of sight, e.g., roofs of buildings, become distorted. Physical cameras are able to resolve this issue by using shifted lenses to perspectively correct captured images with a displaced horizon. To simulate a vertical shift when rendering the textures, two approaches are possible. The first approach translates the camera to the horizon level and applies an enlarged vertical field-of-view that tangents the top or bottom edge of the cylinder. Subsequently, the viewport is clipped and stretched to fill the canvas. Alternatively, a second approach can be applied that renders the texture by using an asymmetric projection. Here, the camera is translated to the horizon level, and the projection transformation is adapted by shortening and stretching the top and bottom horizontal clipping planes respectively.

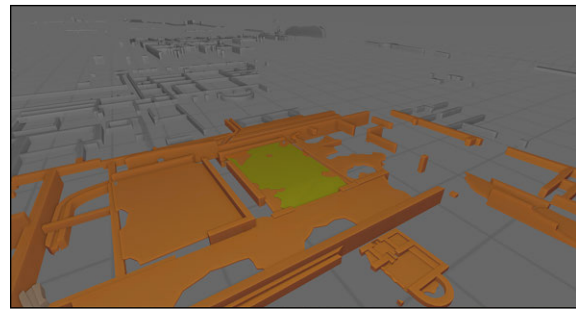


**Figure 9.4:** Server- and client-side visualization of the finding mode. A finding part (red) of a column is highlighted while other findings are displayed in its standard coloring. The context is rendered desaturated.

## Object Highlighting

This section describes the application of object highlighting for the findings mode of Colonia3D. The purpose of this mode is the communication of the findings at their respective geolocations to enable a user to understand the relation between artifact and proposed virtual 3D reconstructions. The applied highlighting facilitates preattentive cognition of artifacts in focus, especially if a viewer faces multiple objects.

Due to the data acquisition and in contrast to the geometric models of the virtual reconstructions, the finding geometry has no textures assigned. Instead, non-photorealistic lighting [99] in combination with unsharp-masking the depth buffer [170] are used to support shape and depth perception. Object highlighting is used to distinguish between selected and unselected objects (Fig. 9.4). In addition thereto, a grid is displayed that approximates the underlying terrain (Fig. 9.5). It facilitates the perception which of the finding objects were originally located above or below the ancient terrain. If switching to the findings mode, the user faces a flow-menu from which he/her can select an active finding (Fig. 9.4(b)). Successively, the application moves the camera closer the findings and highlights it. A slider can be used to blend-in the available reconstructions for the respective location. The coloring can be adjusted by designers. Especially in this application, object highlighting is important, since differences between the 3D projection and the 2D architectural drawing depicted on the touch table usually complicates recognition of the artifacts.



**Figure 9.5:** Object-highlighting in combination with a reference grid in Colonia3D.

The rendering framework of Colonia3D is a scene-graph based system. The rendering pipeline for on-screen object highlighting (Section 8.2) was adapted and integrated in the following way: the root node of the sub-scenegraph representing the findings mode contains a shader program that performs highlighting based on the *activity state* of a scene object. Listing 9.1 shows the respective fragment shader program that performs Gooch Shading [99] used for highlighting. A uniform shader variable (`findingState`) prior to each object in the scene graph indicates if the object should be highlighted during rendering

The rendering framework of Colonia3D is a scene-graph based system. The rendering pipeline for on-screen object highlighting (Section 8.2) was adapted and integrated in the following way: the root node of the sub-scenegraph representing the findings mode contains a shader program that performs highlighting based on the *activity state* of a scene object. Listing 9.1 shows the respective fragment shader program that performs Gooch Shading [99] used for highlighting. A uniform shader variable (`findingState`) prior to each object in the scene graph indicates if the object should be highlighted during rendering

An object, represented as 3D polygonal mesh, can be in one of three modes: *normal*, *highlighted*, and *inactive* (Fig. 9.4(a)). Inactive objects are shown in gray and represent the context of the current location, i.e., all objects that belong to other locations. Objects in normal mode comprises all accessible objects of the current location. During runtime, the rendering framework dynamically changes the configuration of the respective shader variables. This is effective, since the number of finding objects in a specific location is usually small. An alternative integration approach would be

```

1  #version 130
2  uniform int   findingState; // encode object state
3  uniform vec4  warmInactive; uniform vec4  warmNormal; uniform vec4  warmHighlighted;
4  uniform vec4  coolInactive; uniform vec4  coolNormal; uniform vec4  coolHighlighted;
5  uniform float diffuseCool;  uniform float diffuseWarm; uniform vec4  surfaceColor;
6
7  varying float NdotL; // dot product of normal in eye-space and the light vector
8
9  void main(void) {
10     vec4 warmColor = vec4(0.0); vec4 coolColor = vec4(0.0);
11     // select display color
12     switch(findingState) {
13     case 0: { // object is in normal state
14         warmColor = warmNormal; coolColor = coolNormal; }
15     case 1: { // object is highlighted
16         warmColor = warmHighlighted; coolColor = coolHighlighted; }
17     case 2: { // object is displayed inactive
18         warmColor = warmInactive; coolColor = coolInactive; }
19     } //endswitch
20     // compute GOOCH
21     vec3 kCool = min(coolColor.rgb + diffuseCool * surfaceColor.rgb, 1.0);
22     vec3 kWarm = min(warmColor.rgb + diffuseWarm * surfaceColor.rgb, 1.0);
23     // write output
24     gl_FragData[0] = mix(vec4(kCool, coolColor.a), vec4(kWarm, warmColor.a), NdotL);
25     return;
26 }

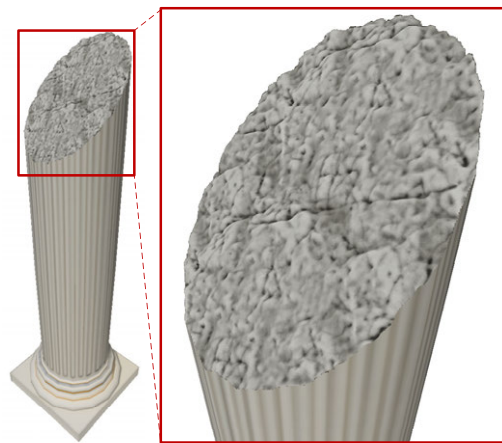
```

**Listing 9.1:** Gooch shader for appearance-based object highlighting in *Colonia3D*.

the application of per-vertex attributes that store an unique object identifier. This approach would increase the overall amount of redundant data to store. In addition to highlighting, the system also automatically creates a camera path to ensure that the highlighted object is always visible to the user. The presented integration approach is easy to implement and scalable over multiple 3D objects in a geovirtual environment. It offers flexible configurations for designers, i.e., to change and adjust the appearance colors of the objects. Combined with automatic camera path planning, the highlighting contributes directly to the success of the findings presentation mode in *Colonia3D*.

### Interactive Relief Clipping Planes and Multiple Cut-Away Views

The concept of clipping planes is well known in computer graphics and can be used to create cut-away views. But clipping against just analytical defined planes is not always suitable for communicating every aspect of such a visualization. For example, in hand-drawn technical illustrations, artists tend to communicate the difference between a cut and a model feature by using non-regular, sketchy cut lines instead of straight ones. To enable this functionality in interactive computer graphics, a technique for rendering *relief clip planes* (RCP) in real-time is presented [TD08d]. Figure 9.6 shows a close-up on a clipped column with applied capping to convey an impression of solid material on the clip surface. Therefore, the clip plane equation is extended with an additional *offset map* (OM), that is represented by a texture map that encodes height values. Subsequently, clipping is performed by varying the clip plane equation with respect to the offset map. Further, a capping technique is used that enables the rendering of caps onto the clipped area to convey the impression of solid material. It avoids a re-meshing of a solid polygonal mesh after clipping is performed. This approach is pixel-precise, applicable in real-time, and takes fully advantage of graphics accelerators.



**Figure 9.6:** Application of relief clipping planes and capping to a column dataset of *Colonia3D*.

```

1  bool clipReliefPlane(in mat4 config,           // configuration matrix
2                     in vec4 point,          // position in eye-space
3                     in sampler2D reliefSampler){ // 2D relief texture
4      // compute plane parametrization in eye space...
5      vec3 O = (gl_ModelViewMatrix * vec4(config[0].xyz, 1.0)).xyz;
6      vec3 A = normalize( gl_NormalMatrix * normalize(config[1].xyz) );
7      vec3 B = normalize( gl_NormalMatrix * normalize(config[2].xyz) );
8      vec3 N = cross(A, B);
9      // project current fragment coordinates on plane
10     vec3 pV = point.xyz - dot(point.xyz - O, N) * N;
11     // compute clip texture coordinates
12     float s = dot(pV - O, A) / length(config[1].xyz);
13     float t = dot(pV - O, B) / length(config[2].xyz);
14     // fetch height... maybe zero
15     float height = texture2D(reliefSampler, vec2(s,t) * config[3].st).x;
16     // compute reference plane
17     float plane = dot(point.xyz, N) - dot(N, O) + (height * config[3].z );
18     return (plane < 0.0 && bool(config[3].w)); // perform clipping
19 }

```

**Listing 9.2:** GLSL implementation to evaluate a relief clipping plane.

**Relief Clipping Planes** Briefly, a  $RCP = (N, P, OM, S)$  is defined by a normal vector  $N$ , an origin  $P$ , the offset map  $OM$ , and a scaling vector  $S = (s_x, s_y, s_z)$ . Given an arbitrary shaped solid mesh and a RCP, clipping is performed on fragment level. For each fragment with a clip space coordinate  $P = (x, y, z)$  the function:

$$\text{clip}(RCP, P) = \begin{cases} \text{true} & P \bullet N - N \bullet O + f(OM, T) \cdot s_z < 0 \\ \text{false} & \text{otherwise} \end{cases} \quad (9.1)$$

$$T = \left( \frac{(P_N - O) \bullet \|U\|}{|U|} \cdot s_x, \frac{P_N - O \bullet \|V\|}{|V|} \cdot s_y \right) \quad (9.2)$$

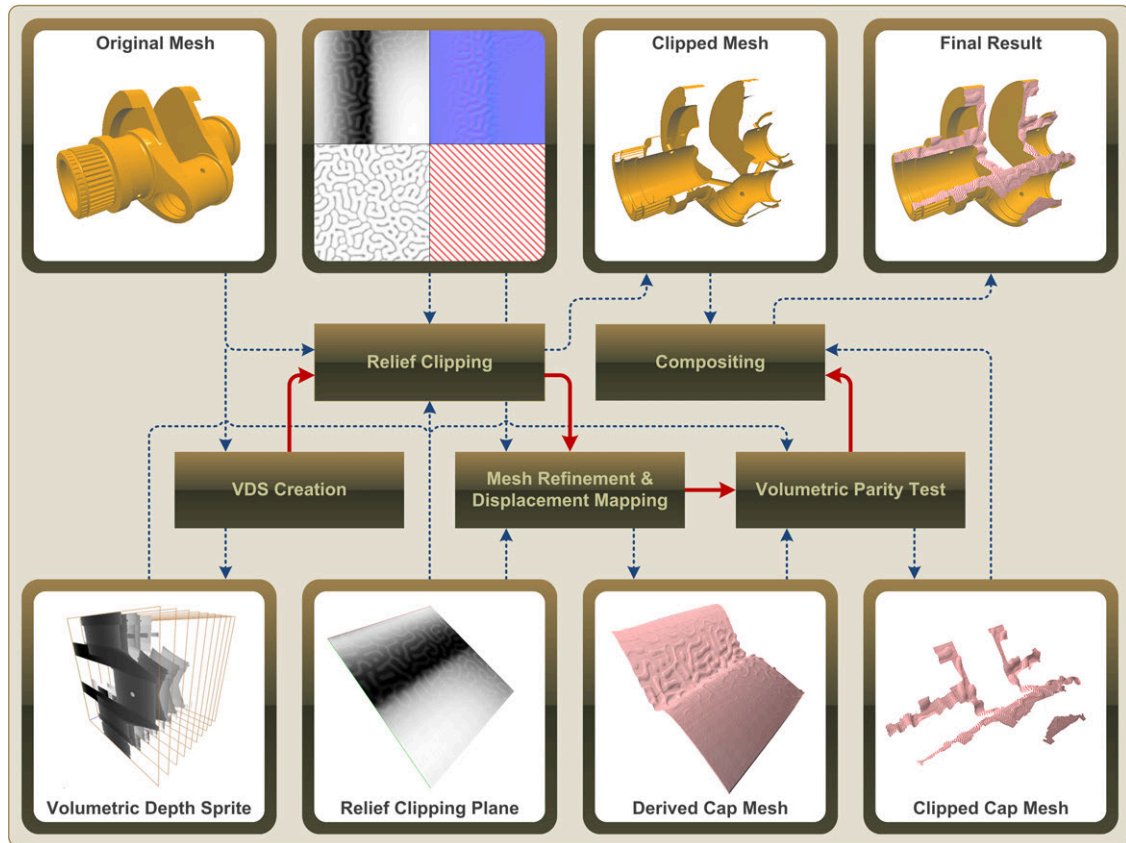
$$P_N = P - ((P - O) \bullet N) \cdot N$$

is evaluated using a fragment shader program (Listing 9.2). Therefore,  $f$  delivers a scalar  $D \in \mathbb{R}$  by (1) generating texture coordinates into the offset map (Eq. 9.2), (2) sampling the  $OM$ , and (3) scaling the resulting height sample by  $s$ . If Equation 9.1 is satisfied, the fragment shader program discards the tested fragment. This step can be performed for a number of clipping planes within a single rendering pass.

**Capping of Solid Meshes** To convey the impression of solid material to visualize the inner structure of an input mesh, it is required to close the cut with a cap surface. Due to the possibly non-regularity of the clip surface, capping techniques based on stencil buffer capabilities [21] cannot be applied. For non-convex shapes, the association of a cap surface to a clipped area cannot be decided in image-space using stencil masks. The proposed image-based approach works for an arbitrary solid. Figure 9.7 (next page) shows the exemplary rendering pipeline for relief clipping planes and capping applied to the domain of technical visualization. Besides the creation of a volumetric depth sprite from the polygonal input mesh in a preprocessing step (Section 4.3), the complete rendering process comprises the following three steps that are performed per frame:

1. Application of relief clipping planes by rendering the solid mesh into the frame buffer with applied relief clipping (Equation 9.1).
2. Creation of a cap mesh from the relief clipping plane. This is implemented using a polygonal cap surface that is derived from the relief clipping plane parametrization. GPU based-mesh refinement [27, 28] is applied to fit the subdivision of the cap mesh to the resolution of the offset map.
3. Clipping of the cap mesh by performing a *volumetric depth test* on a per-fragment basis. It determines if a fragment lies inside the volume and thus is associated with a gap, or if it is





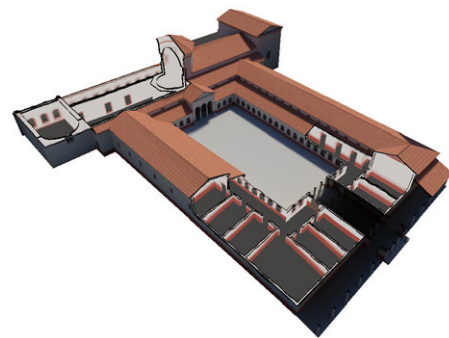
**Figure 9.7:** Conceptual rendering pipeline for relief clipping planes applied to a technical dataset.

located outside the volume and therefore is discarded. In this step per-vertex displacement mapping, and per-fragment shading, and texturing is also performed.

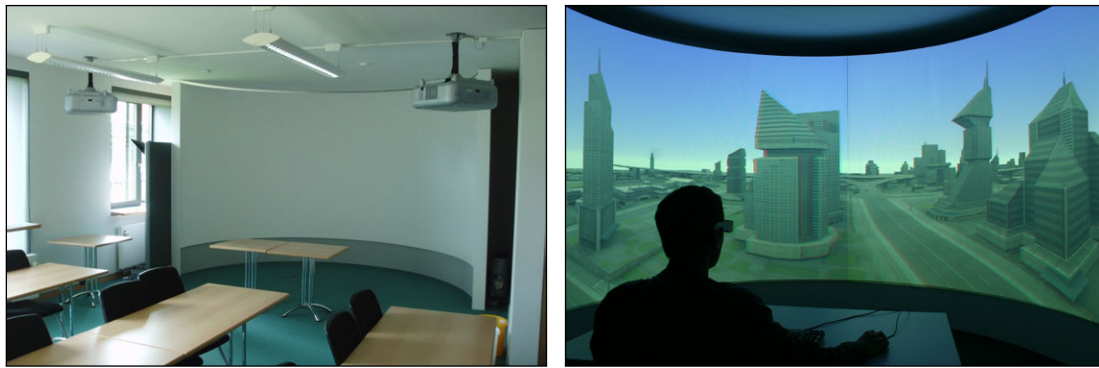
To summarize, this advanced clipping approach is applicable for real-time rendering systems. The hardware-accelerated rendering techniques forms the basis for multiple cut-away views of 3D GeoVE as described next. The technique is limited by two drawbacks: first, it requires an additional data structure (VDS), which is be created during a preprocessing step; and second, to obtain a high visual quality, a sufficient vertex density the cap-surface is required.

**Multiple Cut-away Views** Interactive cut-away views are an important visualization technique that reveals the interior of complex models by clipping either occluding parts or outer layers [161]. Usually, these depictions are static and often created by hand, thus the view point and the displayed cuts are fixed. Interactive cut-away views overcome these drawbacks by enabling the user to choose desired cut planes and cut volumes, while navigating in the 3D geovirtual environment simultaneously.

To enable this functionality in the Colonia3D project, generalized clipping as described in Section 4.3 is applied. It supports clipping against a number of arbitrary solid volumes within a single rendering pass. The polygonal meshes that represent the volumes are designed using standard modeling tools and are positioned into the scene. At runtime, volumetric depth sprites of these volumes are created in a preprocessing step. During rendering, every fragment is tested using the volumetric parity test. The fragment is discarded if its associated position lies inside a specific volume. Figure 9.8 shows an example for applying two different volume cuts to a building



**Figure 9.8:** Cut-away view of the Praetorium data set using two clipping volumes.



(a) System setup for a dual-projector system with a cylindrical projection surface. (b) Passive stereoscopic (anaglyph) rendering using a non-planar projection system.

**Figure 9.9:** Visualization of 3D geovirtual environments using a multi-projector system and a non-planar projection surface.

model in real-time. It reveals parts of the building footprint and internal structures, such as walls and doorways, which otherwise would be hidden to the viewer. The quality of the 3D models (in terms of modeled interior, solid walls, and consistency of polygon orientation), is important for the resulting visual impression. Besides additional configuration issues, the usage of cut-away views introduce a number of challenges and technical implications to the visualization framework: for example, Figure 9.8 shows shading and shadow discontinuities for areas inside and outside the building. These are caused by using a pre-computed lighting approach, which is only valid for views from outside the building. This effect can be compensated partially by using image-based global lighting approaches that approximate ambient occlusion [170].

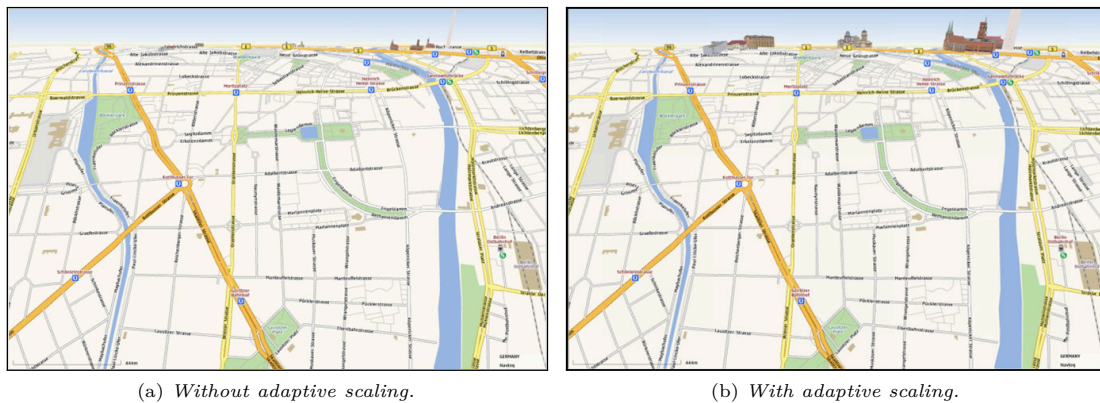
The applied clipping technique is fill-limited, i.e., the runtime performance depends on the number of fragments tested against the volumetric depth sprite. To avoid a significant performance decrease, this technique should be applied only to parts of the 3D scene. To deal with the rendering of hidden but potential visible geometry, a rendering framework has to apply more advanced occlusion culling mechanisms [180].

## 9.2 Non-planar Projection Surfaces

Modern geo-media technology, such as cylindrical projection walls or multi-projector non-planar displays (Fig. 9.9), provide immersive views, enables a more intuitive access of geoinformation to broader audiences and new fields of applications. In contrast to standard 3D perspective views, their implementation is based on non-planar projections. As a characteristic feature, geomedia technology, provides interactivity, a high field-of-view, and high image resolution, which facilitates the immersion of the user [EPTD12]. Further, stereoscopic imaging can improve this immersion effect [TLJD12].

From a cartographic point of view, immersive environments are well-suited as a tool for the effective transmission of complex geoinformation embedded in virtual spaces. Especially psychological depth cues support the intuitive understanding of geoinformation. One main psychological parameter describes the retinal image size. Any restriction of the image size on the retina makes the information transmission less immersive, because any surrounding around the presentation area has an impact on the user's perception. Large projection walls and multi-projector presentation areas enhance retinal image size and therefore support psychological depth cues. Non-planar projections together with stereoscopic imaging further improve the degree of immersion.

In contrast to flat display panels, projector have at least three major advantages: the projector device and the display surface are decoupled, i.e., the depicted image can be larger than the projector device itself. This can be used for the communication of geoinformation to a broader audience. Further, the projected images can be combined, i.e., image of different projectors can be superimposed. Furthermore, the shape of display surface and the displayed images can be



**Figure 9.10:** *Comparative visualization of multi-perspective views combined with adaptive landmark scaling integrated into a navigation system.*

non-planar. The main markets and application of such multi-projector system are control rooms, advertising, visualization, and public information system (e.g., in museums).

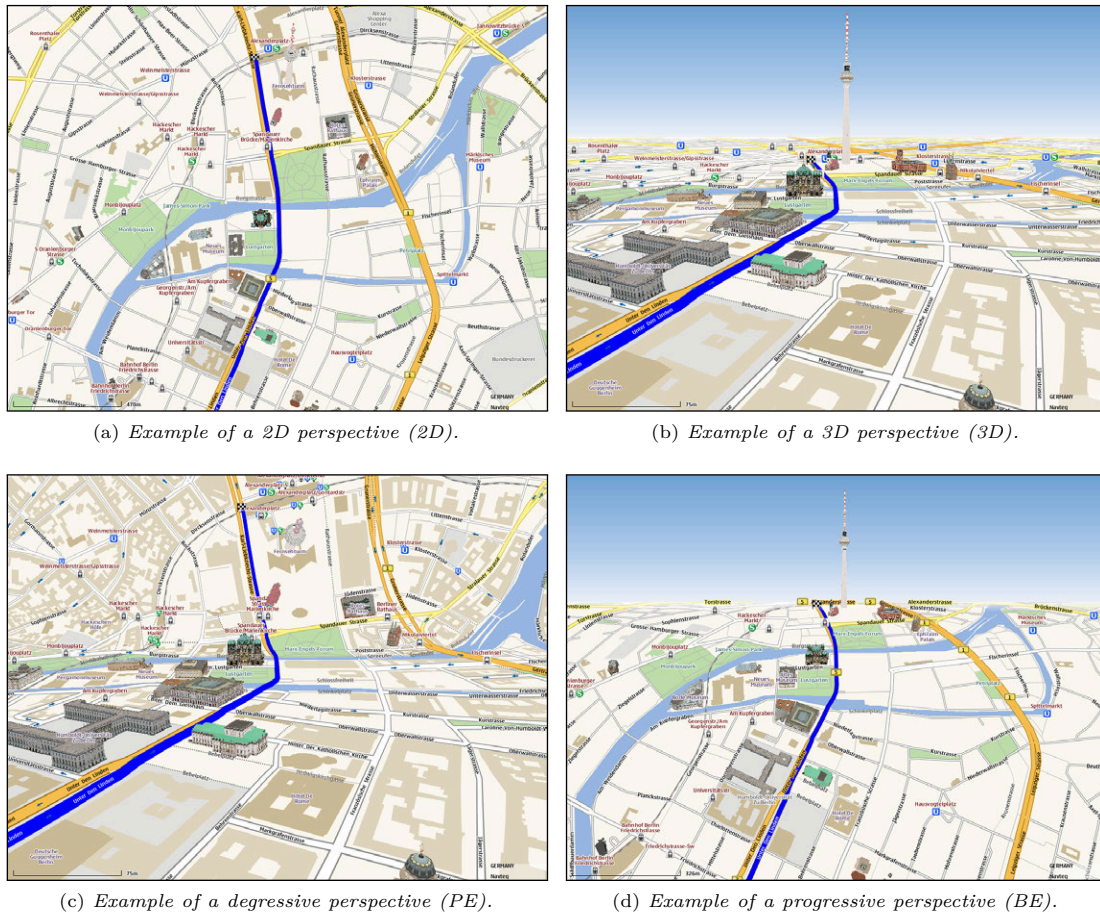
In contrast to flat or planar projection surfaces the usage of non-planar projection surfaces, such as cylindrical or spherical surfaces, enable the application of larger FOV. Thus, a viewer is able to perceive more of the depicted contents of the 3D geovirtual environment. For example, Figure 9.9 shows of a non-planar projection system: a cylindrical wall with a diameter of about four meters that is driven by two projectors. Using a horizontal FOV of  $180^\circ$ , a user inside such projection setup can see what is actually left and right of her/him. In contrast to cave-settings, a continuous non-planar projection surfaces require a specific rendering processes.

The rendering technique for image-based deformations described in Chapter 6 is capable for producing images for such non-planar projection surfaces in real-time. The rendering technique was applied without modifications. The projection is driven by a single graphics accelerator with wide framebuffer of  $2048 \times 768$  pixels resolution. It is split to supply two projectors with a native resolution of  $1024 \times 768$  pixels each. The projectors counterbalance the pincushion distortion automatically. A single cylindrical projection with a horizontal FOV of  $180^\circ$  and a vertical FOV of  $90^\circ$  (see Section 6.2) is used. Further, the setup was also tested using non-planar stereo projections as described in Section 6.5 (Fig. 9.9(b)). This includes the rendering of passive anaglyph stereo, active stereo, and chroma-depth stereo as well.

### 9.3 Multi-perspective Views for Navigation Systems

3D navigation systems and mobile maps are a promising field of application for 3D multi-perspective views. They enable interactive and focus+context visualization by seamlessly combining different visual representations, and are easy to implement. The results presented in this chapter are based on a cooperation with Nokia Gate5 GmbH in Berlin. The specific aim of this project was a prototypical technology transfer and the combination of multi-perspective views (Chapter 7) views with the adaptive landmark scaling (Section 8.2) to mobile devices (Fig. 9.10). Based on the technology transfer, a quantitative and qualitative user study was part of the project. The technology of multi-perspective views is suitable for mobile device due to the following reasons: first, it enables the efficient use of available screen space which is especially important for small displays and small display resolutions; and second, both techniques are easy to implement on GPU and CPU and do not require extensive processing power.

The software architecture of the integration platform was designed to support a wide range of mobile devices, including those which are not capable of performing hardware-accelerated rendering. However, it is possible to transfer the concepts to a CPU-based renderer. The shader-based implementation for multi-perspective views and landmark scaling were converted to per-vertex transformations. Despite minor culling issues, the technology transfer exhibits some limitations: (1) due to missing meta data, all objects marked as landmarks are scaled uniformly. This can lead to



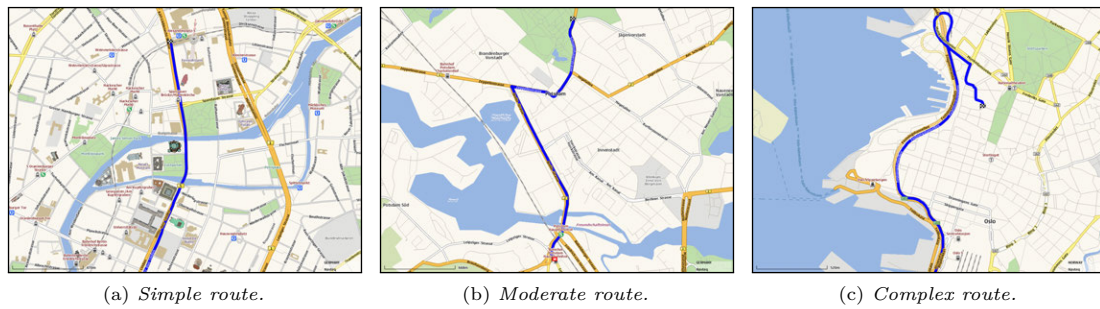
**Figure 9.11:** Exemplary test case for the user evaluation of multi-perspective views.

collisions and visual clutter; and (2), due to missing tessellation features of the integration platform, the vertex density of the scene can be below the required amount. This can be counterbalanced by introducing additional vertices during rendering on client side.

**Test Setup of the User Evaluation** The aim of the user test is to evaluate the acceptance, general preference, and comprehension of multi-perspective views in a navigation scenario. The proposed method is to show side-by-side images using four different perspectives accompanied with the question: "What type of visualization does the user favor to navigate along a route?". Figure 9.11 shows the respective perspectives: an orthographic top-down view (2D) known from common navigation systems, a 3D view using an oblique perspective (3D), a degressive perspective (PE) that shows the top-down view in the lower part of screen, while the upper part shows silhouette of a virtual 3D city model, as well as, a progressive view (BE), that shows the 3D oblique view in the lower part of screen, while the upper part shows top-down view.

Three different route categories of varying complexity are used for the test cases: *simple routes* have a short length and mostly consist of one straight segment (Fig. 9.12(a)). The course of the route is supposed to be easily predicted, even if segments are occluded. *Moderate routes* have an increased length and more turns compared to simple routes (Fig. 9.12(b)). Finally, *complex routes* include a high number of turns and self intersections of the route, e.g., at a motorway junction (Fig. 9.12(c)). The different route categories should reduce the probability for a user to guess the course of the highlighted routes.

During the evaluation, the participants are asked to choose the preferred visualization. The hypothesis that participants would prefer multi-perspective views over classical 2D or 3D views is based on the following rationale: compared to visualization of 3D geovirtual environments using a



**Figure 9.12:** *Exemplary route types of different complexity used for the user test.*

central perspective, multi-perspective views (e.g., a progressive or degressive perspective), make more efficient use of available screen space, reduce the number of different scales and display more elements of the 3D geovirtual environment. Thus, they probably support the user in navigation and orientation tasks more effectively. The test setup was organized as follows: in a sequence of questions, the participant has to choose between two different pictures showing the same route, but with different visualization techniques (e.g., 2D, 3D, progressive, or degressive perspective).

The task is to navigate along a route with the help of a static image from a mobile navigation device. The participant is asked to imagine a navigation task where the highlighted route should be followed. The user study is conducted anonymously and web-based, with 44 participants from 18-55 years with mixed background knowledge about and experience with navigation systems and 3D geovirtual environments.

**Results** Ten routes with a different complexity were prepared that partially contained landmarks. For each route the four visualization configurations are generated. During the user evaluation, 26 image pairs are presented to the user. Each pair depicted the same route using two different perspectives. The user were asked whether they favor one visualization, or they cannot decide for a visualization to favor. The results in Table 9.2 show that 80,7% of the participants favor the orthographic perspective instead of a central perspective. This is reasonable since a 2D map is an established mean for navigation. Furthermore, it can be observed that 76,1% prefer the degressive perspective instead of a central perspective. This indicates a form of acceptance for multi-perspective views for navigation. With the presented technique, it becomes possible to combine the progressive perspective for a low viewing angle with the orthographic perspective for large viewing angles and thus provide the benefits of both visualization in one navigation tool.

**Table 9.2:** *Results for the user evaluation.*

Setup	1 <sup>st</sup> Choice	2 <sup>nd</sup> Choice	No Choice
3D vs. 2D	19%	80%	1%
PE vs. 3D	76%	20%	4%
PE vs. 2D	31%	57%	12%
BE vs. 3D	31%	59%	10%
BE vs. 2D	8%	88%	4%



## Chapter 10

# Conclusions and Future Research

This chapter briefly summarizes and reviews the presented interactive rendering techniques for focus+context visualization of 3D geovirtual environments. It also outlines possible future research directions related to them.

## Conclusions

In general, the presented concepts and rendering techniques allow applications and systems a more effective communication of geoinformation as well as improved ways for orientation and navigation in interactive geovirtual environments by counterbalancing a number of limitations and drawbacks originating from using 3D perspective views for visualization. In particular, the techniques resolve occlusions in near-ground perspectives by using multi-perspective, cut-away, and ghosted views. Further, they are able to reduce multiple cartographic and geometrical scales by combining abstracted and detailed views of virtual 3D city and landscape models within a single image; generalized or abstracted versions of the context region are combined with high-detail geometry of multiple focus regions using 2D and 3D lens-based visualization metaphors. Furthermore, image-based distortions and multi-perspective views facilitate the effective use of available screen space. Finally, the visualization techniques support saliency-guided visualization of landmarks, as well as points, regions, and volumes-of-interest.

The presented rendering techniques, algorithms, and data structures are especially designed for raster-based graphic accelerators and GPU implementations. The flexibility of the programmable graphics pipeline enables the efficient implementation of the visualization techniques suitable that are especially for geometric complex 3D geovirtual environments. Here, the major challenges comprised real-time capabilities of the rendering techniques by exploiting GPU features as well as the integration into existing visualization and rendering pipelines. The application of preprocessing for complex input data, image-based data structures, and the efficient encoding of global scene information into buffer objects allows for performing the rendering in real-time.

The concepts and techniques are capable of using multiple foci, i.e., nesting or overlapping focus regions or volumes. Based on the analysis of existing related and previous work in the area of real-time rendering, digital cartography, and human-computer interaction, this thesis has presented five rendering technologies; these contributions generally include rendering techniques and concepts with applications for but not limited to focus+context visualization.

**3D Generalization Lenses** enable real-time rendering of multiple, overlapping, and nested 3D lenses of arbitrary shapes and contents. The concept is based on a generalized clipping technique that uses a compact image-based data structure (volumetric depth sprite) in combination with a volumetric test, which can be efficiently performed at each programmable pipeline stage. The rendering technique enables the combination of different geometrical representations and variants of 3D GeoVEs (e.g., level-of-abstraction [94]) within a single image. A priority based mapping between lens volumes and geometric contents is used to handle intersected and nested lenses correctly. Further, the image-based data structure offers a wide range of application beyond focus+context visualization such as relief clipping planes or multiple cut-away views. It was applied to data sets of the project Colonia 3D (Section 9.1).

**Dynamic Mapping of Raster Data** overcomes the limitations of graphics hardware with respect to the texturing mechanisms. It presents an extendable concept for mapping dynamic raster-based input data, such as aerial images or image-based simulations results, onto a 3D GeoVE. The implementation is based on projective texture mapping and user-specified color and mask transfer-functions. It provides a fully hardware accelerated rendering technique that enables the display of a high number of projective mappings within a single rendering pass for efficient real-time image synthesis. The capabilities of this flexible technique are used to configure and render multiple overlapping 2D surface lenses, enable style-transfer functions for thematic data, and offers various other applications for 2D, 2.5D, and 3D GeoVE.

**Image-Based Deformations** comprise a parameterizable rendering technique to combine planar and non-planar projections seamlessly. It unifies exiting rendering techniques for the interactive image synthesis of non-planar projections and other artistic projections or 2D image-based distortion effects. Most of all, this technique enables focus+context visualization, such as multi-focal fish-eye views and multiple 2D viewport lenses for 3D GeoVE. Therefore, an efficient image-based rendering technique is introduced that fully exploits graphics hardware and can be applied within a single rendering pass. Despite a lower image-quality compared to geometry-based approaches, the presented techniques is easy to implement and integrate into existing rendering real-time frameworks. It was successfully applied and tested in the project Colonia 3D (Section 9.1). Further, an extension for rendering stereoscopic effects for non-planar projections is introduced to increase immersion for the visualization of 3D GeoVE to broader audiences (Section 9.2).

**Multi-Perspective Views for Spatial 3D Environments** describe a concept and interactive rendering technique of a multi-scale focus+context visualization for 3D GeoVEs similar to Panoramic maps: it enables the seamless combination of different perspectives for simultaneously presenting overview (context) and detailed regions (foci) of virtual 3D city and landscape models. The hardware-accelerated rendering technique can be applied using single or multiple rendering passes based on view-dependent global deformations, which are computed automatically prior to applying a standard projection transformations. It can be easily integrated into existing real-time rendering pipelines and visualization frameworks, and is suitable for the communication of geo-information to a broad audience using cave settings and multi-projector system [EPTD12]. Further, it can represent a fundamental component for future cartography-oriented visualizations [SHTD12, STD11] and comprehensible 3D maps [PSTD12, STKD12]. Furthermore, the concept and rendering technique was ported to the Nokia MOS mobile rendering platform and a user test was conducted that shows the benefits of 3D multi-perspective view over standard 3D perspectives in navigation systems (Section 9.3).

**Highlighting Techniques for 3D Geovirtual Environments** enable interactive saliency-guided focus+context visualization for objects and features located inside or outside the view frustum. Highlighting capability is a fundamental feature of interactive systems. The object-based and image-based rendering techniques use geometric scaling, proxy-objects, and post-processing effects to facilitate pre-attentive cognition of multiple objects- or regions-of-interest. The presented rendering techniques can be easily integrated into existing rendering systems.



## Future Research Directions

The presented techniques form a basis for development of more advanced visualization concepts for 3D GeoVEs. There are numerous aspects for improvement and future work:

**Algorithmic Optimization and Feature Enhancement** The adaptation of the existing implementations to the ongoing development of graphics accelerators is promising to overcome the required preprocessing steps and enable the processing of highly dynamic contents. For example, atomic operations on GPU enable the creation and usage of more efficient data structures. Beside focusing on other rendering primitives such as point clouds, new interaction techniques based on direct input or multi-touch technologies can be researched.

**View-Dependent Parameterization** Especially in 3D geovirtual environments, the view-dependent parameterization and configuration of the visualization techniques comprises a number of potentials for future research [STKD12]. In particular, this includes research on how the parameterization of the respective rendering techniques can be adapted to the virtual camera to ensure optimal results for different viewing settings.

**Transfer to other Visualization Domains** The existing techniques and concepts can be transferred and adapted to new application domains in which the landscape metaphor plays an important role. Besides spatio-temporal geovisualization [BTD12], they can be applied to the software visualization domain to facilitate the readability of software maps [254].

**User Evaluation and Studies** Although some of the presented techniques were included in software system used by non-expert users, an important aspect to focus in future work represents the conduction of qualitative and quantitative user studies to further improve the visualizations techniques once specific applications and systems start to use the presented focus+context techniques.

**Transfer to Indoor-Models** The techniques in this thesis are designed to handle out-door visualization scenarios for 3D GeoVE. With respect to this, research can be conducted for indoor visualization of LOD-4 models. This can include combinations of ghosting, cut-away, and explosion views to support computer generated camera paths, to guarantee that a user is able to preserve the spatial context during the navigation.

**Transfer to Service-oriented Architectures** A major research direction represents the transfer and integration of the visualization techniques into service-oriented architectures to facilitate their application for cloud-computing and mobile devices. This would enable a broad and scalable access using specialized new developed services or combination of existing ones. It would require, for example, a standardization of a general camera model and existing focus+context concepts.



# References

- [1] Pietro Acquisto and Meister Eduard Gröller. A Distortion Camera for Ray Tracing. Technical Report TR-186-2-95-05, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, March 1995.
- [2] Maneesh Agrawala, Denis Zorin, and Tamara Munzner. Artistic Multiprojection Rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 125–136, London, UK, 2000. Springer-Verlag Berlin Heidelberg.
- [3] Tomas Akenine-Möller, Eric Haines, and Natty Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.
- [4] Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. High-Precision Magnification Lenses. In *SIGCHI conference on Human Factors in computing systems*, pages 273–282, Atlanta, États-Unis, April 2010. SIGCHI.
- [5] David Baar. Questions of Focus: Advances in Lens-based Visualizations for Intelligence Analysis. Technical report, IDELIX Software Inc., April 2005.
- [6] William H. Bardel. Depth Cues For Information Design. Master’s thesis, The School of Design, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, May 2001.
- [7] Alan H. Barr. Global and Local Deformations of Solid Primitives. In *SIGGRAPH ’84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 21–30, New York, NY, USA, 1984. ACM Press.
- [8] Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus plus Context Screens: Combining Display Technology with Visualization Techniques. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST ’01, pages 31–40, New York, NY, USA, 2001. ACM P.
- [9] Patrick Baudisch and Ruth Rosenholtz. Halo: A technique for visualizing off-screen objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’03, pages 481–488, New York, NY, USA, 2003. ACM.
- [10] Louis Bavoil and Kevin Myers. Order Independent Transparency with Dual Depth Peeling. Technical Report February, Nvidia Corporation, 2008.
- [11] Salvador Bayarri. Computing Non-Planar Perspectives in Real Time. *Computers & Graphics*, 19(3):431–440, 1995.
- [12] Heinrich Caesar Berann. The World of H.C. Berann. web site, January 2013. <http://www.berann.com/>.
- [13] Bernhard Jenny. Design of a Panorama Map with Plan Oblique and Spherical Projection. In *5th ICA Mountain Cartography Workshop, Bohinj, Slovenia*, pages 121–128, April 2006.
- [14] Jacques Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.

- [15] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and Magic Lenses: The See-Through Interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, New York, USA, 1993. ACM Press.
- [16] Eric A. Bier, Maureen C. Stone, Ken Pier, Ken Fishkin, Thomas Baudel, Matt Conway, William Buxton, and Tony DeRose. Toolglass and Magic Lenses: The see-Through Interface. In *CHI '94*, pages 445–446, New York, NY, USA, 1994. ACM Press.
- [17] Staffan Björk, Lars Erik Holmquist, and Johan Redström. A Framework for Focus+Context Visualization. In *Proceedings of the 1999 IEEE Symposium on Information Visualization, INFOVIS '99*, pages 53–61, Washington, DC, USA, 1999. IEEE Computer Society Press.
- [18] James F. Blinn. A Generalization of Algebraic Surface Drawing. *ACM Transaction Graphics*, 1:235–256, July 1982.
- [19] James F. Blinn. Hyperbolic Interpolation. *IEEE Computer Graphics and Applications*, 12(4):89–94, July 1992.
- [20] David Blythe. The Direct3D 10 System. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 724–734, New York, NY, USA, 2006. ACM Press.
- [21] David Blythe, Tom McReynold, Brad Grantham, Mark J. Kilgard, and Scott R. Nelson. Programming with OpenGL: Advanced Rendering. In A. Rockwood, editor, *SIGGRAPH Course*, New York, NY, USA, 1999. ACM Press.
- [22] Joachim Bobrich. *Ein neuer Ansatz zur kartographischen Verdrängung auf der Grundlage eines mechanischen Federmodells*. Vol. c 455, Deutsche Geodätische Kommission, München, 1996.
- [23] Jürgen Bogdahn and Volker Coors. Using 3D Urban Models for Pedestrian Navigation Support. In *Proceedings of ISPRS Joint Workshop Cityscapes at GeoWeb 2009*, Vancouver, Canada, July 2009.
- [24] Christoph W. Borst, Jan-Phillip Tiesel, Emad Habib, and Kaushik Das. Single-Pass Composable 3D Lens Rendering and Spatiotemporal 3D Lenses. *IEEE Transactions on Visualization and Computer Graphics*, 17(9):1259–1272, 2011.
- [25] Joachim Böttger, Michael Balzer, and Oliver Deussen. Complex Logarithmic Views for Small Details in Large Contexts. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):845–852, September 2006.
- [26] Joachim Böttger, Ulrik Brandes, Oliver Deussen, and Hendrik Ziezold. Map Warping for the Annotation of Metro Maps. *IEEE Computer Graphics and Applications*, 28(5):56–65, September 2008.
- [27] Tamy Boubekeur and Christophe Schlick. Generic Mesh Refinement on GPU. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, HWWS '05*, pages 99–104, New York, NY, USA, 2005. ACM Press.
- [28] Tamy Boubekeur and Christophe Schlick. A Flexible Kernel for Adaptive Mesh Refinement on GPU. *Computer Graphics Forum*, 27(1):102–114, 2008.
- [29] Paul Bourke and Peter Morse. Stereoscopy, Theory and Practice. In *In Workshop Proceedings of VSMM 2007: Exchange and Experience in Space and Place.*, Queensland University of Technology, Brisbane, September 2007.

- [30] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.
- [31] John Brosz, Sheelagh Carpendale, and Miguel Nacenta. The Undistort Lens. *Computer Graphics Forum*, 30(3):881–890, June 2011.
- [32] John Brosz, Faramarz F. Samavati, M. Sheelagh, T. Carpendale, and Mario Costa Sousa. Single Camera Flexible Projection. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 33–42, New York, NY, USA, 2007. ACM Press.
- [33] Stefan Bruckner, M. Eduard Gröller, Klaus Mueller, Bernhard Preim, and Deborah Silver. Illustrative Focus+Context Approaches in Interactive Volume Visualization. In Hans Hagen, editor, *Scientific Visualization: Advanced Concepts*, volume 1 of *Dagstuhl Follow-Ups*, pages 136–162. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2010.
- [34] Stefan Bruckner and Meister Eduard Gröller. Style Transfer Functions for Illustrative Volume Rendering. *Computer Graphics Forum*, 26(3):715–724, September 2007.
- [35] Joachim Böttger, Martin Preiser, Michael Balzer, and Oliver Deussen. Detail-In-Context Visualization for Satellite Imagery. *Computer Graphics Forum*, 27(2):587–596, April 2008.
- [36] Henrik Buchholz. *Real-Time Visualization of 3D City Models*. PhD thesis, Hasso-Plattner-Institut, University of Potsdam, 2006.
- [37] Henrik Buchholz, Johannes Bohnet, and Jürgen Döllner. Smart and Physically-Based Navigation in 3D Geovirtual Environments. In *IV '05: Proceedings of the Ninth International Conference on Information Visualisation*, pages 629–635, Washington, DC, USA, 2005. IEEE Computer Society Press.
- [38] Henrik Buchholz, Jürgen Döllner, Marc Nienhaus, and Florian Kirsch. Real-Time Non-Photorealistic Rendering of 3D City Models. In *Proceedings of the 1st International Workshop on Next Generation 3D City Models*, June 2005.
- [39] Stefano Burigat, Luca Chittaro, and Silvia Gabrielli. Visualizing Locations of Off-Screen Objects on Mobile Devices: A Comparative Evaluation of Three Approaches. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services, MobileHCI '06*, pages 239–246, New York, NY, USA, September 2006. ACM Press.
- [40] Nicholas Burtnyk, Azam Khan, George Fitzmaurice, Ravin Balakrishnan, and Gordon Kurtenbach. StyleCam: Interactive Stylized 3D Navigation using Integrated Spatial & Temporal Controls. In *Proceedings of the 15th annual ACM symposium on User interface software and technology, UIST '02*, pages 101–110, New York, NY, USA, 2002. ACM Press.
- [41] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
- [42] M.S.T. Carpendale and Catherine Montagnese. A Framework for Unifying Presentation Space. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 61–70, New York, NY, USA, 2001. ACM Press.
- [43] Sheelagh Carpendale, John Ligh, and Eric Pattison. Achieving Higher Magnification in Context. In *Proceedings of the 17th annual ACM symposium on User interface software and technology, UIST '04*, pages 71–80, New York, NY, USA, 2004. ACM Press.

- [44] William E. Cartwright. Landmarks and the perception of a space in web-delivered 3d-worlds. In Georg Gartner, William E. Cartwright, and Michael P. Peterson, editors, *Location Based Services and TeleCartography*, Lecture Notes in Geoinformation and Cartography, pages 329–344. Springer, 2007.
- [45] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. MagicSphere: An Insight Tool for 3D Data Visualization. *Computer Graphics Forum*, 13(3):317–328, August 1994.
- [46] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Computing Surveys*, Vol. 41(1):1–31, 2008.
- [47] Jonathan D. Cohen and Dinesh Manocha. *Visualization Handbook*, chapter Model Simplification, pages 393–411. Elsevier, 2005.
- [48] Marcelo Cohen and Ken Brodlie. Focus and Context for Volume Visualization. *Theory and Practice of Computer Graphics*, pages 32–39, 2004.
- [49] Forrester Cole, Doug DeCarlo, Adam Finkelstein, Kenrick Kin, Keith Morley, and Anthony Santella. Directing Gaze in 3D Models with Stylized Focus. In *Proceedings of the 17th Eurographics conference on Rendering Techniques*, EGSR’06, pages 377–387, Aire-la-Ville, Switzerland, Switzerland, 2006. The Eurographics Association.
- [50] Patrick Coleman and Karan Singh. RYAN: Rendering Your Animation Nonlinearly Projected. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, NPAR ’04, pages 129–156, New York, NY, USA, 2004. ACM Press.
- [51] Ferdinand de Saussure. *Grundfragen der allgemeinen Sprachwissenschaft*. Walter de Gruyter, 2001. Fr. original published 1916.
- [52] Paul Debevec, Yizhou Yu, and George Boshokov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. Technical report, Berkeley, CA, USA, 1998.
- [53] Xavier Décoret, Gernot Schaufler, François Sillion, and Julie Dorsey. Multi-layered Impostors for Accelerated Rendering. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Proceedings of Eurographics ’99)*, volume 18, pages 61–73. The Eurographics Association, September 1999.
- [54] Patrick Degener and Reinhard Klein. A Variational Approach for Automatic Generation of Panoramic Maps. *ACM Transactions on Graphics*, 28(1):1–14, 2009.
- [55] Hao Deng, Liqiang Zhang, Jingtao Ma, and Zhizhong Kang. Interactive Panoramic Map-like Views for 3D Mountain Navigation. *Computers & Geosciences*, 37(11):1816–1824, November 2011.
- [56] Paul Joseph Diefenbach. *Pipeline Rendering: Interaction and Realism Through Hardware-based Multi-pass Rendering*. PhD thesis, University of Pennsylvania, 1996.
- [57] Jürgen Döllner and Konstantin Baumann. Geländetexturen als Mittel für die Präsentation, Exploration und Analyse komplexer räumlicher Informationen in 3D-GIS. In V. Coors A. Zipf, editor, *3D-Geoinformationssysteme*, pages 217–230. Wichmann Verlag, 2005.
- [58] Jürgen Döllner, Konstantin Baumann, and Klaus Hinrichs. Texturing Techniques for Terrain Visualization. In *Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000)*, VISUALIZATION ’00, pages 227–234, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.
- [59] Jürgen Döllner and Henrik Buchholz. Continuous Level-of-Detail Modeling of Buildings in Virtual 3D City Models. In *13th ACM International Symposium of Geographical Information Systems (ACM GIS)*, pages 173–181. ACM Press, 2005.

- [60] Jürgen Döllner, Henrik Buchholz, Florian Brodersen, Tassilo Glander, Sascha Jütterschenke, and Alexander Klimetschek. SmartBuildings - A Concept for Ad-Hoc Creation and Refinement of 3D Building Models. In G. Gröger and T. H. Kolbe, editors, *1st International Workshop of 3D City Models*. EuroSDR, June 2005. Online proceedings.
- [61] Jürgen Döllner, Henrik Buchholz, and Haik Lorenz. Ambient Occlusion - ein Schritt zur realistischen Beleuchtung von 3D-Stadtmodellen. *GIS - Zeitschrift für Geoinformatik*, pages 7–13, November 2006.
- [62] Jürgen Döllner, Henrik Buchholz, Marc Nienhaus, and Florian Kirsch. Illustrative Visualization of 3D City Models. In Robert F. Erbacher, Matti T. Roberts, Jonathan C. and Gröhn, and Katy Börner, editors, *Visualization and Data Analysis*, volume 5669 of *Proceedings of the SPIE*, pages 42–51. International Society for Optical Engine (SPIE), 2005.
- [63] Jürgen Döllner and Klaus Hinrichs. The Virtual Rendering System - A Toolkit for Object-Oriented 3D Rendering. In *EduGraphics - CompuGraphics Combined Proceedings*, pages 309–318, 1995.
- [64] Jürgen Döllner and Klaus Hinrichs. A Generic Rendering System. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):99–118, April 2002.
- [65] Jürgen Döllner, Thomas H. Kolbe, Falko Liecke, Takis Sgouros, and Karin Teichmann. The Virtual 3D City Model of Berlin - Managing, Integrating, and Communicating Complex Urban Information. In *25th Urban Data Management Symposium (UDMS)*, 2006. Online proceedings.
- [66] Dominik Göttsche. Playing Ping Pong with Render-To-Texture. Technical report, University of Dortmund, Germany, 2005.
- [67] Shannon Drone. Real-Time Particle Systems On the GPU in Dynamic Environments. In *SIGGRAPH '07*, pages 80–96, New York, NY, USA, 2007. ACM Press.
- [68] Jiangang Duan and Jin Li. Compression of the Layered Depth Image. In *DCC '01*, page 331, Washington, DC, USA, 2001. IEEE Computer Society Press.
- [69] C. Duchêne. Coordinative Agents for Automated Generalisation of Rural Areas. In *Proceedings 5th Workshop on Progress in Automated Map Generalization*, April 2003.
- [70] Juan C. Dürsteler. Focus+context. *Inf@Vis!*, April 2002. <http://www.infovis.net/printMag.php?lang=2&num=85>.
- [71] Jason Dykes, Alan M. MacEachren, and Menno-Jan Kraak. *Exploring Geovisualization*. Elsevier, 2005.
- [72] Max J. Egenhofer and David M. Mark. Naive Geography. In A. U. Frank and W. Kuhn, editors, *COSIT'95: Conference on Spatial Information Theory: A Theoretical Basis for GIS*, number 988 in *Lecture Notes in Computer Sciences (LNCS)*, pages 1–15, Semmering, Austria, 1995. Springer-Verlag Berlin Heidelberg.
- [73] Elmar Eisemann and Xavier Décoret. Fast scene voxelization and applications. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games, I3D '06*, pages 71–78, New York, NY, USA, 2006. ACM Press.
- [74] Mike Eissele, Daniel Weiskopf, and Thomas Ertl. The G<sup>2</sup>-Buffer Framework. In Thomas Schulze, Stefan Schlechtweg, and Volkmar Hinz, editors, *Tagungsband SimVis '04, Magdeburg*, pages 287–298. SCS Publishing House e.V., 2004.

- [75] Birgit Elias. Determination of Landmarks and Reliability Criteria for Landmarks. In *Working Paper of the ICA Workshop on Generalisation and Multiple Representation*, Paris, France, April 2003.
- [76] Geoffrey Ellis, Enrico Bertini, and Alan Dix. The Sampling Lens: Making Sense of Saturated Visualisations. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1351–1354, New York, NY, USA, 2005. ACM Press.
- [77] Stephen R. Ellis. What are Virtual Environments? *IEEE Computer Graphics and Applications*, 14(1):17–22, January 1994.
- [78] Niklas Elmqvist and Philippas Tsigas. A Taxonomy of 3D Occlusion Management for Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1095–1109, January 2008.
- [79] T. Todd Elvins, David R. Nadeau, Rina Schul, and David Kirsh. Worldlets: 3-D Thumbnails for Wayfinding in Large Virtual Worlds. *Presence: Teleoper. Virtual Environ.*, 10(6):565–582, December 2001.
- [80] Cass Everitt. Interactive Order-Independent Transparency. Technical report, NVIDIA Corporation, June 2001.
- [81] Cass Everitt. Projective Texture Mapping. Technical report, NVIDIA Corporation, April 2001.
- [82] Martin Falk, Tobias Schaffhitzel, Daniel Weiskopf, and Thomas Ertl. Panorama Maps with Non-linear Ray Tracing. In *GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 9–16, New York, NY, USA, 2007. ACM Press.
- [83] Margaret M. Fleck. Perspective Projection: the Wrong Imaging Model. Technical Report 95.01, Department of Computer Science, University of Iowa, 1995.
- [84] Theodor Foerster, Lassi Lehto, Tapani Sarjakoski, L. Tiina Sarjakoski, and Jantien E. Stoter. Map Generalization and Schema Transformation of Geospatial Data Combined in a Web Service Context. *Computers, Environment and Urban Systems*, 34(1):79–88, 2010.
- [85] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1990.
- [86] Andrea Forberg. Generalization of 3D Building Data based on a Scale Space Approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(2):104–111, June 2007.
- [87] Georg Fuchs, Matthias Kreuseler, and Heidrun Schumann. Extended Focus & Context for Visualizing Abstract Data on Maps. In *CODATA Prague Workshop Information Visualization, Presentation, and Design*, March 2004. Online Proceedings.
- [88] George W. Furnas. Generalized Fisheye Views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '86, pages 16–23, New York, NY, USA, 1986. ACM Press.
- [89] Verena Giller, Manfred Tscheligi, Johann Schrammel, Peter Fröhlich, Birgit Rabl, Robert Kosara, Silvia Miksch, and Helwig Hauser. Experimental Evaluation of Semantic Depth of Field, a Preattentive Method for Focus+Context Visualization. In M. Rauterberg et al., editor, *Human-Computer Interaction – INTERACT'03*, pages 888–891. IOS Press, 2003.
- [90] Georg Glaeser. Extreme and Subjective Perspectives. In *Topics in Algebra, Analysis and Geometry*, pages 39–51, BPR Médiatanácsadó BT/Budapest, 1999.



- [91] Georg Glaeser and Eduard Gröller. Fast Generation of Curved Perspectives for Ultra-Wide-Angle Lenses in VR Applications. *The Visual Computer*, 15(7/8):365–376, 1999.
- [92] Tassilo Glander, Janett Baresel, and Jürgen Döllner. Überlegungen zum stufenlosen Übergang zwischen verschiedenen generalisierten 3d-stadtmodellrepräsentationen. In *Tagungsband der 3-Ländertagung DGPF*, 2010.
- [93] Tassilo Glander and Jürgen Döllner. Cell-based Generalization of 3D Building Groups with Outlier Management. In *GIS '07: Proceedings of the 15th annual ACMGIS*, pages 1–4, New York, NY, USA, 2007. ACM Press.
- [94] Tassilo Glander and Jürgen Döllner. Techniques for Generalizing Building Geometry of Complex Virtual 3D City Models. In Peter Oosterom, Sisi Zlatanova, Friso Penninga, and Elfriede M. Fendel, editors, *Advances in 3D Geoinformation Systems*, Lecture Notes in Geoinformation and Cartography, pages 381–400. Springer-Verlag Berlin Heidelberg, December 2008.
- [95] Tassilo Glander and Jürgen Döllner. Abstract Representations for Interactive Visualization of virtual 3D City Models. *Computers, Environment and Urban Systems*, 33(5):375–387, 2009.
- [96] R. Steven Glanville. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, chapter Texture Bombing, pages 323–338. Addison-Wesley, 2004.
- [97] Chirs A. Glasbey and Kantilal Vardichand Mardia. A Review of Image Warping Methods. *Journal of Applied Statistics*, 25(2):155–171, 1989.
- [98] Andrew S. Glassner. *Graphics Gems*. Morgan Kaufmann, 1990.
- [99] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A Non-photorealistic Lighting Model for Automatic Technical Illustration. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 447–452, New York, NY, USA, 1998. ACM Press.
- [100] Floraine Grabler, Maneesh Agrawala, Robert W. Sumner, and Mark Pauly. Automatic Generation of Tourist Maps. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–11, New York, NY, USA, 2008. ACM Press.
- [101] Chris Green. Improved Alpha-tested Magnification for Vector Textures and Special Effects. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 9–18, New York, NY, USA, 2007. ACM Press.
- [102] Ned Greene. Environment Mapping and other Applications of World Pojections. *IEEE Computer Graphics and Applications*, 6(11):21–29, November 1986.
- [103] Gerhard Gröger, Thomas H. Kolbe, Angela Czerwinski, and Claus Nagel. *OpenGIS City Geography Markup Language (CityGML) Encoding Standard*. Open Geospatial Consortium Inc., August 2008.
- [104] Sudipto Guha, Shankar Krishnan, Kamesh Munagala, and Suresh Venkatasubramanian. Application of the Two-sided Depth Test to CSG Rendering. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, I3D '03, pages 177–180, New York, NY, USA, 2003. ACM Press.
- [105] Sean Gustafson, Patrick Baudisch, Carl Gutwin, and Pourang Irani. Wedge: Clutter-free Visualization of Off-screen Locations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 787–796, New York, NY, USA, April 2008. ACM Press.

- [106] Sean G. Gustafson and Pourang P. Irani. Comparing Visualizations for Tracking Off-screen Moving Targets. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '07, pages 2399–2404, New York, NY, USA, April 2007. ACM Press.
- [107] Andreas Gustafsson. Interactive Image Warping. Master's thesis, Faculty of Information Technology, 1993.
- [108] Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. *ACM SIGMOD Record*, 14(2):47–57, June 1984.
- [109] Bao H., Ying J., and Peng Q. Non-Linear View Interpolation. In *The Journal of Visualization and Computer Animation*, volume 10, pages 233–241(9). John Wiley & Sons, Ltd., October/December 1999.
- [110] Christian Häberling, Hansruedi Bär, and Lorenz Hurni. Proposed Cartographic Design Principles for 3D Maps: A Contribution to an Extended Cartographic Theory. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 43(3):175–188, 2008.
- [111] Julian Hagenauer and Alexander Zipf. Generating Focus Maps Using Open Standards. In *Extended Abstracts of GIScience*, 2010.
- [112] Güther Hake, Dietmar Grünreich, and Liqiu Meng. *Kartographie*. Walter de Gruyter, Berlin, New York, 8 edition, 2002.
- [113] Mark Harris. Dynamic Texturing. NVIDIA Corporation, May 2004.
- [114] Jan-Henrik Haunert and Leon Sering. Drawing Road Networks with Focus Regions. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2555–2562, December 2011.
- [115] Helwig Hauser. Generalizing Focus+Context Visualization. In Georges-Pierre Bonneau, Thomas Ertl, and Gregory M. Nielson, editors, *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Mathematics and Visualization, pages 305–327. Springer-Verlag Berlin Heidelberg, 2006.
- [116] Wolfgang Heidrich and Hans-Peter Seidel. View-independent Environment Maps. In *HWWS '98: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 39–46, New York, NY, USA, 1998. ACM Press.
- [117] Pedro Hermosilla and Pere-Pau Vázquez. Single Pass GPU Stylized Edges. In F. Serón, O. Rodríguez, J. Rodríguez, and E. Coto, editors, *IV Iberoamerican Symposium in Computer Graphics (SIACG)*, pages 1–8, 2009.
- [118] Jean-Claude Heudin. *Virtual Worlds: Synthetic Universes, Digital Life, and Complexity (New England Complex Systems Institute Series on Complexity)*. Perseus Books Group, January 1999.
- [119] Sébastien Hillaire, Anatole Lécuyer, Rémi Cozot, and Géry Casiez. Depth-of-Field Blur Effects for First-Person Navigation in Virtual Environments. *IEEE Computer Graphics and Applications*, 28(6):47–55, November 2008.
- [120] Sébastien Hillaire, Anatole Lécuyer, Rémi Cozot, and Géry Casiez. Using an Eye-Tracking System to Improve Camera Motions and Depth-of-Field Blur Effects in Virtual Environments. In *Proceedings of IEEE Virtual Reality Conference 2008 (VR 2008)*, pages 47–50, Reno, Nevada, USA, March 2008. IEEE Computer Society Press.
- [121] Kasper Hornbæk and Erik Frøkjær. Reading of Electronic Documents: The Usability of Linear, Fisheye, and Overview+Detail Interfaces. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 293–300, New York, NY, USA, 2001. ACM Press.

- [122] Wei-Hsien Hsu, Kwan-Liu Ma, and Carlos Correa. A Rendering Framework for Multi-scale Views of 3D Models. *ACM Transactions on Graphics*, 30(6):131:1–131:10, December 2011.
- [123] Jing Huang, Tamy Boubekeur, Tobias Ritschel, Matthias Holländer, and Elmar Eisemann. Separable Approximation of Ambient Occlusion. In N. Avis and S. Lefebvre, editors, *Eurographics 2011 - Short papers*, pages 29–32, Llandudno, UK, 2011. The Eurographics Association.
- [124] Takeo Igarashi and Dennis Cosgrove. Adaptive Unwrapping for Interactive Texture Painting. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 209–216, New York, NY, USA, 2001. ACM Press.
- [125] William E. Brandstetter III, Joseph D. Mahsman, Cody J. White, Sergiu M. Dascalu, and Frederick C. Harris Jr. Multi-Resolution Deformation in Out-of-Core Terrain Rendering. *I. J. Comput. Appl.*, 18(4):262–272, 2011.
- [126] Eduard Imhof. *Thematische Kartographie*. Walter de Gruyter, January 1972.
- [127] John P. Isaacs, Daniel J. Gilmour, David J. Blackwood, and Ruth E. Falconer. Immersive and Non-immersive 3D Virtual City: Decision Support Tool for Urban Sustainability. *Journal of Information Technology in Construction*, 16:151–161, January 2011.
- [128] T. J. Jankun-Kelly and Kwan-Liu Ma. MoireGraphs: Radial Focus+Context Visualization and Interaction for Graphs with Visual Nodes. In *Proceedings of the Ninth annual IEEE conference on Information visualization*, INFOVIS'03, pages 59–66, Washington, DC, USA, 2003. IEEE Computer Society.
- [129] Bernhard Jenny and Tom Patterson. Introducing Plan Oblique Relief. *Cartographic Perspectives*, 57(21):21–40, March 2007.
- [130] Helen Jenny, Bernhard Jenny, and Lorenz Hurni. Interactive Design of 3D Maps with Progressive Projection. *The Cartographic Journal*, 47(3):211–221, August 2010.
- [131] Roland Jesse and Tobias Isenberg. Use of Hybrid Rendering Styles for Presentation. In *In Poster Proceedings of WSCG 2003*, 2003.
- [132] Roland Jesse and Tobias Isenberg. Use of Hybrid Rendering Styles for Presentation. In *Poster Proceedings of WSCG 2003*, pages 57–60, 2003. Short Paper.
- [133] Bin Jiang and Christophe Claramunt. A Structural Approach to the Model Generalization of an Urban Street Network. *Geoinformatica*, 8(2):157–171, June 2004.
- [134] Hyungeun Jo, Sungjae Hwang, Hyunwoo Park, and Jung hee Ryu. Mobile Augmented Reality: Aroundplot: Focus+Context Interface for Off-screen Objects in 3D Environments. *Computers & Graphics*, 35(4):841–853, August 2011.
- [135] Markus Jobst and Jürgen Döllner. 3D City Model Visualization with Cartography-Oriented Design. In Manfred Schrenk, Vasily V. Popovich, Dirk Engelke, and Pietro Elisei, editors, *13th International Conference on Urban Planning, Regional Development and Information Society (REAL CORP)*, pages 507–516. CORP - Competence Center of Urban and Regional Planning, 2008.
- [136] Markus Jobst and Jürgen Döllner. Better Perception of 3D-Spatial Relations by Viewport Variations. In *VISUAL '08: Proceedings of the 10th international conference on Visual Information Systems*, pages 7–18, Berlin, Heidelberg, 2008. Springer-Verlag Berlin Heidelberg.
- [137] Sara Johansson and Mikael Jern. GeoAnalytics: Visual Inquiry and Filtering Tools in Parallel Coordinates Plots. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, GIS '07, pages 33:1–33:8, New York, NY, USA, 2007. ACM Press.

- [138] Angie Johnson, Emine M. Thompson, and Kenny R. Coventry. Human Perception, Virtual Reality and the Built Environment. In Ebad Banissi, Stefan Bertschi, Remo Aslak Burkhard, John Counsell, Mohammad Dastbaz, Martin J. Eppler, Camilla Forsell, Georges G. Grinstein, Jimmy Johansson, Mikael Jern, Farzad Khosrowshahi, Francis T. Marchese, Carsten Maple, Richard Laing, Urska Cvek, Marjan Trutschl, Muhammad Sarfraz, Liz J. Stuart, Anna Ursyn, and Theodor G. Wyeld, editors, *Information Visualisation (IV), 2010 14th International Conference*, pages 604–609. IEEE Computer Society, 2010.
- [139] Martin Kada. 3D Building Generalisation. In *Proceedings of 22nd International Cartographic Conference, La Coruña, Spain*, July 2005. CD Proceedings.
- [140] Denis Kalkofen, Erick Mendez, and Dieter Schmalstieg. Interactive focus and context visualization for augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '07*, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society Press.
- [141] Pushpak Karnik, David Cline, Stefan Jeschke, Anshuman Razdan, and Peter Wonka. Route Visualization using Detail Lenses. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):235–247, 2009.
- [142] Alan Keahey. The Generalized Detail-In-Context Problem. In *INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization*, pages 44–51, Washington, DC, USA, 1998. IEEE Computer Society Press.
- [143] Oliver Kersting and Jürgen Döllner. Interactive Visualization of 3D Vector Data in GIS. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems, GIS '02*, pages 107–112, New York, NY, USA, 2002. ACM Press.
- [144] John Kessenich. *The OpenGL Shading Language Language Version: 1.50 Document Revision: 9*. The Khronos Group Inc., July 2009.
- [145] Scott Kircher and Alan Lawrance. Inferred Lighting: Fast Dynamic Lighting and Shadows for Opaque and Translucent Objects. In *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games, Sandbox '09*, pages 39–45, New York, NY, USA, 2009. ACM Press.
- [146] Scott Kirkpatrick, Charles Daniel Gelatt Jr., and Mario P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.
- [147] Florian Kirsch and Jürgen Döllner. OpenCSG: A Library for Image-Based CSG Rendering. In *Proceedings of USENIX 2005*, pages 129–140, 2005.
- [148] Alexander Klippel and Kai-Florian Richter. Chorematic Focus Maps. In Georg Gartner, editor, *Location Based Services & Telecartography*, pages 39–44, Technische Universität Wien, Wien, 2004. Geowissenschaftliche Mitteilungen.
- [149] Robert Kosara, Silvia Miksch, and Helwig Hauser. Semantic Depth of Field. In *INFOVIS '01: Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, pages 97–105, Washington, DC, USA, 2001. IEEE Computer Society Press.
- [150] Robert Kosara, Silvia Miksch, and Helwig Hauser. Focus+Context Taken Literally. *IEEE Computer Graphics and Applications*, Vol. 22(1):22–29, 2002.
- [151] Robert Kosara, Silvia Miksch, Helwig Hauser, Johann Schrammel, Verena Giller, and Manfred Tscheligi. Useful Properties of Semantic Depth of Field for Better F+C visualization. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, pages 205–210, Aire-la-Ville, Switzerland, 2002. The Eurographics Association.
- [152] Menno-Jan Kraak. Some Aspects of Geovisualization. *GeoInformatics*, pages 26–37, 2002.

- [153] Heidi Lam, Ronald A. Rensink, and Tamara Munzner. Effects of 2D Geometric Transformations on Visual Memory. In *Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization (APGV)*, APGV '06, pages 119–126, New York, NY, USA, 2006. ACM Press.
- [154] John Lamping, Ramana Rao, and Peter Pirolli. A Focus+Context Technique based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
- [155] Eugene Lapidous and Guofang Jiao. Optimal Depth Buffer for Low-Cost Graphics Hardware. In *HWWS '99*, pages 67–73, New York, NY, USA, 1999. ACM Press.
- [156] Y. C. Lee, Angela Kwong, Lilian Pun, and Andy Mack. Multi-Media Map for Visual Navigation. *Journal of Geospatial Engineering*, 3(2):87–96, dec 2001.
- [157] Sylvain Lefebvre and Hugues Hoppe. Perfect spatial hashing. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 579–588, New York, NY, USA, 2006. ACM Press.
- [158] Aaron Lefohn. Interactive Visualization of Volumetric Data on Consumer PC Hardware. In *Tutorial in 2003 IEEE Conference of Visualization*, 2003.
- [159] Jed Lengyel and John Snyder. Rendering with Coherent Layers. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 233–242, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
- [160] Ying K. Leung and Mark D. Apperley. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, June 1994.
- [161] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive Cutaway Illustrations of Complex 3D Models. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM Press.
- [162] Baoquan Liu, Li-yi Wei, and Ying-Qing Xu. Multi-Layer Depth Peeling via Fragment Sort. Technical Report MSR-TR-2006-81, Microsoft Research Asia, June 2006.
- [163] Fang Liu, Meng-Cheng Huang, Xue-Hui Liu, and En-Hua Wu. Efficient Depth Peeling via Bucket Sort. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, pages 51–57, New York, NY, USA, 2009. ACM Press.
- [164] Youquan Liu, Xuehui Liu, and Enhua 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 Press.
- [165] Julian Looser, Raphael Grasset, and Mark Billinghurst. A 3D Flexible and Tangible Magic Lens in Augmented Reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–4, Washington, DC, USA, 2007. IEEE Computer Society Press.
- [166] William E. Lorensen. Geometric Clipping Using Boolean Textures. In *Proceedings of the 4th conference on Visualization '93*, VIS '93, pages 268–274, Washington, DC, USA, 1993. IEEE Computer Society Press.
- [167] Haik Lorenz and Jürgen Döllner. Dynamic Mesh Refinement on GPU using Geometry Shaders. In *Proceedings of the 16th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG2008)*, Plzen, Czech Republic, February 2008. UNION Agency - Science Press. Online Proceedings.

- [168] Haik Lorenz and Jürgen Döllner. High-quality non-planar projections using real-time piecewise perspective projections. In Alpesh Kumar Ranchordas, João Madeiras Pereira, Hélder J. Araújo, and João Manuel R. S. Tavares, editors, *Computer Vision, Imaging and Computer Graphics. Theory and Applications. International Joint Conference, VISIGRAPP 2009, Lisboa, Portugal, February 5-8, 2009. Revised Selected Papers*, volume 68 of *Communications in Computer and Information Science*, pages 45–58. Springer-Verlag Berlin Heidelberg, 2010.
- [169] Haik Lorenz, Matthias Trapp, Markus Jobst, and Jürgen Döllner. Interactive Multi-Perspective Views of Virtual 3D Landscape and City Models. In Lars Bernard, Anders Friis-Christensen, and Hardy Pundt, editors, *11th AGILE International Conference on GI Science*, Lecture Notes in Geoinformation and Cartography, pages 301–321. Springer-Verlag Berlin Heidelberg, 2008. Best Paper Award.
- [170] Thomas Luft, Carsten Colditz, and Oliver Deussen. Image Enhancement by Unsharp Masking the Depth Buffer. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1206–1213, New York, NY, USA, 2006. ACM Press.
- [171] Bernd Lutz. *Konzepte für den Einsatz von Virtueller und Erweiterter Realität zur interaktiven Wissensvermittlung*. PhD thesis, Technische Universität Darmstadt, 2004.
- [172] Alan M. Maceachren, Robert Edsall, Daniel Haug, Ryan Baxter, George Otto, Raymon Masters, Sven Fuhrmann, and Liujian Qian. Virtual Environments for Geographic Visualization: Potential and Challenges. In *Proceedings of the ACM Workshop on New Paradigms in Information Visualization and Manipulation*, pages 35–40, New York, NY, USA, 1999. ACM Press.
- [173] Alan M. MacEachren, Mark Gahegan, William Pike, Isaac Brewer, Guoray Cai, Eugene Lengerich, and Frank Hardisty. Geovisualization for Knowledge Construction and Decision Support. *IEEE Computer Graphics and Applications*, 24(1):13–17, January 2004.
- [174] Jock Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2):110–141, April 1986.
- [175] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. *Methods for Non-Linear Least Squares Problems. Informatics and Mathematical Modelling*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2nd edition, 2004.
- [176] Mark J. Kilgard,. NVIDIA OpenGL Extension Specifications. Technical report, NVIDIA Corporation, May 2004.
- [177] Tim Marsh and Peter C. Wright. Using Cinematography Conventions to Inform Guidelines for the Design and Evaluation of Virtual Off-Screen Space. In *Proceedings of AAAI 2000 Spring Symposium Series on Smart Graphics*, pages 123–127. AAAI Press, January 2000.
- [178] Domingo Martín, Salvador Gonzalez García, and Juan Carlos Torres. Observer Dependent Deformations in Illustration. In *NPAA '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 75–82, New York, NY, USA, 2000. ACM Press.
- [179] Ioana M. Martin. Hybrid Transcoding for Adaptive Transmission of 3D Content. In *IEEE International Conference Multimedia and Expo (ICME)*, volume 1, pages 373–376, Thomas J. Watson Res. Center, Hawthorne, NY, USA, August 2002. IBM, IEEE Computer Society Press.
- [180] Oliver Mattausch, Jiří Bittner, and Michael Wimmer. Chc++: Coherent hierarchical culling revisited. *Computer Graphics Forum (Proceedings Eurographics 2008)*, 27(2):221–230, April 2008.

- [181] Liqiu Meng and Andrea Forberg. 3D Building Generalization. In W. Mackaness, A. Ruas, and T. Sarjakoski, editors, *Challenges in the Portrayal of Geographic Information: Issues of Generalisation and Multi Scale Representation*, chapter 11, pages 211–232. Elsevier Science Ltd., Amsterdam, 2006.
- [182] Sebastian Möser, Patrick Degener, Roland Wahl, and Reinhard Klein. Context Aware Terrain Visualization for Wayfinding and Navigation. *Computer Graphics Forum*, 27(7):1853–1860, October 2008.
- [183] Kevin Myers and Louis Bavoil. Stencil Routed A-Buffer. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches*, page 21, New York, NY, USA, 2007. ACM Press.
- [184] Max L. Nelson. Computer Graphics Distortion for IMAX and OMNIMAX Projection. In *Proceedings of Nicograph '83*, pages 137–159, December 1983.
- [185] Petra Neumann and Sheelagh Carpendale. Taxonomy for Discrete Lenses. Technical Report 2003-734-37, Department of Computer Science, University of Calgary, December 2003.
- [186] Marc Nienhaus and Jürgen Döllner. Edge-Enhancement - An Algorithm for Real-Time Non-Photorealistic Rendering. *International Winter School of Computer Graphics, Journal of WSCG*, 11(2):346–353, 2003.
- [187] Marc Nienhaus, Florian Kirsch, and Jürgen Döllner. Sketchy Illustrations for Presenting the Design of Interactive CSG. In *Proceedings of the 14th International Conference Information Visualisation*, volume 0, pages 772–777, Los Alamitos, CA, USA, 2006. IEEE Computer Society.
- [188] Matej Novotny. Visually Effective Information Visualization of Large Data. In *In 8th Central European Seminar on Computer Graphics (CESCG 2004)*, pages 41–48. CRC Press, 2004.
- [189] NVIDIA Corporation. *OpenGL Cube Map Texturing*, May 1999.
- [190] Itzhak Omer, Ran Goldblatt, Karin Talmor, and Asaf Roz. *Complex Artificial Environments - Simulation, Cognition and VR in the Study and Planning of Cities*, chapter Enhancing the Legibility of Virtual Cities by Means of Residents' Urban Image: a Wayfinding Support System, pages 245–258. Springer-Verlag Berlin Heidelberg, 2006.
- [191] John O'Rourke and Greg James. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, chapter Real-Time Glow, pages 343–362. Addison-Wesley, May 2004.
- [192] Tom Patterson. A View From on High: Heinrich Berann's Panoramas and Landscape Visualization Techniques For the US National Park Service. In *Cartographic Perspectives*, number 36, pages 38–65. NACIS, 2000.
- [193] Qunsheng Peng, Xiaogang Jin, and Jieqing Feng. Arc-Length-Based Axial Deformation and Length Preserved Animation. In *CA '97: Proceedings of the Computer Animation*, pages 86–94, Washington, DC, USA, 1997. IEEE Computer Society Press.
- [194] Emil Persson. *ATI Radeon HD 2000 Programming Guide*. AMD Graphics Products Group, June 2007.
- [195] John W. Peterson. Arc Length Parameterization of Spline Curves. Technical report, Taligent, Inc., 1998.
- [196] Emmanuel Pietriga, Olivier Bau, and Caroline Appert. Representation-independent in-place magnification with sigma lenses. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):455–467, May 2010.

- [197] Cyprien Pindat, Emmanuel Pietriga, Olivier Chapuis, and Claude Puech. JellyLens: Content-aware Adaptive Lenses. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, pages 261–270, New York, NY, USA, 2012. ACM Press.
- [198] Jennifer A. Polack-Wahl, Les A. Piegl, and Marc L. Carter. Perception of Images Using Cylindrical Mapping. *The Visual Computer*, 13(4):155–167, 1997.
- [199] Thomas Porter and Tom Duff. Compositing Digital Images. *Proceedings of ACM SIGGRAPH 1984*, 18(3):253–259, January 1984.
- [200] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time Hatching. In *Proceedings of ACM SIGGRAPH 2001*, SIGGRAPH '01, pages 579–584, New York, NY, USA, August 2001. ACM Press.
- [201] Simon Premože. Computer Generation of Panorama Maps. In *Proceedings of the 3rd ICA Mountain Cartography Workshop*, Mt. Hood, Oregon, May 2002.
- [202] Huamin Qu, Haomian Wang, Weiwei Cui, Yingcai Wu, and Ming-Yuen Chan. Focus+Context Route Zooming and Information Overlay in 3D Urban Environments. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1547–1554, 2009.
- [203] Paul Rademacher. View-dependent Geometry. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 439–446, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [204] Wolf-Dieter Rase. Fischaug-Projektionen als kartographische Lupen. In F. Dollinger and J. Strobl, editors, *Angewandte Geographische Informationsverarbeitung*, volume 26 of *Salzburger Geographische Materialien*, pages 115–122. Selbstverlag des Instituts für Geographie der Universität Salzburg, 1997.
- [205] Jiann-Yeou Rau, Liang-Chien Chen, Fuan Tsai, Kuo-Hsin Hsiao, and Wei-Chen Hsu. LOD Generation for 3D Polyhedral Building Model. In *Advances in Image and Video Technology*, pages 44–53, Berlin Heidelberg New York, 2006. Springer-Verlag Berlin Heidelberg.
- [206] Tumasch Reichenbacher and Olivier Swienty. Attention-Guiding Geovisualisation. In *Proceedings of the 10th AGILE International Conference on Geographic Information Science*, Aalborg, Denmark, May 2007. CD Proceedings.
- [207] NVIDIA Developer Relations. *NVIDIA GPU Programming Guide*. NVIDIA Corporation, August 2005. Version 2.4.0.
- [208] Richard Franklin Riesenfeld. *Applications of B-Spline Approximation to Geometric Problems of Computer-aided Design*. PhD thesis, Syracuse University, Syracuse, NY, USA, 1973.
- [209] George G. Robertson, Stuart K. Card, and Jack D. Mackinlay. Information Visualization Using 3D Interactive Animation. *Communications of the ACM*, 36(4):57–71, April 1993.
- [210] George G. Robertson and Jock D. Mackinlay. The Document Lens. In *Proceedings of the 6th annual ACM symposium on User interface software and technology*, UIST '93, pages 101–108, New York, NY, USA, 1993. ACM.
- [211] Anthony C. Robinson. Highlighting Techniques to Support Geovisualization. Technical report, GeoVISTA Center, Department of Geography, The Pennsylvania State University, 2006.
- [212] John F. Roddick, Kathleen Hornsby, and Denise de Vries. A Unifying Semantic Distance Model for Determining the Similarity of Attribute Values. In *Proceedings of the 26th Australasian computer science conference - Volume 16*, ACSC '03, pages 111–118, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.



- [213] David F. Rogers. *An Introduction to NURBS: With Historical Perspective (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, 2000.
- [214] Guodong Rong. *Jump Flooding Algorithm on Graphics Hardware and its Applications*. PhD thesis, National University of Singapore, 2007.
- [215] Guodong Rong and Tiow-Seng Tan. Jump Flooding in GPU with Applications to Voronoi Diagram and Distance Transform. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 109–116, New York, USA, 2006. ACM Press.
- [216] Timo Ropinski. Exploration of Geo-Virtual Environments using 3D Magic Lenses. In *EURESCO-ESF Conference on Geovisualization*, 2004.
- [217] Timo Ropinski and Klaus Hinrichs. An Image-Based Algorithm for Interactive Rendering of 3D Magic Lenses. Technical Report 03/04 - I, FB 10, Institut für Informatik, Westfälische Wilhelms-Universität Münster, 2004.
- [218] Timo Ropinski and Klaus Hinrichs. Real-Time Rendering of 3D Magic Lenses having Arbitrary Convex Shapes. *WSCG*, 12(1-3):379–386, February 2004.
- [219] Timo Ropinski, Klaus Hinrichs, and Frank Steinicke. A Solution for the Focus and Context Problem in Geo-Virtual Environments. In *Proceedings of the 4th ISPRS Workshop on Dynamic and Multi-dimensional GIS (DMGIS05)*, pages 144–149, 2005.
- [220] Timo Ropinski, Frank Steinicke, and Klaus Hinrichs. An Efficient Approach for Emphasizing Regions of Interest in Ray-Casting based Volume Rendering. September 2005.
- [221] Lutz Ross, Birgit Kleinschmit, Jürgen Döllner, and Anselm Kegel. Geovirtual Urban Environments as Media for the Communication of Information related to Managing Urban Land. In *2nd International Conference on Managing Urban Land - Towards More Effective And Sustainable Brownfield Revitalisation Policies.*, pages 577–582. Bundesministerium für Bildung und Forschung, 2007.
- [222] Takafumi Saito and Tokiichiro Takahashi. Comprehensible Rendering of 3-D Shapes. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques, SIGGRAPH '90*, pages 197–206, New York, NY, USA, 1990. ACM Press.
- [223] Pedro V. Sander, Diego Nehab, and Joshua Barczak. Fast Triangle Reordering for Vertex Locality and Reduced Overdraw. *ACM Transactions on Graphics*, 26(3):89:1–89:9, July 2007.
- [224] Scott Schaefer, Travis McPhail, and Joe Warren. Image Deformation using Moving Least Squares. *ACM Transactions on Graphics*, 25(3):533–540, July 2006.
- [225] Arne Schilling and Alexander Zipf. Generation of VRML City Models for Focus based Tour Animations: Integration, Modeling and Presentation of Heterogeneous Geo-data Sources. In *Proceedings of the eighth international conference on 3D Web technology, Web3D '03*, pages 39–ff, New York, NY, USA, 2003. ACM Press.
- [226] Markus Schneider. Uncertainty Management for Spatial Data in Databases: Fuzzy Spatial Data Types. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases, SSD '99*, pages 330–351, London, UK, UK, 1999. Springer-Verlag Berlin Heidelberg.
- [227] Tobias Schwarz, Fabian Hennecke, Felix Lauber, and Harald Reiterer. Perspective+Detail: A Visualization Technique for Vertically Curved Displays. In *Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI '12*, pages 485–488, New York, NY, USA, 2012. ACM Press.

- [228] Stan Sclaroff and Alex Pentland. Generalized Implicit Functions for Computer Graphics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques, SIGGRAPH '91*, pages 247–250, New York, NY, USA, 1991. ACM Press.
- [229] Thomas W. Sederberg and Scott R. Parry. Free-form Deformation of Solid Geometric Models. *SIGGRAPH '86*, 20(4):151–160, 1986.
- [230] Mark Segal and Kurt Akeley. *The OpenGL Graphics System: A Specification (Version 3.2 (Core Profile))*. The Khronos Group Inc., July 2009.
- [231] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, and Paul Haeberli. Fast Shadows and Lighting Effects Using Texture Mapping. *SIGGRAPH '92*, 26(2):249–252, 1992.
- [232] Amir Semmo, Jan Eric Kyprianidis, and Jürgen Döllner. Automated Image-Based Abstraction of Aerial Images. In Marco Painho, Maribel Yasmina Santos, and Hardy Pundt, editors, *Geospatial Thinking*, Lecture Notes in Geoinformation and Cartography, pages 359–378. Springer-Verlag Berlin Heidelberg, 2010.
- [233] Seppo Äyräväinen. 3D Magic Lenses, Magic Lights, and their Application for Immersive Building Services Information Visualization. Technical report, Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, 2003.
- [234] Monika Sester. Generalization Based on Least Squares Adjustment. *International Archives of Photogrammetry and Remote Sensing*, 33:931–938, 2000.
- [235] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered Depth Images. In *SIGGRAPH '98*, pages 231–242, New York, NY, USA, 1998. ACM Press.
- [236] Karan Singh. A Fresh Perspective. In *Graphics Interface*, pages 17–24, 2002.
- [237] Martin Spindler, Marco Bubke, Tobias Germer, and Thomas Strothotte. Camera Textures. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 295–302, New York, NY, USA, 2006. ACM Press.
- [238] Sibylle D. Steck and Hanspeter A. Mallot. The Role of Global and Local Landmarks in Virtual Environment Navigation. *Presence: Teleoper. Virtual Environ.*, 9(1):69–83, February 2000.
- [239] Richard A. Steenblik. The Chromostereoscopic Process: A Novel Single Image Stereoscopic Process. *Proceedings of SPIE: True Three-Dimensional Imaging Techniques & Display Technologies*, 0761:27–34, June 1987.
- [240] Stanislav L. Stoev, Dieter Schmalstieg, and Wolfgang Strasser. The Through-the-Lens Metaphor: Taxonomy and Application. In *Proceedings of the IEEE Conference on Virtual Reality (VR)*, pages 285–286. IEEE Computer Society Press, 2002.
- [241] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. The Movable Filter as a User Interface Tool. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence, CHI '94*, pages 306–312, New York, NY, USA, 1994. ACM Press.
- [242] Rahul Swaminathan, Michael D. Grossberg, and Shree K. Nayar. A Perspective on Distortions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume II, pages 594–601. IEEE Computer Society Press, June 2003.
- [243] Richard Szeliski and Heung-Yeung Shum. Creating Full View Panoramic Image Mosaics and Environment Maps. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

- [244] Shigeo Takahashi, Kenichi Yoshida, Kenji Shimada, and Tomoyuki Nishita. Occlusion-Free Animation of Driving Routes for Car Navigation Systems. *IEEE Transactions on Visualization and Computer Graphics*, 12:1141–1148, 2006.
- [245] Christopher C. Tanner, Christopher J. Migdal, and Michael T. Jones. The Clipmap: A Virtual Mipmap. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 151–158, New York, NY, USA, 1998. ACM Press.
- [246] Natalya Tatarchuk. Real-Time Tessellation on GPU. In *Course 28: Advanced Real-Time Rendering in 3D Graphics and Games*. ACM SIGGRAPH, New York, NY, USA, 2007. ACM Press.
- [247] Andrei Tatarinov. Instanced Tessellation in DirectX10. In *GDC Games Developer Conference*. NVIDIA Corporation, 2008.
- [248] Frank Thiemann and Monika Sester. Segmentation of Buildings for 3D-Generalisation. In *Working Paper of the ICA Workshop on Generalisation and Multiple Representation*, Leicester, UK, 2004.
- [249] Kelvin Thompson. Graphics gems. chapter Alpha blending, pages 210–211. Academic Press Professional, Inc., San Diego, CA, USA, 1990.
- [250] Jan-Phillip Tiesel and Christoph W. Borst. Single-pass Rendering of Composable Volumetric Lens Effects. In *SIGGRAPH '09: Posters*, SIGGRAPH '09, pages 91:1–91:1, New York, NY, USA, 2009. ACM Press.
- [251] Jan-Phillip Tiesel, Kaushik Das, Gary L. Kinsland, Christopher M. Best, Vijay B. Baiyya, and Christoph W. Borst. Composable Volumetric Lenses for Surface Exploration. In *Proceedings of the 2009 IEEE Virtual Reality Conference*, VR '09, pages 291–292, Washington, DC, USA, 2009. IEEE Computer Society.
- [252] W. R. Tobler. A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography*, 46(2):234–240, 1970.
- [253] Christian Tominski, Georg Fuchs, and Heidrun Schumann. Task-Driven Color Coding. In *12th International Conference on IEEE Information Visualization*, pages 373–380. IEEE Computer Society Press, 2008.
- [254] Jonas Trümper and Jürgen Döllner. Extending Recommendation Systems with Software Maps. In *Proceedings of the 3rd International ICSE Workshop on Recommendation Systems for Software (RSSE)*, pages 92–96. IEEE Computer Society, 2012.
- [255] Edward R. Tufte. *Envisioning Information*. Graphic Press, Cheshire, Connecticut, 1990.
- [256] Ken Turkowski. Making Environment Maps from Fisheye Photographs. July 1999.
- [257] Philippas Tsigas Ulf Assarsson, Niklas Elmqvist. Image-Space Dynamic Transparency for Improved Object Discovery in 3D Environments. Technical Report 2006-10, Department of Computer Science & Engineering, Chalmers University of Technology and Goeteborg University, Sweden, 2006.
- [258] Scott Vallance and Paul Calder. Multi-perspective Images for Visualisation. In *VIP '01: Proceedings of the Pan-Sydney area workshop on Visual information processing*, pages 69–76, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.
- [259] Wouter van Oortmerssen. FisheyeQuake/PanQuake, January 2002.

- [260] Eduardo Veas, Raphael Grasset, Ernst Kruijff, and Dieter Schmalstieg. Extended Overview Techniques for Outdoor Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):565–572, April 2012.
- [261] John Viega, Matthew J. Conway, George Williams, and Randy Pausch. 3D Magic Lenses. In *UIST '96*, pages 51–58, New York, NY, USA, 1996. ACM Press.
- [262] Norman G. Vinson. Design Guidelines for Landmarks to Support Navigation in Virtual Environments. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '99, pages 278–285, New York, NY, USA, 1999. ACM.
- [263] Ivan Viola, Armin Kanitsar, and Meister Eduard Groller. Importance-Driven Volume Rendering. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 139–146, Washington, DC, USA, 2004. IEEE Computer Society Press.
- [264] Chuck Walbourn. Direct3D 10 Programming Guide Excerpts. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 369–446, New York, NY, USA, 2007. ACM Press.
- [265] Yu-Shuen Wang and Ming-Te Chi. Focus+Context Metro Maps. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2528–2535, December 2011.
- [266] Yu-Shuen Wang, Tong-Yee Lee, and Chiew-Lan Tai. Focus+Context Visualization with Distortion Minimization. *IEEE Transactions on Visualization and Computer Graphics*, 14:1731–1738, November 2008.
- [267] Colin Ware. *Information Visualization - Perception for Design*. Morgan Kaufmann Series. Morgan Kaufmann Publishers, 2nd edition, 2004.
- [268] Mark J. Ware, Christopher B. Jones, and Nathan Thomas. Automated Map Generalization with Multiple Pperators: A Simulated Annealing Approach. *International Journal of Geographical Information Science*, 17(8):743–769, 2003.
- [269] Matthias Wloka. *ShaderX3*, chapter Improved Batching via Texture Atlases, pages 155–167. Charles River Media, 2005.
- [270] Markus Wolff and Hartmut Asche. Geospatial Modelling of Urban Security: A Novel Approach with Virtual 3D City Models. In *Proceeding sof the international conference on Computational Science and Its Applications, Part I*, ICCSA '08, pages 42–51, Berlin, Heidelberg, 2008. Springer-Verlag Berlin Heidelberg.
- [271] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- [272] Jo Wood, Sabine Kirschenbauer, Jürgen Döllner, A. Lopes, and Lars Bodum. *Exploring Geovisualization*, chapter Using 3D in Geovisualization, pages 295–312. Elsevier, 2005. ISBN 0-08-044531-4.
- [273] Chris Wynn. OpenGL Render-to-Texture. In *GDC Games Developer Conference*. NVIDIA Corporation, October 2002.
- [274] Han-Bing Yan, Shi-Min Hu, and Ralph R. Martin. 3D Morphing using Strain Field Interpolation. *Journal of Computer Science and Technology*, 22(1):147–155, January 2007.
- [275] Yonggao Yang, Jim X. Chen, and Mohsen Beheshti. Nonlinear Perspective Projections and Magic Lenses: 3D View Deformation. *IEEE Computer Graphics and Applications*, pages 76–84, 2005.

- [276] Yonggao Yang, Jim X. Chen, Woosung Kim, and Changjin Kee. Nonlinear Projection: Using Deformations in 3D Viewing. In Jim X. Chen, editor, *Visualization Corner*, pages 54–59. IEEE, March/April 2003.
- [277] Jingyi Yu and Leonard McMillan. A Framework for Multiperspective Rendering. In *Rendering Techniques*, pages 61–68, 2004.
- [278] Jingyi Yu, Leonard McMillan, and Peter Sturm. Multiperspective Modeling, Rendering, and Imaging. In *ACM SIGGRAPH ASIA 2008 courses*, SIGGRAPH Asia '08, pages 14:1–14:36, New York, NY, USA, 2008. ACM Press.
- [279] El-Said M. Zahran, Lloyd D. Bennett, and Martin J. Smith. An Approach to Represent Air Quality in 3D Digital City Models for Air Quality-related Transport Planning in Urban Areas. In W. Tizani, editor, *Proceedings of the International Conference on Computing in Civil and Building Engineering (ICCCBE)*, 2010.
- [280] A. Zanella, M. S. T. Carpendale, and M. Rounding. On the Effects of Viewing Cues in Comprehending Distortions. In *Proceedings of the second Nordic conference on Human-computer interaction*, NordiCHI '02, pages 119–128, New York, NY, USA, 2002. ACM Press.
- [281] Polle T. Zellweger, Jock D. Mackinlay, Lance Good, Mark Stefik, and Patrick Baudisch. City Lights: Contextual Views in Minimal Space. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pages 838–839, New York, NY, USA, 2003. ACM Press.
- [282] Zhe Zhang and David F. McAllister. A Uniform Metric for Anaglyph Calculation. In Andrew J. Woods, Neil A. Dodgson, John O. Merritt, Mark T. Bolas, and Ian E. McDowall, editors, *Proceedings of the SPIE*, volume 6055 of *Stereoscopic Displays and Virtual Reality Systems XIII*, pages 366–377, San Jose, CA, February 2006.
- [283] Xin Zhao, Wei Zeng, Xianfeng David Gu, Arie E. Kaufman, Wei Xu, and Klaus Mueller. Conformal Magnifier: A Focus+Context Technique with Local Shape Preservation. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1928–1941, 2012.
- [284] Kun Zhou, Zhong Ren, Stephen Lin, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Real-Time Smoke Rendering Using Compensated Ray Marching. Technical Report MSR-TR-2007-142, Microsoft Research, September 2007.
- [285] Alexander Zipf and Kai-Florian Richter. Using Focus Maps to Ease Map Reading. *Künstliche Intelligenz*, 4:35–37, 2002.



# Publication Overview

- [BTD12] Stefan Buschmann, **Matthias Trapp**, and Jürgen Döllner. Challenges and Approaches for the Visualization of Movement Trajectories in 3D Geovirtual Environments. In *Proceedings of GIScience workshop on GeoVisual Analytics, Time to Focus on Time*, September 2012. CD Proceedings.
- [EPTD12] Juri Engel, Sebastian Pasewaldt, **Matthias Trapp**, and Jürgen Döllner. An Immersive Visualization System for Virtual 3D City Models. In *20th International Conference on Geoinformatics (GEOINFORMATICS 2012)*, pages 1–7, June 2012.
- [GPTD09] Tassilo Glander, Denise Peters, **Matthias Trapp**, and Jürgen Döllner. 3D Wayfinding Choremes: A Cognitively Motivated Representation of Route Junctions in Virtual Environments. In *12th AGILE International Conference on GI Science*, Lecture Notes in Geoinformation and Cartography, pages 407–427. Springer Berlin Heidelberg, June 2009.
- [GTD07] Tassilo Glander, **Matthias Trapp**, and Jürgen Döllner. A Concept of Effective Landmark Depiction in Geovirtual 3D Environments by View-Dependent Deformation. In *4th International Symposium on LBS and Telecartography*, November 2007. CD Proceedings.
- [GTD08] Tassilo Glander, **Matthias Trapp**, and Jürgen Döllner. Konzepte für die Generalisierung von 3D-Gebäudemodellen. In *Mitteilungen des Bundesamtes für Kartographie und Geodäsie*, volume 41 of *Mitteilungen des BKG*, pages 33–45. Bundesamt für Kartographie und Geodäsie, February 2008. Arbeitsgruppe Automation in Kartographie, Photogrammetrie und GIS.
- [GTD10] Tassilo Glander, **Matthias Trapp**, and Jürgen Döllner. 3D Isocontours – Real-time Generation and Visualization of 3D Stepped Terrain Models. In Stefan Seipel and Hendrik Lensch, editors, *Eurographics 2010 Shortpaper*, pages 17–20, Norrköping, Sweden, May 2010. The Eurographics Association.
- [GTD11] Tassilo Glander, **Matthias Trapp**, and Jürgen Döllner. Concepts for Automatic Generalization of Virtual 3D Landscape Models. In *Proceedings of the annual conference of Digital Landscape Architecture (DLA)*, pages 127–135, May 2011.
- [GTD12] Tassilo Glander, **Matthias Trapp**, and Jürgen Döllner. Concepts for Automatic Generalization of Virtual 3D Landscape Models. *gis.SCIENCE*, 25(1):18–23, 2012.
- [HTGD09] Benjamin Hagedorn, **Matthias Trapp**, Tassilo Glander, and Jürgen Döllner. Towards an Indoor Level-of-Detail Model for Route Visualization. In *Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, MDM '09, pages 692–697, Washington, DC, USA, May 2009. IEEE Computer Society.
- [LTD09] Haik Lorenz, **Matthias Trapp**, and Jürgen Döllner. Interaktive, multiperspektivische Ansichten für geovirtuelle 3D-Umgebungen. *Kartographische Nachrichten*, 4:175–181, September 2009.

- [LTJD08] Haik Lorenz, **Matthias Trapp**, Markus Jobst, and Jürgen Döllner. Interactive Multi-perspective Views of Virtual 3D Landscape and City Models. In Lars Bernard, Anders Friis-Christensen, and Hardy Pundt, editors, *11th AGILE International Conference on GI Science*, Lecture Notes in Geoinformation and Cartography, pages 301–321. Springer Berlin Heidelberg, May 2008. **Best Paper Award**.
- [MTK<sup>+</sup>08] Stefan Maass, **Matthias Trapp**, Jan Eric Kyprianidis, Jürgen Döllner, Michael Eichhorn, Rafael Pokorski, Johannes Bäuerlein, and Henner v. Hesberg. Techniques For The Interactive Exploration Of High-Detail 3D Building Reconstruction Using The Example Of Roman Cologne. In M. Loannides, A. Addison, A. Georgopoulos, and L. Kalisperis, editors, *14th International Conference on Virtual Systems and Multimedia (VSMM 2008)*, pages 223–229. Archaeolingua, October 2008.
- [PSTD12] Sebastian Pasewaldt, Amir Semmo, **Matthias Trapp**, and Jürgen Döllner. Towards Comprehensible Digital 3D Maps. In Markus Jobst, editor, *Service-Oriented Mapping 2012 (SOMAP 2012)*, pages 261–276. Jobstmedia Management Verlag, Wien, 2012.
- [PTD11] Sebastian Pasewaldt, **Matthias Trapp**, and Jürgen Döllner. Multiscale Visualization of 3D Geovirtual Environments Using View-Dependent Multi-Perspective Views. *Journal of WSCG*, 19(3):111–118, February 2011.
- [SHTD12] Amir Semmo, Dieter Hildebrandt, **Matthias Trapp**, and Jürgen Döllner. Concepts for Cartography-Oriented Visualization of Virtual 3D City Models. *PFG Photogrammetrie, Fernerkundung, Geoinformation*, 2012(4):455–465, August 2012.
- [STD11] Amir Semmo, **Matthias Trapp**, and Jürgen Döllner. Ansätze zur kartographischen Gestaltung von 3D-Stadtmodellen. In *31. Wissenschaftlich-Technische Jahrestagung der DGPF*, volume 20 of *Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e. V.*, pages 473–482. Landesvermessung und Geobasisinformation Brandenburg, April 2011.
- [STKD12] Amir Semmo, **Matthias Trapp**, Jan Eric Kyprianidis, and Jürgen Döllner. Interactive Visualization of Generalized Virtual 3D City Models using Level-of-Abstraction Transitions. *Computer Graphics Forum*, 31(3):885–894, June 2012. Proceedings EuroVis 2012.
- [TBPD11] **Matthias Trapp**, Christian Beesk, Sebastian Pasewaldt, and Jürgen Döllner. Interactive Rendering Techniques for Highlighting in 3D Geovirtual Environments. In Thomas Kolbe, Gehard König, and Claus Nagel, editors, *Advances in 3D Geo-Information Sciences*, volume 9 of *Lecture Notes in Geoinformation and Cartography*, pages 197–210. Springer Berlin Heidelberg, January 2011.
- [TD07] **Matthias Trapp** and Jürgen Döllner. Automated Combination of Real-Time Shader Programs. In Paolo Cignoni and Jiri Sochor, editors, *Eurographics 2007 Shortpaper*, pages 53–56. Eurographics, The Eurographics Association, September 2007.
- [TD08a] **Matthias Trapp** and Jürgen Döllner. A Generalization Approach for 3D Viewing Deformations of Single-Center Projections. In José Braz, Nuno Jardim Nunes, and Joao Madeiras Pereira, editors, *International Conference on Computer Graphics Theory and Applications (GRAPP)*, pages 163–170. INSTICC Press, February 2008. **Best Paper Selection**.
- [TD08b] **Matthias Trapp** and Jürgen Döllner. Efficient Representation of Layered Depth Images for Real-time Volumetric Tests. In Ik Soo Lim and Wen Tang, editors, *EG UK Theory and Practice of Computer Graphics (2008) Conference*, pages 9–16. UK Chapter of the Eurographics Association, The Eurographics Association, August 2008.



- [TD08c] **Matthias Trapp** and Jürgen Döllner. Real-Time Volumetric Tests Using Layered Depth Images. In Katerina Mania and Erik Reinhard, editors, *Eurographics 2008 Shortpaper*, pages 235–238. Eurographics, The Eurographics Association, April 2008.
- [TD08d] **Matthias Trapp** and Jürgen Döllner. Relief Clipping Planes for Real-Time Rendering. In *ACM SIGGRAPH Asia 2008 - Sketch Program*, Singapore, December 2008.
- [TD09a] **Matthias Trapp** and Jürgen Döllner. Dynamic Mapping of Raster-Data for 3D Geovirtual Environments. In *Proceedings of the 2009 13th International Conference Information Visualisation*, pages 387–392, Washington, DC, USA, July 2009. IEEE Computer Society Press.
- [TD09b] **Matthias Trapp** and Jürgen Döllner. Generalization of Single-Center Projections Using Projection Tile Screens. In José Braz, Alpesh Kumar Ranchordas, Joao Madeiras Pereira, and Hélder J. Araújo, editors, *Computer Vision and Computer Graphics. Theory and Applications*, volume 24 of *Communications in Computer and Information Science (CCIS)*, pages 55–69. Springer Berlin Heidelberg, January 2009.
- [TD10] **Matthias Trapp** and Jürgen Döllner. Interactive Rendering to Perspective Texture-Atlases. In Stefan Seipel and Hendrik Lensch, editors, *Eurographics 2010 Shortpaper*, pages 81–84, Norrköping, Sweden, May 2010. The Eurographics Association.
- [TGBD08] **Matthias Trapp**, Tassilo Glander, Henrik Buchholz, and Jürgen Döllner. 3D Generalization Lenses for Interactive Focus + Context Visualization of Virtual City Models. In *Proceedings of the 2008 12th International Conference Information Visualisation*, pages 356–361, Washington, DC, USA, July 2008. IEEE Computer Society.
- [TLD09] **Matthias Trapp**, Haik Lorenz, and Jürgen Döllner. Interactive Stereo Rendering For Non-Planar Projections of 3D Virtual Environments. In Alpesh Ranchordas, João Pereira, and Paul Richard, editors, *GRAPP 2009 - 4th International Conference on Computer Graphics Theory and Applications*, pages 199–204. INSTICC Press, February 2009.
- [TLJD12] **Matthias Trapp**, Haik Lorenz, Markus Jobst, and Jürgen Döllner. Enhancing Interactive Non-Planar Projections of 3D Geovirtual Environments with Stereoscopic Imaging. In Manfred Buchroithner, editor, *True-3D In Cartography - 1<sup>st</sup> International Conference on 3D Maps*, Lecture Notes in Geoinformation and Cartography, pages 297–312. Springer Berlin Heidelberg, 2012.
- [Tra07] **Matthias Trapp**. Analysis and Exploration of Virtual 3D-Citymodels using 3D Information Lenses. Diplomarbeit, Hasso Plattner Institut, University Potsdam, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH, Prof.-Dr.-Helmert-Str. 2-3, D-14482 Potsdam, January 2007.
- [Tra09] **Matthias Trapp**. Interaktive Visualisierung des Römischen Kölns. *HPI Magazine*, 6:4–5, 2009. German.
- [TSD11] **Matthias Trapp**, Amir Semmo, and Jürgen Döllner. Colonia3D. In *Tagungsband der 9. Konferenz Kultur und Informatik - Multimediale Systeme*, pages 201–212. Werner Hülsbusch Verlag, May 2011.
- [TSD13] **Matthias Trapp**, Sebastian Schmechel, and Jürgen Döllner. Interactive Rendering of Complex 3D-Treemaps. In *International Conference on Computer Graphics Theory and Applications (GRAPP)*. INSTICC Press, February 2013. to be published.
- [TSHD09] **Matthias Trapp**, Lars Schneider, Norman Holz, and Jürgen Döllner. Strategies for Visualizing Points-of-Interest of 3D Virtual Environments on Mobile Devices. In *6th International Symposium on LBS & TeleCartography*. CD Proceedings, September 2009. **Selected for Journal Publication.**

- [TSL<sup>+</sup>11] **Matthias Trapp**, Lars Schneider, Christine Lehmann, Norman Holz, and Jürgen Döllner. Strategies for Visualizing 3D Points-of-Interest on Mobile Devices. *Journal of Location Based Services (JLBS)*, 5(2):79–99, June 2011.
- [TSP<sup>+</sup>10] **Matthias Trapp**, Amir Semmo, Rafael Pokorski, Claus-Daniel Herrmann, Jürgen Döllner, Michael Eichhorn, and Michael Heinzelmänn. Communication of Digital Cultural Heritage in Public Spaces by the Example of Roman Cologne. In M. Ioannides, editor, *Digital Heritage, Proceedings of 3rd EuroMed Conference*, Lecture Notes in Computer Science (LNCS), pages 262–276. Springer Berlin Heidelberg, November 2010. **Best Paper Award.**
- [TSP<sup>+</sup>12] **Matthias Trapp**, Amir Semmo, Rafael Pokorski, Claus-Daniel Herrmann, Jürgen Döllner, Michael Eichhorn, and Michael Heinzelmänn. Colonia 3D - Communication of Virtual 3D Reconstructions in Public Spaces. *International Journal of Heritage in the Digital Era (IJHDE)*, 1(1):45–74, January 2012. **Selected for Cover Image.**

# Eidesstattliche Erklärung

## Declaration of Academic Honesty

Hiermit versichere ich, dass ich die vorliegende Dissertation ohne Hilfe Dritter und ohne Zuhilfenahme anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I hereby declare in lieu of an oath that this thesis has been written by myself without any external unauthorized help, that it has been neither presented to any institution for evaluation nor previously published in its entirety or in parts.

Potsdam, den 23. Januar 2013

Potsdam, January 23, 2013

---

Matthias Trapp