

**MULTIFACETED FACADE TEXTURES  
FOR 3D CITY MODELS**

**Jürgen BOGDHANN**

**Ph.D. Thesis**

**2012**

# **Multifaceted Façade Textures for 3D City Models**

Jürgen Bogdahn

School of the Built Environment

Salford University

Submitted in Partial Fulfillment of the Requirements of the Degree of

*Doctor of Philosophy (PhD)*

March 2012

# Contents

<b>List of Figures</b>	<b>vii</b>
<b>Listings</b>	<b>x</b>
<b>Acknowledgements</b>	<b>xi</b>
<b>Declaration</b>	<b>xii</b>
<b>Glossary</b>	<b>xiii</b>
<b>Abstract</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Research Problem . . . . .	1
1.2 Contribution to Knowledge . . . . .	3
1.3 Aim & Objectives . . . . .	4
1.4 Structure of the Thesis . . . . .	7
<b>2 Background &amp; Motivation</b>	<b>10</b>
2.1 Relevant Work . . . . .	11
2.1.1 Use of 3D city models . . . . .	11
2.1.2 Data Acquisition . . . . .	16
2.1.3 Procedural Modelling . . . . .	17
2.2 Background . . . . .	19
2.2.1 Six Types of 3D City Models . . . . .	20
2.2.2 Multipurpose Model Management . . . . .	23
2.2.3 Management Systems for Urban Models . . . . .	25
2.2.4 Spatial Data Infrastructures . . . . .	28
2.3 Motivation . . . . .	31
2.3.1 Textures in current 3D City Models . . . . .	31
2.3.2 Photo-Realism and 3D City Models . . . . .	34

**CONTENTS**

---

- 2.3.3 3D City Models & Map Illustration . . . . . 38
  - 2.3.3.1 2D Map Rendering . . . . . 40
  - 2.3.3.2 2.5D Map Rendering . . . . . 41
  - 2.3.3.3 3D Map Rendering . . . . . 42
  - 2.3.3.4 Extension of Map-Space . . . . . 44
- 2.3.4 Level of Detail and Level of Realism . . . . . 45
- 2.3.5 Additional Information in Façade Textures . . . . . 48
- 2.4 Summary . . . . . 50
- 3 Methodology . . . . . 52**
  - 3.1 Constructive Research . . . . . 52
    - 3.1.1 Six Stages of Constructive Research . . . . . 54
  - 3.2 Realizing the Stages of the Constructive Approach . . . . . 55
  - 3.3 Literature Review . . . . . 57
  - 3.4 Proof of Concept by Prototyping . . . . . 60
  - 3.5 Validation by Case Study . . . . . 62
  - 3.6 Summary . . . . . 66
- 4 Texture Content Model . . . . . 68**
  - 4.1 The Image . . . . . 70
  - 4.2 Video . . . . . 72
  - 4.3 The Texture . . . . . 74
  - 4.4 Data Sources for Textures . . . . . 77
  - 4.5 Texture Atlas . . . . . 79
  - 4.6 Modelling of Null-Texture Content . . . . . 81
    - 4.6.1 Zones . . . . . 81
    - 4.6.2 Layer . . . . . 82
      - 4.6.2.1 Description The Pulse Function . . . . . 85
      - 4.6.2.2 Alternative Descriptions . . . . . 90
      - 4.6.2.3 Operations . . . . . 91
    - 4.6.3 The Real-Time Layer . . . . . 93
      - 4.6.3.1 Real-Time Content Integration . . . . . 95

## CONTENTS

---

4.6.3.2	The Player Components . . . . .	97
4.6.3.3	Summary . . . . .	99
4.6.4	Mapping of Information to Zones/Real-time layer . . . . .	99
4.6.4.1	Image to Zone . . . . .	99
4.6.4.2	Measurement Set / Attributes to Zone . . . . .	101
4.6.4.3	Real-Time Content to Real-time Layer . . . . .	101
4.7	The Reconstruction Implementation . . . . .	102
4.8	Integration into Spatial Data Infrastructures . . . . .	104
4.8.1	Flexible Integration into Client-Server Process Chains . . . . .	105
4.8.2	Scenario 1: Server Side Texture reconstruction . . . . .	105
4.8.3	Scenario 2: Client Side Reconstruction and Processing . . . . .	107
4.8.4	Scenario 3: Pre-processing for Clients with Special Requirements . . . . .	109
4.9	Discussion . . . . .	112
4.9.1	Pulse Function Approach . . . . .	112
4.9.2	A Similar Approach for Thematic Visualization . . . . .	113
4.9.3	Real-Time Layer . . . . .	115
4.9.4	Integration into (3D-) SDIs . . . . .	116
<b>5</b>	<b>Proof of Concept</b> . . . . .	<b>117</b>
5.1	The Rendering Pipeline . . . . .	118
5.2	The Programmable Elements of the Rendering Pipeline . . . . .	121
5.3	Data Management and 3D Viewer Integration . . . . .	124
5.4	The Pre-Processing Step implemented as Utility Functions . . . . .	128
5.5	Data Texture . . . . .	129
5.6	The Texture Atlas . . . . .	134
5.7	The Rendering Concept . . . . .	135
5.7.1	The Construction of Façade Texture Content . . . . .	136
5.7.2	The Real-Time Content . . . . .	141
5.7.3	Video . . . . .	142
5.7.4	Metering Data . . . . .	144

## CONTENTS

---

5.7.5	Rendering Real-Time Data embedded in the Façade Texture . . .	145
5.8	Summary . . . . .	148
5.8.1	Performance . . . . .	149
5.8.2	Scalability . . . . .	151
5.8.3	Real-Time Content . . . . .	153
<b>6</b>	<b>Pedestrian Navigation using 3D City Models – a Case-Study</b>	<b>154</b>
6.1	Concepts of Navigation - Cars compared to Pedestrians . . . . .	156
6.2	Landmarks . . . . .	161
6.3	The MoNa3D Project . . . . .	163
6.3.1	The MoNa3D Spatial Data Infrastructure . . . . .	165
6.3.2	Storage Efficient Synthetic Textures . . . . .	167
6.3.3	The Smartphone Client Prototype . . . . .	169
6.3.4	Summary . . . . .	169
6.4	Visualization of Buildings and Landmarks for Pedestrian Navigation . . .	170
6.4.1	Level of Realism . . . . .	170
6.4.2	Navigation Hints . . . . .	174
6.4.3	Shop Signs . . . . .	179
6.4.4	Link to Indoor Navigation . . . . .	180
6.5	Summary . . . . .	182
<b>7</b>	<b>Conclusions &amp; Future Work</b>	<b>186</b>
7.1	Conclusions . . . . .	186
7.1.1	Objective 1: Flexible Description . . . . .	186
7.1.2	Objective 2: Adjustability and additional Information . . . . .	188
7.1.3	Objective 3: Real-time Content Changes . . . . .	189
7.1.4	Objective 4: 3D-SDI Integration . . . . .	191
7.1.5	Objective 5: Evaluation of the presented Approach . . . . .	193
7.1.6	Summary . . . . .	194
7.2	Future Work . . . . .	195
7.2.1	Layer/Zone Model . . . . .	195
7.2.2	Data Acquisition . . . . .	196

## CONTENTS

---

7.2.3	Real-Time Layer . . . . .	197
7.2.4	3D Spatial Data Infrastructures . . . . .	198
7.2.5	Pedestrian Navigation . . . . .	198
	<b>References</b>	<b>199</b>

# List of Figures

1.1	The Thesis Structure . . . . .	8
2.1	Example for 3D building models in a Nav-System . . . . .	13
2.2	Visualization of air pollution . . . . .	14
2.3	Visualization of simulation scenarios . . . . .	15
2.4	Oblique imagery for generating 3D city models . . . . .	17
2.5	Façade grammars . . . . .	18
2.6	Type-3 Model . . . . .	22
2.7	Type-6 Model . . . . .	23
2.8	CAT3D as basis for OGC Services . . . . .	28
2.9	Schema of Surface Properties . . . . .	33
2.10	Simple World Map . . . . .	40
2.11	2.5D-Surfaces as Map-Space . . . . .	41
2.12	3D Map with Tree-Icons . . . . .	42
2.13	3D Map-Terrain with Building-Icons . . . . .	43
2.14	Extension of Map-Space . . . . .	44
2.15	CityGML LoD concept . . . . .	46
3.1	Constructive Research . . . . .	53
3.2	Research Stages . . . . .	56
4.1	Image coordinate system . . . . .	71
4.2	Texture Mapping . . . . .	76
4.3	Null-Texture Concept . . . . .	77
4.4	Texture Atlas Concept . . . . .	80
4.5	Using BSP to create texture atlas . . . . .	81
4.6	Defining a Zone . . . . .	82
4.7	The Layer Concept . . . . .	84
4.8	The Level of Realism concept . . . . .	85



## LIST OF FIGURES

---

4.9	Pulse functions overlap . . . . .	87
4.10	Multiple use of Pulses . . . . .	88
4.11	False Zones . . . . .	89
4.12	Painter's Algorithm . . . . .	92
4.13	Thermal Image Content vs Façade Structure . . . . .	94
4.14	Real-Time Layer Concept . . . . .	96
4.15	Real-Time Layer Implementation Concept . . . . .	98
4.16	Possible reconstruction levels for texture content . . . . .	106
4.17	Portrayal W3DS . . . . .	107
4.18	Thin Client using Shader on GPU . . . . .	109
4.19	Fat Client using Shader on GPU . . . . .	111
4.20	Surface Properties Tool . . . . .	114
5.1	OpenGL Operations . . . . .	119
5.2	OpenGL Programmable Processors . . . . .	122
5.3	Fragment processor inputs and outputs . . . . .	123
5.4	The CAT3D Architecture . . . . .	125
5.5	The CAT3D Data Model . . . . .	126
5.6	The Extended Texture Model . . . . .	127
5.7	Layer encoding into Data Texture . . . . .	130
5.8	Pulse Rows in DataTexture . . . . .	132
5.9	Bounding Box for Zones in one Layer . . . . .	137
5.10	Two level rendering . . . . .	140
5.11	Video-Player implementation . . . . .	143
5.12	Pachube-Player implementation . . . . .	145
5.13	Real-Time Layer Rendering . . . . .	147
5.14	Prototype Rendering Performance Measurement . . . . .	149
6.1	Example for decision support in car navigation systems . . . . .	157
6.2	Lines to Approximate Square . . . . .	158
6.3	Kunstmuseum Stuttgart . . . . .	162
6.4	MoNa3D Architecture . . . . .	165

## LIST OF FIGURES

---

6.5	Emphasizing landmarks in 3D Scene . . . . .	172
6.6	Sketch Rendering . . . . .	173
6.7	The Pre-verbal Message . . . . .	175
6.8	Visual Hints for Landmarks . . . . .	176
6.9	Navigation Hints in Façade . . . . .	177
6.10	Bubbles for linking nav-instruction to landmark . . . . .	178
6.11	Highlights for façade elements . . . . .	181
6.12	Navigation hint for correct entrance and floor of a building . . . . .	183
7.1	Window Extraction Example . . . . .	197

# Listings

4.1 Pseudo code for screen scanning (Heckbert, 1986) . . . . . 75

# Acknowledgements

First and foremost I would like to thank my supervisor Mr. Andy Hamilton, head of the Virtual Planning Group at Salford University and my second supervisor Prof. Dr. Volker Coors from HFT Stuttgart University of Applied Sciences. Both of my supervisors always supported me during this PhD research with invaluable advice. I want to thank both of them for their continuous efforts to guide me through this process and I am grateful for their patience and positive attitude, which provided a very fruitful working environment.

I would also like to thank my colleagues in the projects MoNa3D, VEPs and CityNet for their support. I always enjoyed discussions, meetings, coffee breaks and all the other occasions for a little chat. I learned a lot of things in these very interesting projects, not only in terms of my research but also for life.

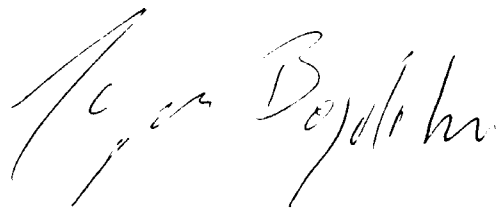
I would also like to thank all the colleagues at HFT Stuttgart for their support and the never ending interest in my work. Very special thanks to the staff at the School of the Built Environment at Salford University, who helped me with all administrative questions and who were very supportive throughout the years.

Finally I would like to thank my family for their encouragement and support. And I am especially grateful to have my girlfriend Michaela by my side, who, as always, believed in me.

# Declaration

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This work has not previously been presented in identical or similar form to any other UK or foreign examination board.

March 14, 2012

A handwritten signature in black ink, appearing to read "Ayman Baydoun". The signature is written in a cursive style with a large initial 'A'.

# Glossary

<b>ADE</b>	Application Domain Extension
<b>API</b>	Application Programming Interface
<b>AR</b>	Action Research
<b>Bbox</b>	Bounding Box
<b>BRep</b>	Boundary Representation
<b>BSPTree</b>	Binary Space Partitioning Tree
<b>CAD</b>	Computer Aided Design
<b>CAT3D</b>	CityModel Administration Toolkit 3D
<b>CityGML</b>	City Geography Markup Language
<b>dpi</b>	Dots per inch
<b>DSM</b>	Digital Surface Model
<b>DTM</b>	Digital Terrain Model
<b>EDA</b>	Exploratory Data Analysis
<b>FBO</b>	Frame Buffer Object
<b>GIS</b>	Geo-Information System
<b>GLSL</b>	OpenGL Shading Language
<b>GML</b>	Geography Markup Language
<b>GPU</b>	Graphics Processing Unit
<b>HCI</b>	Human Computer Interaction
<b>JMF</b>	Java Media Framework
<b>KML</b>	Keyhole Markup Language
<b>L-System</b>	Lindenmayer-System
<b>LIDAR</b>	Light Detection And Ranging
<b>LoD</b>	Level of Detail
<b>LoR</b>	Level of Realism
<b>lpi</b>	lines per inch
<b>MoNa3D</b>	Mobile Pedestrian Navigation in 3D (project)
<b>NPR</b>	Non-Photorealistic Rendering
<b>OGC</b>	Open Geospatial Consortium
<b>OpenLS</b>	OpenGIS Location Service
<b>PDA</b>	Personal Digital Assistant

## Glossary

---

<b>PV</b>	Photovoltaic
<b>SDI</b>	Spatial Data Infrastructure
<b>SWT</b>	Standard Widget Toolkit
<b>TBO</b>	Texture Buffer Object
<b>TIN</b>	Triangulated Irregular Network
<b>VEPs</b>	Virtual Environmental Planning System (Project)
<b>ViSC</b>	Visualization in Scientific Computing
<b>VR</b>	Virtual Reality
<b>VRML</b>	Virtual Reality Modelling Language
<b>W3DS</b>	Web 3D Service
<b>WFS</b>	Web Feature Service

# Abstract

Three-dimensional digital representations of cities are widely used today, from urban planning to navigation systems, emergency response and to energy and flood simulations. Many of these scenarios can be served by one multipurpose 3D city model that has the semantic and attribute information depth that is required (besides the geometrical detail). These multipurpose models do not only represent the geometrical properties and textures and materials, which would be sufficient for pure visualization of the urban space, they also model semantic entities like walls, roofs, ground, etc. And all these parts, as well as the buildings, as specific, identifiable entities, can be linked to additional information and data sets from other sources.

However, although these models have the required information-richness and can be used beyond pure visualization, one part of these models is still treated the same way as for pure visualization models: the textures. Textures in most of today's city models are still a tool to enhance the photo-realistic appearance. The primary task of the textures is still to add the 'naturalistic' elements that are not modelled in geometry. These elements are mainly located in the façades, namely windows, doors, signs, fire escapes and many more.

The presented work investigates how textures can be used for information visualization, which is more useful for the aforementioned multipurpose city models. A new texture concept is presented that is based on flexible content, which is managed in layers. In this way it is possible to adapt the appearance of buildings (especially façades) to the actual scenario. The concept also allows the integration of additional information into the façade, enhancing the 3D city model. In this way it is possible to generate scenario specific façade textures integrating the relevant information into the texture content.



# Chapter 1

## Introduction

### 1.1 Research Problem

In recent years many new applications for digital three-dimensional urban models have appeared. Many cities have a digital counterpart nowadays either modelled by private companies or by municipalities themselves. Very often these 3D city models were built for urban planning purposes and have the task to represent the urban space in a (photo-) realistic way to assess newly planned buildings in the existing spatial context. These models are more or less built from a computer graphics perspective, which intends to build an indistinguishable digital model of the real world.

However, over time more and more focus has been laid on 3D urban data and 3D visualization in the GIS (Geo-Information System) field as well. In this world a 3D city model is not only a graphical representation (a 'collection of triangles and textures' that represents the form of urban space) it also carries semantic information. In GIS features are defined, which can be identified and linked to other data. This 'GIS perspective' on 3D urban models created a further type of city models with different intensions. Photo-realistic representation is still a scenario, which these models want to serve, but in addition they also address issues of (visual) analysis, information modelling and information creation as well as other similarly related aims. These additional aims beyond realistic visualization turn these models into 3D geo-data models. This becomes very obvious when looking at the CityGML standard defined by the Open Geospatial Consortium (OGC), which defines a semantic data structure for 3D urban models. This

## 1. INTRODUCTION

---

structure includes much more information than necessary for pure visualization purposes. Hence, the intension of the model is to serve more than this single scenario. This intension opened the way into a new research field on its own, which includes aspects of various related disciplines, like computer graphics, GIS, geovisualization, etc. The city models that are under investigation in this field are multipurpose data sets and can be used for many different applications ranging from urban planning, disaster and emergency response to navigation systems and simulation tools (e.g. energy consumption or photovoltaic potential).

However, looking at these multipurpose 3D city models in terms of textures not much has changed in comparison to models for photo-realistic visualization. For the graphical models textures were and are still used in order to enhance the photo-realistic appearance of the model. The textures do normally apply a certain material to a surface or they add elements visually, which are not modelled in geometry. Especially façade textures have the purpose to add details that are not present in 3D city models geometrically. Windows, doors, signs, ledges and other façade elements are normally not modelled in geometry and are visually added by applying a photo-texture. And this is also true for today's multipurpose models. As the geometrical acquisition and the modelling of façade details is too expensive and time consuming textures are still used to add these elements in order to achieve a photo-realistic coherence with the real world, although photo-realistic visualization is only one purpose of these models among many others.

This is why this work is going to investigate how textures can be used in additional ways to realistic visualization. As the purpose of 3D city models is not only the realistic representation of urban space other illustration types need to be supported as well. Like in a thematic map, not representing spatial information in a photo-realistic way, urban objects might represent thematic information on their surface. Therefore this work investigates how the texture content model needs to be designed to better serve requirements of thematic and non-photorealistic visualization of 3D city models.

### 1.2 Contribution to Knowledge

As already mentioned in the previous section there are city models, which are created as multipurpose semantic information spaces rather than graphical visualization models. This difference in purpose also demands other ways of visualization and illustration of multipurpose models. They are not only a 'perfect visual copy' of the real world that can be freely navigated and observed from different perspectives, e.g. for urban planning purposes. Models are of an information space nowadays. They consist of identifiable objects, which can be linked to an almost unlimited number of external data sets, which could be visualized in specific ways in order to analyse urban phenomena. Models can also be used to for simulations or can be integrated into applications like navigation support systems. In many applications city models act as an integral part of the problem solving process and not only as visual background information. It is quite obvious that the intension of 3D city models, when they are modelled in ways like CityGML defines, has grown from pure photo-realistic representation to multiple purposes in different scenarios. This requires a different visual appearance for each specific use-case.

However, this work addresses an element of 3D city models, which has not adapted to the aforementioned changes. Textures are mainly used to apply the real-world (façade-) image to the geometry of the model in order to enhance its realistic appearance. Textures are used to hold image information, which represents one fixed representation of the façade/building at a given point in time (when the picture was taken). For today's 3D city models a texture is the 'container' for an image, which in most cases shows the real-world representation of the façade. This is still the same purpose as for purely visually oriented models. But this concept is limiting in regards to today's multipurpose city models.

Therefore this work defines a new model for (façade-) texture content. This new model helps to organize content (with focus on building façades) in a way so it can be rear-

## 1. INTRODUCTION

---

ranged for specific purposes. The content is sub-divided into zones and layers, which allow adding further information into the texture content mixing it with realistic elements if necessary. This new way of texture content definition makes it possible to generate thematic map-like textures, which integrate different information into one visual representation. In this way the presented work helps to see textures of urban objects not only as a container for static real-world images but as a channel for flexible and adjustable content in order to serve a multitude of visualization needs.

### 1.3 Aim & Objectives

In the previous section the standard use of textures was identified as limiting in terms of 3D city models and a more flexible solution is needed for today's multipurpose models. It is necessary to have a closer look at the concept of texture content, especially in regards to emerging hardware capabilities and new possibilities to directly influence the rendering process. Therefore the overall aim of this work can be formulated as:

**Determine how façade textures can contribute to information visualization for multipurpose, semantic 3D city models.**

This aim concentrates on the texture content as such and how it can be changed into a flexible and adaptable channel for various types of information and how different city model representations can be supported. In addition this work also looks at issues of 3D spatial data infrastructures and the possibility to integrate a new texture concept and its capabilities into this context. This is a necessary step as multipurpose 3D city models are often part of this kind of environment and are often integrated into SDI (Spatial Data Infrastructure) processes and workflows. Furthermore this work also evaluates the capabilities of the new concept in a case-study in order to find out how well the new content model supports 3D city model use-cases.

## 1. INTRODUCTION

---

**Objective 1: Find a way to describe the content of (façade-) textures that allows the flexible representation and combination of different content as well as the integration of additional (non-photorealistic) information.**

This objective describes the need to flexibilize the texture content. Normally a photo image is used as the façade texture, which consists of one fixed representation of the façade at a specific point in time (when the photo was taken). For multipurpose models there needs to be a multipurpose visual representation as well. Therefore it is necessary to define the texture content according to specific use-cases and scenario dependent requirements. Therefore a texture content concept needs to be found that can add specific content and can exclude other content if the scenario requires this. Another aspect of this flexibility is that scenarios might require additional content to be added besides the photo-realistic elements of the façade. As the multipurpose models consist of identifiable objects that are often linked to significant amount of additional attribute data, the need for visualizing these attributes is very immanent in specific use-cases.

**Objective 2: Define concepts in order to adjust the ratio of realism as well as the ratio between realistic elements and the integrated additional information**

As photo-realism is not the appropriate form of visualization for all use-cases in regards to multipurpose models it should be possible to adjust the ratio of realistic content in textures. Nowadays the adjustability of realism for most models consists of the options 'with photo-textures' or 'false coloured' (coloured icons, see chapter 2.3.3.3). A more fine-grained adjustability should be achieved by a new texture model, providing the possibility to accentuate relevant information. For example, when attribute information is visualized the number of realistic façade elements should be able to be reduced in order to make the other information more prominent.

## 1. INTRODUCTION

---

**Objective 3: Implement a suitable prototype in order to test the flexibility of (façade-) texture content and its appearance in regards to real-time contextual events.**

In regards to flexibility and the ability to use the new texture concept in analysis scenarios or for use-cases where context dependent changes of the model's appearance are required, it would be necessary that the texture content can change in real time. Real-time in this context is defined by an acceptable response time from the point when the new appearance was requested to the point when the change in the texture content is finished. When the response time can be held relatively low this would allow interactive changes of illustrations. In combination with the first two objectives formulated above this would mean that switching between different façade representations and different sets of information/data can be realized for the user in a similar way to a GIS. This would allow using the texture content model for knowledge retrieval and analysis of urban space and support the multipurpose nature of 3D city models in this respect.

**Objective 4: Investigate how the new texture model can be integrated into 3D-SDIs and how services can address the absence of certain client capabilities.**

As multipurpose 3D city models are used in various different scenarios they are normally managed in a special way. Very often they are stored in spatial databases and managed by a dedicated system, which provides (web-) interfaces for clients to retrieve the suitable model for a specific purpose. Furthermore these dedicated management systems are often integrated into SDIs where the 3D model can be merged with other data on request and be integrated into scenario related workflows. Because multipurpose 3D city models are often used in a SDI environment it needs to be investigated how the new texture concept can be integrated into this environment. Aspects on client/server capabilities and related distribution of workload need to be addressed. The texture content model should be able to be used in a flexible way and be able to integrate into different stages of workflows and processes common in (3D-) SDIs.

## 1. INTRODUCTION

---

**Objective 5: Evaluate the capabilities of the new texture content model in a specific use-case.**

The newly developed model for describing (façade-) texture content is evaluated for a specific use-case. It needs to be found out how the new capabilities and the flexibility of the concept can support certain scenarios in which 3D city models are used. The work presents a case-study that evaluates how flexible texture content can be used in the case-study scenario. By generalising the findings of the case-study, this work tries to estimate the benefits for the whole field of digital 3D urban models.

### 1.4 Structure of the Thesis

The remainder of this thesis is organized in the following way (see figure 1.1).

**Chapter 2** shows the wider context of the research and provides a selection of relevant work. The chapter describes the background on how 3D city models are produced and in which scenarios they are used. Different types of models and variations in their structure are also presented based on the work of Stadler and Kolbe (2007). From this description of the background the chapter develops the motivation for this research and the rationale for investigating a new texture content model for 3D city models.

**Chapter 3** presents the methodology used for the presented work. The chapter describes the process of constructive research and in which way it aims to generate new knowledge. In a second part this chapter shows which research methods are actually used in this work to perform the different stages of constructive research, namely literature review, prototyping and case-study.

**Chapter 4** describes the new model for (façade-) texture content that is going to be used to arrange information appropriately for flexible visualization of city models. The chapter develops the new texture content model based on basic elements (image, texture, etc.) and introduces new elements (zones, layers, etc.) and concepts (Level of

## 1. INTRODUCTION

---

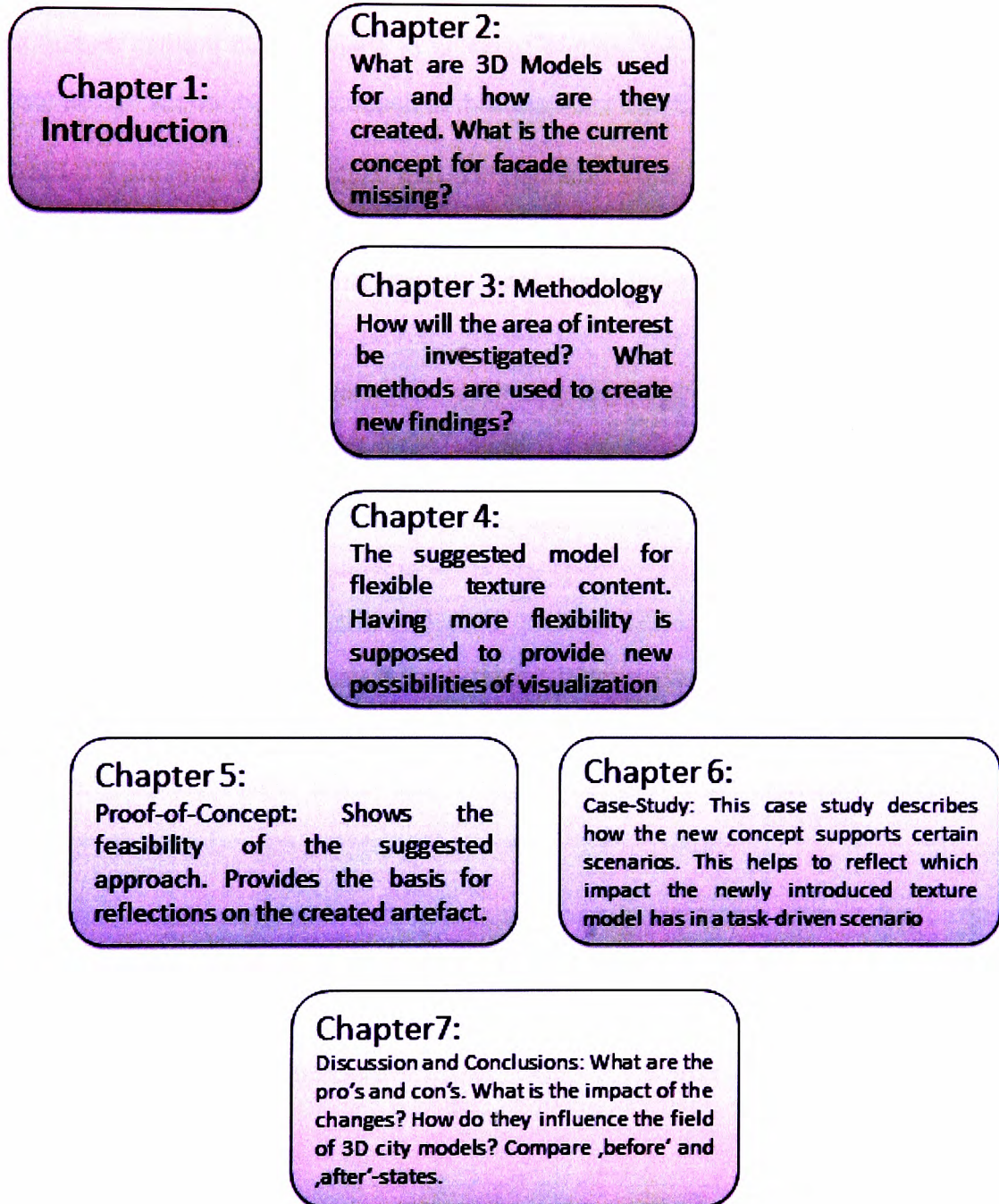


Figure 1.1: The Thesis Structure - The structure of the remaining parts of the dissertation



## 1. INTRODUCTION

---

Realism, Real-Time Layer) in order to manage the content. This chapter also discusses issues about how to integrate the new content approach into the environment in which 3D city models often exist. This integration into (3D-) SDIs is a very important part that the texture content model needs to achieve. Hence, the use of the texture model in client-server scenarios is discussed in this chapter as well.

**Chapter 5** presents the prototype that is implemented in order to prove the feasibility of the new texture content model. The second aspect that the prototype should prove is the ability to change the texture content in real-time according to user input or changes triggered by the system/application during rendering. This issue is important when 3D models are used in scenarios where the model appearance needs to change during the visualization and the model is not loaded in a specific representation that does not change while the model is in use.

**Chapter 6** presents a case-study in which the new texture content model is assessed and it is expected to be of mutual benefit for the task-solving process. Some aspects of the new content model are compared to the traditional texturing approach and benefits of the new approach are discussed. The chapter intends to show the applicability of the texture content model and to outline extended visualization capabilities, which are regarded as contributions to existing knowledge and concepts.

**Chapter 7** provides conclusions and an outlook on future work. This chapter revisits the objectives formulated in chapter 1.3 and sums up the results in regards to these objectives. The second part of the chapter formulates starting points for future research in relation to specific aspects that were discussed in this work.

# Chapter 2

## Background & Motivation

This chapter is going to provide an overview of the current state-of-the-art and the context in which this research is conducted. It starts with relevant work in the field of 3D city models in order to provide an overview of the wider research field and it also includes a section discussing different data acquisition methods. The variety of acquisition and modelling approaches reflects the different types of models that exist as well as the aspect that models are captured and built differently for different purposes. A further part of the context summary is going to present the definition for six types of models and their differences, how they are managed and integrated into spatial data infrastructures.

The state-of-the-art and relevant work in this chapter provides the current situation of 3D city modelling and visualization and shows the existing situation in this field. Besides demonstrating and clarifying the context of the research it also describes the current situation to which the results of the work need to be compared (see chapter 3). In the sphere of differences between current state-of-the-art and the newly developed model for texture content (presented in chapter 4) new findings are expected to be found and to be evaluated.

The second aim of this chapter is to show the motivation and rationale for the conducted research. By looking at the current situation it identifies drawbacks and limitations of the state-of-the-art and specifies the need for the development of a novel artefact/construction in order to change and improve the current situation (chapter 2.3).

## 2. BACKGROUND & MOTIVATION

---

Hence, the chapter acts as the context of the work and provides a basis for validating the result of the work on the one hand. On the other hand it uses the current situation, more precisely the lack in current knowledge, to explain the rationale and motivation of the work.

### 2.1 Relevant Work

In this part examples of relevant work are presented. The section provides an overview on the state-of-the-art in 3D city modelling as well as a selection of use cases, in which these models are used. Besides the different forms of use this part of the work also includes state-of-the-art data acquisition methods and a section on procedural model generation based on grammars and specialized algorithms. Knowing the current state of 3D city models and their purpose provides the context in which this research is undertaken. Therefore it describes the 'status quo' of the research field to which the results of this work is going to be compared to. Hence, a profound knowledge about the relevant work and the current concept of 3D city models is essential. The use-cases and ways of data capture presented in this section reflect this existing concept.

#### 2.1.1 Use of 3D city models

Three dimensional urban models play a growing role not only in terms of visualization purposes but also as well regarded data sets in GIS and related disciplines. This *'is illustrated by some state-of-the-art cases, worldwide, suggesting that rapid changes are taking place in the ways such visualisations are being developed. We note developments in remotely sensed survey and in the development of 3D models integral to spatial databases as reflected in GIS'* (Shiode, 2001). They are useful in many scenarios and can provide an extra benefit when integrated into other applications. Shiode (2001) identified *'four different categories of use, (1) planning and design, (2) infrastruc-*

## 2. BACKGROUND & MOTIVATION

---

*ture and facility services, (3) commercial sector and marketing, and (4) promotion and learning of information on cities*. In specific scenarios it appears beneficial to have a visual/photo-realistic representation of the city (planning and design) in order to make design decisions or to learn about the geographical layout of the city ((4) promotion and learning of information on cities). *'Planning and detailed design reviews as well as problems of site location, community planning and public participation all require and are informed by 3D visualisation. The focus is upon aesthetic considerations of landscapes as well as daylight and line-of-sight. Visual representation of environmental impact is also widely supported by 3D models'* (Shiode, 2001). In other scenarios like infrastructure and facility services it might be necessary to work with more detailed models with a set of semantic information and attributes attached to the semantic entities. This allows performing analyses of the urban objects, etc.

When 3D models are integrated into other applications there are two ways how the models can support the specific scenario and the application in which it is integrated. For some scenarios the 3D model provides the context for other spatial information. In recent car navigation systems, for example, the 3D model acts as the background for the visualized route and the direction instructions of the system (fig. 2.1). For some scenarios the 3D model provides the context for other spatial information. In recent car navigation systems, for example, the 3D model acts as the background for the visualized route and the direction instructions of the system (fig. 2.1).

The process of calculating the route and providing guidance instructions as such does not include any 3D data or is linked to the 3D model. Placing 3D building models onto the tilted map with the route visualization should provide the context for the user to support the self-localization and navigation process. However, there are also efforts to integrate 3D information/the 3D model into this process directly. The project MoNa3D (Mobile Pedestrian Navigation in 3D) (Coors and Zipf, 2007) tries to integrate 3D city models into the pedestrian navigation process by including landmarks, building objects 'knowing' about their function as landmark, into route instructions and visualize them

## 2. BACKGROUND & MOTIVATION



**Figure 2.1: Example for 3D building models in a Nav-System** - An example for 3D buildings in a car navigation system providing spatial context (from <http://www.navigon.com/>, copyright Navigon AG)

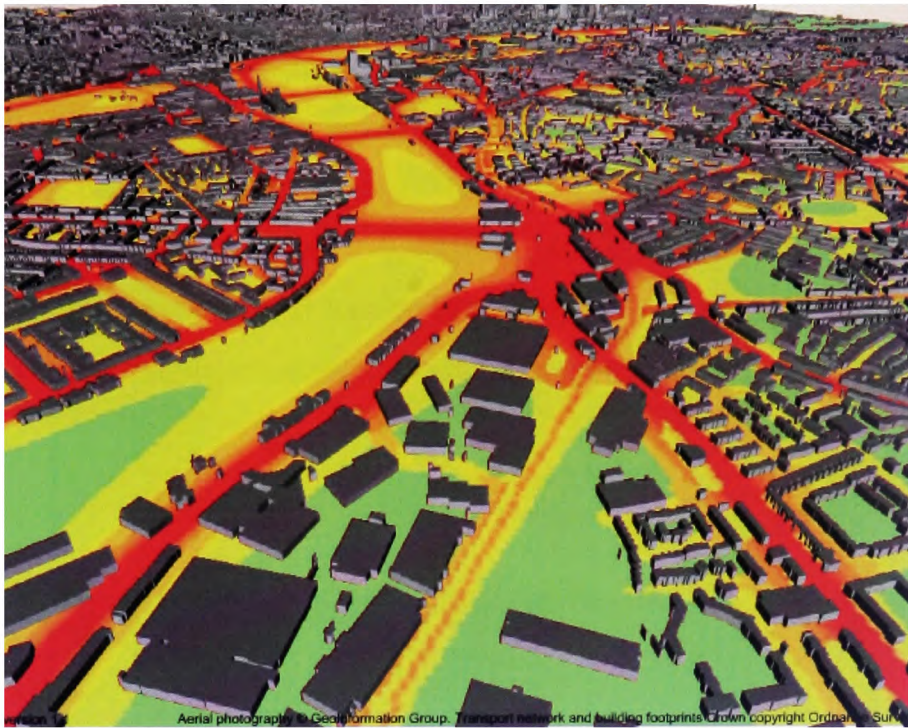
accordingly. Here the 3D model is tied stronger into the application and the solution of the actual task. The landmarks for the specific route are automatically identified by an algorithm that scans the 3D model and they are integrated into navigation instructions (see chapter 6).

Another example is using 3D city models for simulations of environmental influences. Floods as well as air- and noise pollution need to be based on some kind of 3D data. Three-dimensional digital city models can be used to provide the geometrical input data for these simulations. Kokas (2008) provides some flooding simulation results based on his models and Czerwinski et al. (2006) provides a report on noise simulations using 3D city models as a basis. Another example for using 3D city models as input for simulations is presented in Strzalka et al. (2010). Here the modelled 3D buildings are used to extract properties like volume, height, orientation and area of walls, etc., which are then used as one of the required inputs for an energy demand estimation simulation. This simulation cannot only be performed for one single building but for a whole borough or a whole city. In addition digital city models can also be used to

## 2. BACKGROUND & MOTIVATION

---

visualize simulations results and provide the spatial context for the produced output. Other examples are the Air Pollution Map for London ([www.londonair.org.uk](http://www.londonair.org.uk), 2011) (fig. 2.2) and Kokas (2008) provides some examples on flood simulation visualizations in his thesis for Heerbrug, Switzerland and London.



**Figure 2.2: Visualization of air pollution** - Air pollution overlaid onto the 3D model (LoD1) of London (from <http://www.londonair.org.uk/london/asp/virtualmaps.asp>)

In the field of flood simulations Schulte and Coors (2008) present an Application Domain Extension (ADE) in order to integrate hydrological data into the CityGML standard. In this way the 3D city model and flood data can be exchanged together in an open data format and can also be processed together, e.g. for visualization or for rerunning the simulation and verifying it.

A further example for using 3D city models in an application context is presented in Knapp et al. (2007). The presented system for public participation in urban planning uses digital 3D models of the development area in a web-based system to communicate with citizens about existing and planned buildings. In their prototype the 3D model is not only a visualization tool but part of the user-interface. Citizens can post

## 2. BACKGROUND & MOTIVATION

---

a comment, or access the comment of others, by selecting the according building directly in the 3D model. Exploring what-if scenarios is also possible by switching certain planning proposals (modelled in 3D) on/off in the 3D scene. In this case the model is not only a way of visualizing a planning proposal and thereby communicating in a one-to-many fashion (municipality to citizens) but also allowing a discussion which is a many-to-many fashion (municipality  $\Leftrightarrow$  citizens and citizen  $\Leftrightarrow$  citizen). In literature one can find more scenarios in which 3D city models are used beyond pure visualization and to help to solve other tasks. There are examples for emergency response (Kolbe, 2008), simulation of wave propagation ((Fabritius et al., 2008), (Schmitz and Kobbelt, 2006)), simulation and training (Bildstein, 2005) (fig. 2.3), public safety (Qiu et al., 2004) and the visualization of change in urban areas over time (Weber et al., 2009).



**Figure 2.3: Visualization of simulation scenarios** - Examples for simulation and training scenarios using virtual 3D environments(from Bildstein (2005))

## 2. BACKGROUND & MOTIVATION

---

### 2.1.2 Data Acquisition

The 3D models for these and many other applications are generated in many different ways, which can include various sensors and algorithms. As cities are quite large structures and it is necessary to model a quite significant number of objects, therefore normally sensors are used that can acquire a big amount of data in reasonably short time (for a comprehensive summary see Hu et al. (2003)). Appropriate and intelligent algorithms and software tools help to manage and analyse these large data sets in order to extract the required information (urban objects). Another strategy is to produce city models procedurally (e.g. Fabritius et al. (2008)) according to a specific algorithm which can also use a set of rules that can be combined in an intelligent way. This procedural approach will be described in detail in the next section. A quite common approach to model buildings, vegetation and streets is based on aerial photogrammetry. Aerial photos can be used to gather urban structures in an (semi-) automatic way. Examples can be found in Brenner et al. (2001) giving an overview on different modelling concepts for 3D city models. In Kokas (2008), Schwalbe et al. (2005) and Brenner and Haala (1998) approaches for acquiring building data from aerial laser scans (LIDAR) are described and an approach using satellite images for automatic 3D building reconstruction can be found in Lafarge et al. (2007). A relatively new data source for detecting and modelling the geometry and acquiring textures (façade textures) of buildings is oblique imagery (Wang et al. (2008), Lorenz and Döllner (2006)). Hamruni (2010) also describes an approach using oblique images to model 3D urban buildings and façade textures. The retrieval of façade images is possible because not only vertical images (nadir) are taken but oblique views are recorded as well. In the oblique pictures the façade structures can be better analysed and modelled.

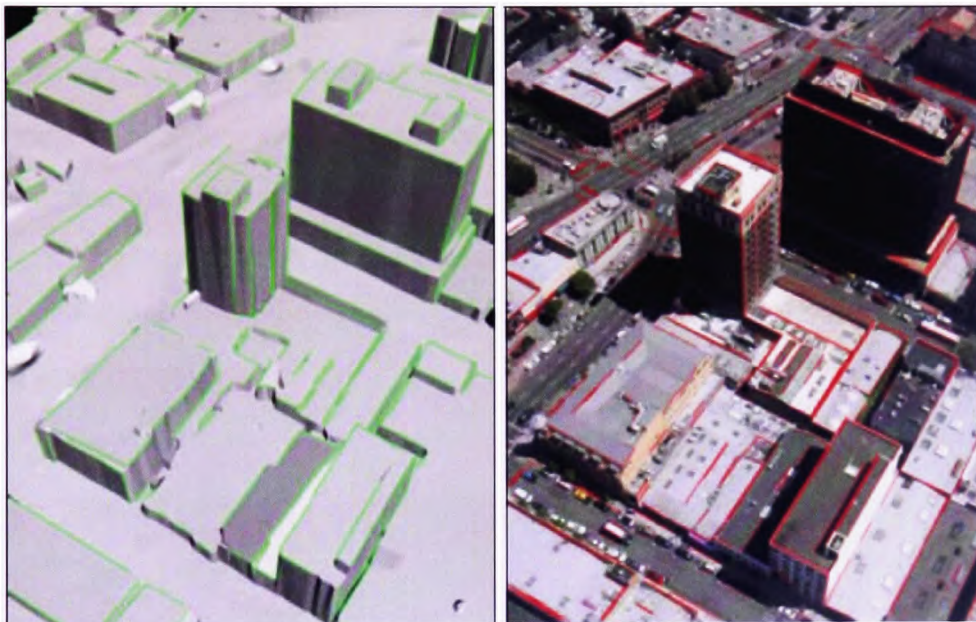
Especially the generation of façade images and textures, respectively façade structures, is investigated by many researchers. Different data capture methods are presented, e.g Laycock and Day (2006), as well as algorithms to remove objects that occlude parts of the façade (Ortin and Remondino, 2005). As well as the problem of



## 2. BACKGROUND & MOTIVATION

---

occluded areas another aspect of textures is investigated: The automatic mapping of the image to the building's geometry (Kada et al., 2005). As the process of applying textures manually to the building geometry is very time-consuming and expensive an automated solution is beneficial. Besides the concepts for terrestrial images there are also methods presented for oblique imagery (Frueh et al. (2004), Lorenz and Döllner (2006)).



**Figure 2.4: Oblique imagery for generating 3D city models** - (a) Visible 3D lines of the model in green; (b) 2D lines in the image in red.(from Frueh et al. (2004))

### 2.1.3 Procedural Modelling

As mentioned before, the modelling process for entire cities is still time consuming and rather expensive because it is still not fully automated. In scenarios where it is not necessary to produce an identical copy of a real world city (or the real world city does not exist as a reference) procedural approaches can be used to generate buildings and other urban objects according to (user-) defined rules. Coelho et al. (2007) presents an approach that is based on Lindenmayer-Systems (L-Systems). A further example for modelling façades using 'a hierarchical typed graph' is given by

## 2. BACKGROUND & MOTIVATION

---

Finkenzeller and Schmitt (2006) and described in detail in Finkenzeller (2008). Parish and Müller (2001) present a further concept that is based on extended L-Systems for creating street networks and building geometry. Wonka et al. (2003) do present concepts for procedural modelling for virtual cities based on what they call split grammars, which *'draws from the work on shape grammars pioneered by Stiny'* (Wonka et al., 2003). Müller et al. (2006) consequently combined the approaches of Wonka et al. (2003) and Parish and Müller (2001) to the CGA Shape-concept for procedural modelling of even more detailed building mass models. In their paper Müller et al. (2007) the concept is also extended on the modelling of façades. This approach allows extracting façade elements and rules for the façade structure. The 'style' of the extracted façade grammar can then be applied to arbitrary building geometries (fig. 2.5). A more general procedural approach for texture synthesis for wall materials can be found in Legakis et al. (2001).



**Figure 2.5: Façade grammars** - Façade that is subdivided into smaller elements. A grammar is derived from this description(from Müller et al. (2007))

## 2. BACKGROUND & MOTIVATION

---

Lipp et al. (2008) developed a concept for users to vary the façade rules giving designers more influence on the final appearance of the façade style. A comparable similar approach for façade texture construction based on real world images is presented in Ricard et al. (2008). In contrast to Müller et al. (2007), this approach '*uses a merging of various histograms resulting from image analysis algorithms (hough, edge and background detector) to detect the main features of a façade*' (Ricard et al., 2008) inside the real world image. Façade grammars are also interesting for the concept presented in this work. For generating façade grammars features and elements of the façade need to be detected and their distribution on the wall needs to be identified. This information could also be used as the input for the texture model described in chapter 4.

Other authors also investigate façades and try to detect and extract features inside of façade images. Lee and Nevatia (2004) presents a concept for automatic window detection using '*a profile projection method, which exploits the regularity of the vertical and horizontal window placement*'. In Wang et al. (2002) another approach for façade structure extraction is presented. Ripperda and Brenner (2007) present a concept for façade reconstruction that is '*based on a formal grammar to derive a structural façade description in the form of a derivation tree and uses a stochastic process based on reversible jump Markov Chain Monte Carlo (rjMCMC)*'. Becker et al. (2008) describes a method for façade reconstruction based on terrestrial laser scan point clouds. Missing windows that are not detected due to low data acquisition quality are inserted into the façade using a formal grammar that is extracted from formerly reconstructed façades, which act as a knowledge base.

### 2.2 Background

This part of the chapter is taking a closer look at the actual structure of 3D city models and at definitions of various manifestations of actual models. Stadler and Kolbe (2007) provide a very good discussion on six types of 3D city model structures. The different

## 2. BACKGROUND & MOTIVATION

---

types of models can be related to different ways of data acquisition/modelling as well as different terms of use (described in the previous sections). More sophisticated models in terms of additional information and semantically rich data sets can be modelled using formats like CityGML. These models can be characterized as multi-purpose data sets as they can include much more information about urban space/urban objects than necessary for pure visualization purposes. Therefore these models are also managed in a different way in order to provide access to information included in the data models. The relevant infrastructure and concepts for model management are also presented in this section to show the (technical) environment in which these 3D city models exist.

### 2.2.1 Six Types of 3D City Models

As one can see in chapter 2.1, 3D city models can be created in very different ways for very different purposes. Therefore the structure as well as the amount of (semantic) information that is included into these models can differ quite significantly. There are different types of city models and not all of them can be used for all purposes. Stadler and Kolbe (2007) provide a profound discussion on different 3D city model manifestations and define six types of models. In their paper they discuss the two characteristics of 3D city models, the geometric and the semantic model, based on the CityGML standard (OGC, 2008a) and show the interrelation and detail in which the two parts are represented for the six model types. This thesis follows their definition of the six model types and refers to them using the same numbers (type-1 to type-6 models).

The six cases from Stadler and Kolbe (2007) are:

- Case 1: Only geometry, no semantics
- Case 2: Only semantics, but no geometry
- Case 3: Simple objects with unstructured geometry
- Case 4: Simple objects with structured geometry

## 2. BACKGROUND & MOTIVATION

---

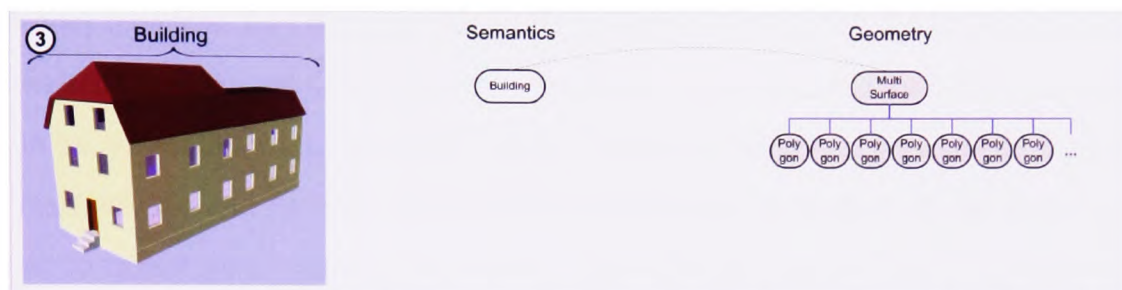
- Case 5: Complex objects with unstructured geometry
- Case 6: Complex objects with structured geometry

The distinct cases (1-6) defined by Stadler and Kolbe (2007) depend on the two aforementioned characteristics of 3D city models: geometry and semantics. The authors refer to ISO 19109 and the GML3 standard (Geography Markup Language) for the CityGML model, which describes building objects. Basically all classes in CityGML's semantic model are *'derived from the basic class feature, defined in ISO 19109 and GML3'* (Stadler and Kolbe, 2007). For the geometrical model the CityGML standard uses *'objects of the GML3's geometry model which is based on the standard ISO 19107 'Spatial Schema' (Herring, 2001), representing 3D geometry according to the well-known Boundary Representation (BRep, Foley et. al, 1995). CityGML actually uses only a subset of the GML3 geometry package'* (Stadler and Kolbe, 2007).

For example, a type-3 model (fig. 2.6) defines a simple building object in the semantic model and unstructured geometric instances to form the shape of the building. Type-1 models do not even have a semantic entity for 'urban objects'. These models exist only of unstructured geometry that models the visual appearance of urban space. Models of this type are *'based on 3D graphics formats like VRML, X3D, KML, U3D or legacy CAD formats'* (Stadler and Kolbe, 2007). However, introducing a minimum of conventions it is possible to generate type-1 models that can easily be converted into type-3 models. In VRML (Virtual Reality Modelling Language) (Web3D Consortium, 2003) for example, it would be possible to define each semantic building as a group node named after the ID of the actual building (as a link to further non-visual data). This group node would act as a container for the formerly unstructured geometry. Hence, it is possible to a certain extent to transform models from one type to another.

Besides type-6 models, which will be described later in this section, type-3 models are in focus in this work, because they provide sufficient semantic information to fulfil the requirements of a number of scenarios in which 3D city models are used. In type-3

## 2. BACKGROUND & MOTIVATION



**Figure 2.6: Type-3 Model** - Simple object with unstructured geometry (Case 3)(Stadler and Kolbe, 2007)

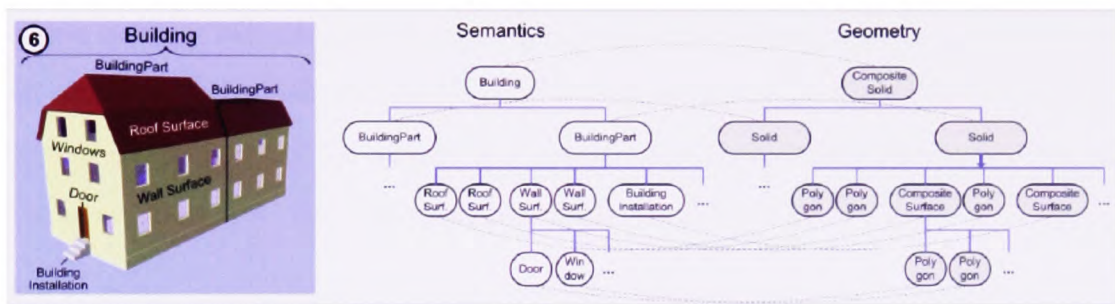
models it is possible to identify semantic objects (and their type). It is possible to identify single buildings in the data set and retrieve an identifier, which allows accessing any non-visual data that is linked to the object, e.g. building attributes like year of construction. Therefore, these models can be used in information systems where the city model acts as a freely navigable 3-dimensional interface to further data linked to the 3D representation.

For example, in the EU-project 'VEPs' (Knapp et al., 2007), which investigated the use of 3D city models in public participation in urban planning, it is possible to navigate a 3D scene of existing and planned objects. It is possible for users to get additional information about objects in the scene and to leave comments (linked to buildings) by choosing the building in the 3D scene. In this case the visual aspect of the model was important, but also the semantic entity that needs to be present in order to link data/comments or access additional information. Semantic modelling of walls, roof or building floors, etc. was not necessary because the scenario did not require them in this case.

The models that this work investigates in detail are the type-6 models (fig. 2.7). This type of model provides a very high '*spatio-semantic coherence*' (Stadler and Kolbe, 2007), models semantic and geometry in the same detail and also links the two elements of the 3D model. Walls, roofs and ground elements of the building hull, for example, are modelled in semantics and a separate geometry is associated with them. Hence, in type-6 models each semantic entity has a geometrical counterpart. Ad-

## 2. BACKGROUND & MOTIVATION

ditional attributes can be assigned to the semantic entities and an information-rich, meaningful model can be created. In contrast to type-3 models even more scenarios can be served as more information about the objects and their structure is present. Type-6 models can therefore be regarded as multipurpose models, especially in contrast to type-1 pure visualization models. The CityGML data format is a well-known format that can describe type-3 as well as type-6 city models<sup>1</sup>. Many examples can be found in literature for CityGML use cases supporting the multipurpose nature of type-6 models.



**Figure 2.7: Type-6 Model** - Complex object with fully coherent spatio-semantic structure (Case 6) (Stadler and Kolbe, 2007)

### 2.2.2 Multipurpose Model Management

Multipurpose, semantically rich models demand for better management systems, which can provide specific subsets of the model as well as scenario specific information. Having this kind of management systems at hand appears to be crucial as the models are supposed to be used in different scenarios. Therefore it is necessary to be able to query exactly the 'form' of model that is required in the specific scenario.

The previous section outlined that type-6 models have a high coherence between the semantic and the geometrical part of the model and that these models can be also

<sup>1</sup>CityGML supports cases 2 to 6. Case 1 is not directly covered, because CityGML is based on the feature model of ISO 19109 and therefore always needs to assign geometries to (at least simple) semantic objects' (Stadler and Kolbe, 2007). In this work case 3 and 6 models are in focus, which can be described by CityGML.

## 2. BACKGROUND & MOTIVATION

---

linked to various external data sets. This can be regarded as one of the major reasons for the multipurpose nature of type-6 models, they can serve various scenarios because they provide more information about the semantics and structure of urban space and they can be linked to scenario specific information on data level (not only visually superimposed).

However, as there is not only one specific use-case into which the model is permanently integrated, there are probably many different 'user'-systems, which have very specific requirements and also different hardware capabilities. Furthermore, it is extremely unlikely that arbitrary clients in various scenarios work with exact the same part of the model or with the model as a whole. Moreover, it is necessary to provide different spatial parts of the model for a specific use-case, most probably in a specific style or an adequate Level of Detail (LoD) (see chapter 2.3.4).

With all these requirements in mind (and there are certainly several more) it is obvious that this flexibility can hardly be achieved by a file-base city model. Especially when separate (file-based) models are used for separate use-cases, e.g. in specific departments of a municipality there is a danger of inconsistency, storage of the same information in several places (inefficient) and possibly the establishment of 'information islands' that produce considerable problems when need to be merged with 'outside' data or models.

A possible solution for managing the models more efficiently and in a more flexible way can be found in the GIS world. Geographic data is very often managed in dedicated GI systems that very often make use of spatially enabled database systems. A similar approach can be observed in the case of multipurpose 3D city models, which show the characteristics of 3D geo-data. These models are also managed in spatial database systems and administered by dedicated 3D model management systems (e.g. J. Haist (2005)). The next section is going to describe these systems in more detail.



## 2. BACKGROUND & MOTIVATION

---

### 2.2.3 Management Systems for Urban Models

As outlined in the previous section it is very complicated to serve multipurpose models by a file-based approach and therefore dedicated systems are in use to provide models in diverse scenarios. This section looks at the 3D management systems in more detail and on the requirements they have to fulfil. It will also look at the CAT3D framework (CityModel Administration Toolkit) that was developed by the author of this thesis and extended in term of this work. By using this framework as an example possible implementation options for service processes are discussed to show how the flexible output of these systems can be achieved.

The management systems that this work refers to need to serve different outputs based on one centralized model (not based on separate models for each scenario) in:

- the right size (spatial extent as well as data size)
- the right data format
- the right type (1-6)
- with the right information density
- in the appropriate visual representation

The **size** of the model depends on two aspects: 1) for specific scenarios it is not necessary to use the whole 3D model. The application scenario only focuses on a specific quarter or the area along a certain route. Here size refers to the spatial extent and is scenario-driven. In some cases the streaming of small portions of the model is also part of the scenario. 2) Specific client hardware can only handle small amounts of data (e.g. a mobile device) because of their hardware configuration or limits in the transmission of data (wireless networks, mobile phone networks). In this case the model should be small in terms of data size. This refers to the extent of the model but also to information density or the LoD. This aspect is influenced by hardware requirements, but is also influenced by the use-case in which the model is used.

## 2. BACKGROUND & MOTIVATION

---

The **correct data format** is a quite obvious requirement as different client platforms usually work on different data formats. This format is also linked to the way the model is used on client side. For visualization data formats like VRML (Web3D Consortium, 2003) are used. When transmitting data to a remote information system (e.g. a GIS or another application server in a SDI) the full range of information is needed and a format like CityGML would be preferred.

Aspects like **information density** and **suitable visualization** are more scenario-driven as certain use-cases do not require the full information that is present in the model. The system itself and the implemented interfaces should allow filtering content accordingly. But not only the capabilities of the middle-tier and the interfaces are important in this aspect, the data model as such needs to be in a form that allows combining information streams and not only represent pre-defined, fixed representation of one type of information. In terms of visualization not only the standard representation of photo-realism might be required by the client, but a combination of information sets to solve specific tasks. In the field of geovisualization this is regarded as very important for knowledge construction and information retrieval (Dykes et al., 2005). Interfaces and the management system should support these capabilities in order to allow multiple visual representations of the 3D city models.

In order to fulfil these requirements and flexibly provide output for scenario-driven mash-ups of SDI services the 3D city model management system needs to be implemented in a flexible way and to handle most diverse requests and serve various information requirements. Systems like the City Server (J. Haist, 2005) provide this kind of flexibility and the required functionality. Another system that is developed by the author of this thesis can also act as a centralized 3D model repository. The CAT3D framework provides functionality for managing 3D city models and serves them to other systems through the OGC Web3D Service interface (W3DS). The framework acts as a basis for developing the solutions presented in this work and to assess SDI related questions. The flexibility of the framework is ensured by a modularized architecture

## 2. BACKGROUND & MOTIVATION

---

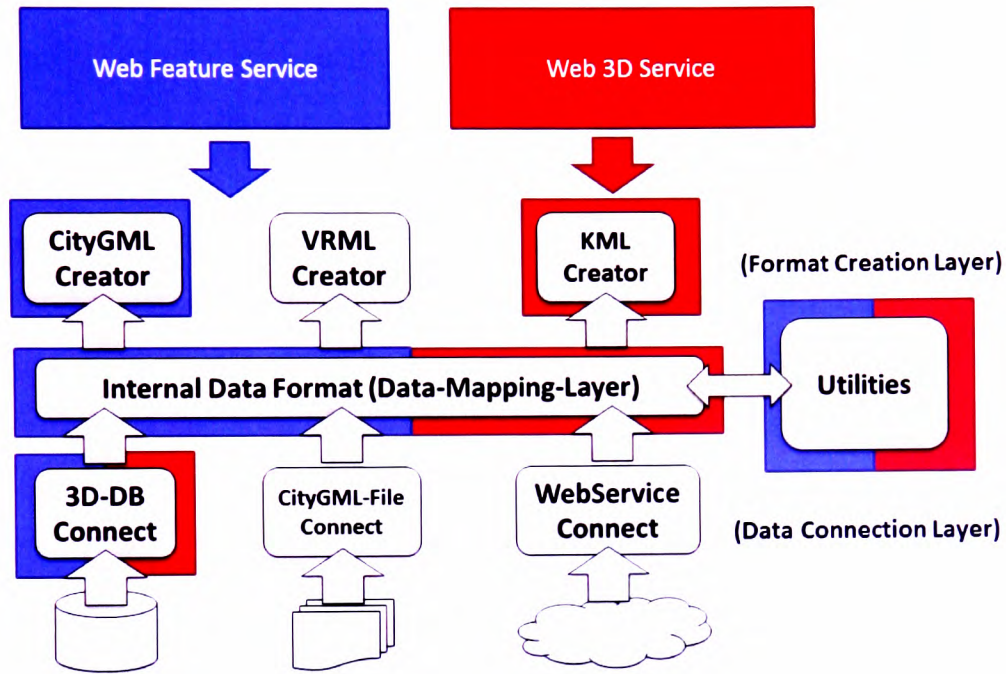
that groups modules in several logical layers. The general architecture and description of CAT3D can be found in fig. 2.8 (grey boxes). The framework is built of three general layers: Data Connection, Data Mapping and Format Creation. The Data Connection layer symbolizes modules that establish a connection to a data source and translate the external information into the internal data representation (Data Mapping Layer). Here the information can be integrated with data from other sources or transformed (e.g. coordinate reference system). The translation from the internal representation into a specific output format is done by one of the modules in the Format Creation Layer. Here the required information is read from the internal representation and translated into an output format requested by the client application through one of the service interfaces.

It can also be seen in figure 2.8 that it is not only possible to support one specific data service interface. It is possible to implement a W3DS (red), for example, that provides 3D scene graphs for rendering (e.g. type-1 or type-3 models). However, it is also possible to implement a Web Feature Service (blue) interface (WFS) that provides CityGML output, which includes the full type-6 model with all of its semantics and attributes in addition to the geometry. The two interfaces on top of the 3D data management system ensure the aforementioned requirements of delivering the right 'model type' for a specific scenario or workflow through the appropriate interface.

Feeding the data that is requested through those interfaces involves several components of the framework to be linked together to a process chain. One process chain is needed for the WFS workflow (blue elements) and one for the W3DS workflow (red elements). As modules of the framework are reusable the specific functionality does not need to be re-implemented for each new process chain of other service interfaces.

The flexible management systems provide an appropriate repository in which ideally type-6 models should be stored in order to have the full range of information available. Upon request of specific clients it is still possible to provide type-1 or type-3 models depending on the intended use and the capabilities of the client system. As

## 2. BACKGROUND & MOTIVATION



**Figure 2.8: CAT3D as basis for OGC Services** - Service implementations using specific modules of the framework (red - W3DS; blue - WFS)

the presented type of management systems are capable of providing several types of output and serve a multitude of scenarios, they are also integrated into (3D-) spatial data infrastructures in order to support external process chains and interact with other systems in this infrastructure.

### 2.2.4 Spatial Data Infrastructures

As already discussed a file-based per use-case solution for the investigated multipurpose models appears to be inefficient and keeps the danger of inconsistencies and is also inefficient in terms of data storage, as models or parts of them have to be stored multiple times. But not only serving multiple clients/scenarios is an issue for the management of 3D city models. City models can also exist of data that is provided from different distributed sources. Especially within municipalities data for the 3D model can be acquired, maintained and distributed by separate departments, e.g. trees/vegetation, roads, terrain, buildings can come from separate departments or organizational units.

## 2. BACKGROUND & MOTIVATION

---

In order to exchange this information, link and merge it in order to build the 3D model of the city a certain infrastructure and management needs to be established. This is especially true if the required information, respectively the required model for a specific scenario is generated in (near) real-time on request and is not a pre-processed model for the specific use case. This interaction can become very complex and achieving acceptable results grounded on file-based models is almost impossible or very costly. So for real-time integration of data into a model various (sub-) systems of the SDI need to communicate with each other. This is often achieved by open standard interfaces and data formats inside the SDI.

According to Wytzisk and Sliwinski (2004)

*'a SDI can be understood as a multi-levelled, scalable, and adaptable collection of technical and human services, which are interconnected across system, organisational, and administrative boundaries via standardized interfaces. Those services enable users from different application domains to participate in value chains by gaining a seamless access to spatial information and geo-processing resources. Scalability and adaptability reflect here the evolutionary nature of a SDI where stakeholders join or leave, technical or human services are launched, substituted, or discontinued, and novel means are integrated without any interference with already existing SDI operations. It also mirrors the need to allow users to adjust their usage behaviour as well as to reconfigure the processes executed in their application domain. In this context, technical services refer to interoperable and modular software entities that provide users with an access to distributed geo-processing functionality and spatial information resources through an open and standardized interface. Geographic information (GI) services are a subset of technical services. They enable the discovery, retrieval, processing, manipulation, analysis, and visualization of spatial data from multiple sources via a communication network.'*

Several of the technical services, respectively their interfaces, Wytzisk and Sliwinski (2004) refer to are defined by open OGC standards, for the field of (3D-) geospatial

## 2. BACKGROUND & MOTIVATION

---

data and they also apply for 3D city models. The new content concept for textures needs to be able to integrate into these concepts and the relevant interfaces to be used for type-6 city models in a consequent manner.

A very important aspect of a SDI is the exchange of data and information in a standardized way. By having standardized interfaces at hand different services and systems can be linked and utilised in order to build scenario specific processes and workflows. For different processes the relevant services can be linked in different ways in a relatively uncomplicated manner, as the interfaces are well known. The OGC defines several different data standards and interfaces, which can be used to build up SDIs. Interfaces include specifications for data access (WFS, W3DS, etc.) but also access to geo-processing services like the Web Processing Service, Web Coverage Processing Service or parts of the OpenLS services<sup>1</sup> (OpenGIS Location Service). These interfaces can be used to perform specific processes for given geo-spatial data sets.

Looking at the concept of exchanging data for different purposes and different clients type-6 models can hardly be managed in a file-based multi-instance way. In order to assess the possibility to integrate the new texture content model this work looks at the scenario in which the 3D city model is managed in a centralized system that provides the urban model to various clients and other services inside the SDI. The centralized system for 3D model access is assumed to be similar to the CAT3D framework presented in the previous section. It was outlined that it is necessary to manage this type of model in a system that is flexible in terms of data management and data processing. This system facilitate the maintenance of one type-6 model (e.g. for a whole city) and provide it, through appropriate interfaces, to other systems in the SDI (other services) or to end-user applications. The centralized city model can be better maintained and there is only one 'contact point' within the SDI for accessing the required city model data. Examples for specific client-server scenarios are provided in chapter 4.8 suggesting possible ways of integrating the developed texture model.

---

<sup>1</sup>see the OGC Website for mentioned standards: [www.opengeospatial.org](http://www.opengeospatial.org)

### 2.3 Motivation

#### 2.3.1 Textures in current 3D City Models

Despite the multipurpose nature of type-6 models and the assumption that versatile visualization is needed, textures currently still appear to be regarded as containers for photo-images of the real world object. Textures are mapped to geometry in order to enhance the realistic appearance of the model (as in type-1). Textures normally add missing geometrical detail visually. In terms of façades this would mean an observer can see building floors, windows, doors, ledges, etc. although there are no geometrical entities modelled. The texture holds a pixel matrix that is filled with colour values normally taken from a photo. Of course, the content of the matrix can be replaced by loading different colour values, e.g. a thermal image. However, the loaded information (thermal or real world image) shows one fixed representation of the object and photo-textures are used for one fixed purpose: provide a maximum of realism for the 3D building. Examples for other forms of texture content are rare. As buildings are mostly modelled in LoD2, without façade details, for the majority of buildings, textures are used to add the missing elements visually. In terms of a whole city with several thousand buildings, acquisition of façades through photos is currently easier than modelling the windows and other elements geometrically, when visualization matters are concerned.

Nevertheless, there are some examples in recent literature that use textures for information visualization, hence holding other content than the photo-realistic representation and adding additional information. Buchholz (2006) presents a use-case for his multi-resolution texture atlas. Here pre-computed visibility information is used as textures for the 3D model. In this case the combination of realistic and thematic content is pre-generated and it is not clear if interactive changes of content can be achieved. Maaß (2009) describes approaches for automatic placement of textual annotations (labels) in 3D city models and also examines building surfaces as spaces where this extra

## 2. BACKGROUND & MOTIVATION

---

content can be placed. This approach can be compared to equivalent approaches in cartography for placing text in maps. Lorenz and Döllner (2010) present a concept for 'Surface properties and their application' using the programmable rendering pipeline of modern graphics hardware to combine, analyse and visualize these properties in the 3D model.

*Surface properties 'describe data attached to 3D feature surfaces; i.e., for each surface location, a data value of an arbitrary, but homogeneous, type can be stored. Surface properties are not limited to the Earth's surface but apply to any 3D feature, such as buildings or bridges. The simplest surface property is a constant value assigned to the feature's entire surface. Usually, surface properties provide location-dependent, varying values. Typically, such a property is sampled regularly and stored as a collection of 2D rasters along with one unique mapping function per surface patch (e.g., a polygon). A surface property covers the complete surface and maps a unique raster portion to each surface patch'* (Fig. 2.9) (Lorenz and Döllner, 2010).

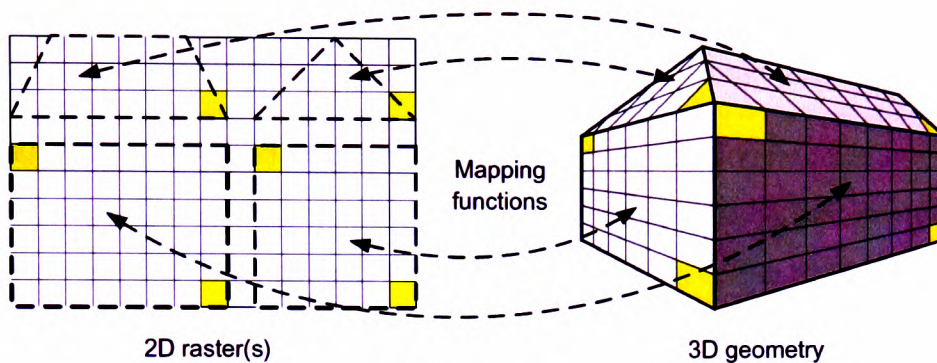
The visualization approach presented by Lorenz and Döllner (2010) supports the hypothesis of this work that 3D city models do not necessarily have to be photo-realistic. When used for analysis or information visualization purposes other visualization principles need to be applied in order to communicate the relevant aspects that are actually in focus. Lorenz and Döllner (2010), as well as this work, argue that surfaces of buildings and other objects are more than the geometrical frame for photo-textures.

Surfaces can transport much more information than the naturalistic appearance of an object and textures can act as information channels within 3D city models. In their work two examples are presented where surface properties can be used: photovoltaic potential (PV-potential) and residential quality. For distinct raster points on the surface relevant input values can be combined and the surface property can be calculated. The colour coded result is then applied to geometry. For the PV-potential analysis input information can be shadowing values, radiance levels, etc. that are distributed



## 2. BACKGROUND & MOTIVATION

---



**Figure 2.9: Schema of Surface Properties** - Schematic view of a raster-based surface property. Each surface patch is mapped to an associated unique raster area. (Lorenz and Döllner, 2010)

over the surface of the building. These values are combined and the resulting PV-potential value for each distinct position on the surface is calculated. In their prototype system this process is implemented directly as part of the rendering pipeline using shader programmes. The analysis of the input data and the combination with other information can directly be influenced by altering the shader code.

The described approaches work on solutions to place information on the surface of building objects and use them for information visualization. Especially the work of Lorenz and Döllner (2010) tries to integrate several information sets and visualize them on the object's surface. However, the texture is still regarded as a pixel matrix, which is filled by calculating a certain colour value for each texel<sup>1</sup>. A structure to organize specific content inside the texture is not exposed to the user. In digital maps, as well as GIS and CAD (Computer Aided Design) systems layers are used to organize content and provide specific control over the content. This should also be achieved for the model presented in this work. Looking at the aspects of flexibility and real-time changes of content as well as combination of different information the standard texture approach does not work well. Of course, flexibility can be achieved by providing many images containing different content combinations that are pre-processed. Switching between those pre-created representations is possible but can be regarded as ineffi-

---

<sup>1</sup>a texel in a texture is the equivalent to a pixel in an image

## 2. BACKGROUND & MOTIVATION

---

cient. Besides, if there are many possible combinations of content this would multiply the numbers of individual textures/images that need to be stored and transmitted.

As already mentioned before, when textures for 3D city building façades are regarded as links to digital images, they provide one fixed representation that is contained in the image. Looking at procedural façade concepts (e.g. Müller et al. (2007)) these are more flexible and can produce different output. However, the grammar appears to be designed to produce valid façades for arbitrary building forms. The purpose of the façade is to be valid and realistic. Once the right representation is found it would remain constant over the lifetime of the model visualization. So it is more or less a design tool.

In general textures are widely used in order to provide a photo-realistic appearance of digital models and this is quite sensible for visually oriented models, hence when realism is the intended goal of a specific visualization. However, there is some evidence that non-photorealistic representations, especially for more geo-data oriented models, in specific scenarios can be beneficial for knowledge generation and analysis. But even in the related research field of computer graphics where the photo-realistic aspect plays an important role (see next section) one can find evidence that other visualization approaches are also useful.

### 2.3.2 Photo-Realism and 3D City Models

In computer graphics traditionally there was, and still is in many applications, a focus on achieving the most naturalistic appearance of a computer generated scene in comparison to the real world. The correctness of the depicted scene is seen as the most important aspect and achieving the perfect illusion in making the computer generated rendering indistinguishable from the real world scene.

## 2. BACKGROUND & MOTIVATION

---

Schirra and Scholz (1998) describe this tendency as:

*'Most approaches in model-based computer visualizations carry a more or less implicit dedication to naturalism. Realism is seen as the ultimate goal of any effort of creating pictorial representation of reality as it is or could be, and naturalism as the single stylistic method to achieve realism.'*

Stappers et al. (2003) identified a similar tendency:

*'Most work in developing VR' (Virtual Reality) 'has been aimed at reproducing the natural world, making the stimulation the observer receives indistinguishable from the 'real' thing. We argue that this 'stimulation correspondence' view of VR has a number of drawbacks if we want to make optimal use of VR as a tool. Correspondence to the natural world is not always necessary or even desirable, is often wasteful, and tends to inhibit the development of new possibilities that VR offers.'*

In Stappers et al. (2003) the basic problem with virtual reality and realism *'is that it puts an objective world first, the user's experience, his tasks, and the information he needs for those tasks second'*. Thereby it is underpinned that a naturalistic representation is not always the required information a user needs to fulfil a specific task, a fact which is sometimes not taken into consideration when 3D city models are generated or visualized. *'But if VR is treated as an expressive medium to be used as a tool, rather than a slavish reproduction of the everyday world, interesting shortcuts and extensions emerge, including possibilities for creating more 'expressive' forms of VR'* (Stappers et al., 2003).

The 'ultimate goal' of (photo-) realism in computer graphics is accompanied by a more recent trend of non-photorealistic rendering (NPR) that is actually focusing on other aspects of real world objects and the question if naturalism might be the wrong way of visualization for each and every application of computer generated renderings or real time visualization. Schirra and Scholz (1998) provide a very sophisticated and elaborate review on the question if realism and abstraction exclude each other or if

## 2. BACKGROUND & MOTIVATION

---

they are rather mutually dependent. At this point a selection of their arguments will be examined on the background of façade textures of 3D city models.

Abstraction can be defined as the *'intentional omitting of aspects that would have to be present for a realistic representation . . . or as the pictorial representation of aspects that are actually not visible at all'* (Schirra and Scholz, 1998). This definition of abstraction is regarded as the contrast of *'realism seemingly guaranteed by a causal relation, in photography, holding between the scene represented and the image produced'*. However, a strict either-or approach seems not appropriate for them in terms of computer visualization.

Intentional omitting of (visual) aspects can be underpinned by the argument that in some cases the focus of a task is not merely on the visual appearance of objects. Objects generally have much more properties than the ones that can be observed visually. Non-visual aspects like function, meaning, semantic relevance, etc. cannot be perceived visually in a naturalistic representation. On the other hand, a completely abstracted representation can also be counter-productive, when the naturalistic aspect is omitted completely the anchor point for the non-visual information is not given and the reference to an actual spatial object is not present anymore (compare Schirra and Scholz (1998)).

The definition by Schirra and Scholz (1998) of a 'Purposeful Picture' shows that taking the scenario into account in which a rendering is going to be used is essential for the question how much realism and abstraction needs to be used in order to support the task that has to be solved. In that way the term 'realism' does not define one definite state of representation, but a property that is dependent on the circumstances in which a rendered image is used. And this is also true for urban models. For 3D city models possible application scenarios are numerous.

As described in the previous section multipurpose virtual cities can be used in many different fields with a variety of different requirements for the specific task they are

## 2. BACKGROUND & MOTIVATION

---

going to support. Taking this into account the visual representation of the model in use can definitely not be reduced to a naturalistic representation. Furthermore, the depiction in an either naturalistic or 'completely' abstracted way seems to be restricting as well. Some scenarios tend to require a more fine-grained adjustability of the degree of realism/abstraction in terms of the visualization of urban space. Ferwerda (2003) described functional realism as one of three types of realism that can be defined for computer generated visualizations. For him functional realism is defined by '*knowledge about the meaningful properties of the objects in a scene*' is provided, '*such as their shapes, sizes, positions, motions, and materials*', which '*allows an observer to make reliable judgments and perform useful visual tasks.*' This definition of realism supports the concept of the previously mentioned purposeful pictures and suggests a similar point of view on the rendering of 3D city models. Especially virtual cities with their characteristics of an urban information space and their multipurpose nature should be rendered in a task driven way. Taking this requirement into account the use of static photo-images in order to apply the façade material to the model seems to be non-optimal for these kinds of models. The texturing of prominent elements of these models should be more flexible and should provide a set of capabilities in order to enable the user to adjust the representation of the model according to its purpose of use. And as well as in computer graphics a black-or-white approach of naturalistic or abstracted representation would not really solve the problem completely. An adjustable level of realism/abstraction, where realistic and abstracted elements can be integrated into the textures needs to be provided, so that numerous applications can be served in terms of textures for urban objects.

Looking at the concept of purposeful pictures for computer graphics there is a similar philosophy that could be named purposeful 3D models. This type of models should serve a specific task and therefore provide a meaningful set of information. This information must then be visualized in an appropriate way. Appropriate here means meaningful for the scenario in which the model is used. As data sets of 3D city models

## **2. BACKGROUND & MOTIVATION**

---

can include a vast amount of information it is certainly necessary to filter the required information for the given use-case and find the best way to visualize this information in order to generate the purposeful 3D model. The 'best way' for visualizing the model lies in the appropriate ratio between realism and abstraction, where abstraction also means integrating non-visual information into the depiction.

### **2.3.3 3D City Models & Map Illustration**

In the previous section it was outlined that a purposeful representation is not always achieved by photo-realistic modelling of the real world. More specifically looking at 3D city models in relation to useful visualization approaches it is certainly beneficial to consider geovisualization and map illustration to find appropriate solutions. In terms of geovisualization the goal is to present the information that type-6 models provide in a meaningful way that is easy to understand. Additional information linked to these models can come from various sources, examples are: the use of buildings (or building parts), the year of construction, energy consumption of different apartments, occupancy, insulation level of windows, etc. The number of data sets that can be linked to building entities or their semantic sub-parts is almost unlimited.

In terms of 3D city models and geo-spatial data in general the task of meaningful representation is highly complex and involves several research challenges. The mere size of geographical data sets can be significant and data usually comes from different providers or sources. In most cases the form of the data is not homogeneously modelled and in some cases the reference system also differs and the data needs to be transformed before it can be visualized. Hence, the appropriate combined visualization of heterogeneous data sets is a challenge in itself, if visual analysis of interrelations and links needs to be conducted.

## 2. BACKGROUND & MOTIVATION

---

In order to achieve this goal geovisualization integrates ' *approaches from visualization in scientific computing (ViSC), cartography, image analysis, information visualization, exploratory data analysis (EDA) as well as GIScience*' (Dykes et al., 2005).

Especially influences and knowledge about geographic visualization comes from the field of cartography. MacEachren and Kraak (1997) describe the role of cartography in this way:

*'Cartography has much to offer the scientific community through its long history of design and production of visual representation of the earth, its knowledge of geographical (and cartographical) information systems, and its experience with linking digital and visual geographic representation.'*

However, they also recognized that cartography, on the other side, can also benefit from developments in computer science, interface design (HCI), three-dimensional computer modelling and related methods and technologies can also enhance and support cartographic visualization. This links back to the field of 3D city models. The advancements of computer technology generate a whole new challenge for cartography and geovisualization, which is actually centred on purposeful systems for (spatial) knowledge generation.

Dykes et al. (2010) identified cities as a space where an increasing amount of data is collected and is also essential to processes within the city or to the citizens living in urban space. The data analysis and information construction is, in this geo-referenced scenario/environment, also a field which geovisualization can help to recognize inter-relations and patterns in the data and draw conclusions from the analysis of this data. Urban space can therefore be a very important 'use-case' for appropriate data management, visualization and knowledge construction, as we might need to manage our cities better in the future due to many challenges concerning the environment, increasing numbers of people moving to cities, traffic, public transport, health, urban (re-) development, etc.

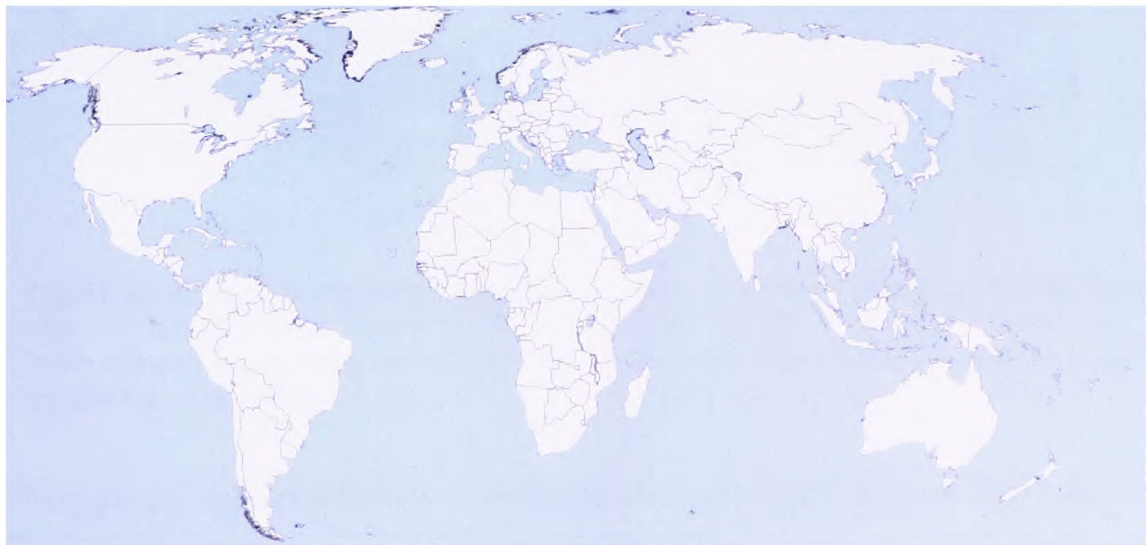
## 2. BACKGROUND & MOTIVATION

---

Reconsidering that cartography plays a vital role in terms of geovisualization this work is taking a look at map visualization and how buildings (and other urban objects) being part of 3D city models are used in (3D-) maps.

### 2.3.3.1 2D Map Rendering

Figure 2.10 shows a simple map of the earth. In this case the map is a 2D map, which means the space that holds the map content (referred to as map-space in this work) is a 2D plane. This plane is the screen plane in case of a digital map or the 'paper'-plane for a printed map. The content in this map (or the data set) are flat 2D-polygons representing the borders of countries (white fill). The map implicitly contains a further polygon describing the oceans and other water bodies on earth (blue fill). This 2D map therefore has a 2D map-space with polygonal map content. Icons or text can be placed 'on top' of the content, e.g. the flag of each country inside the country's border. The icons are not part of the actual map data, but visual elements that can help to better understand the data, thus enhancing the visual information.



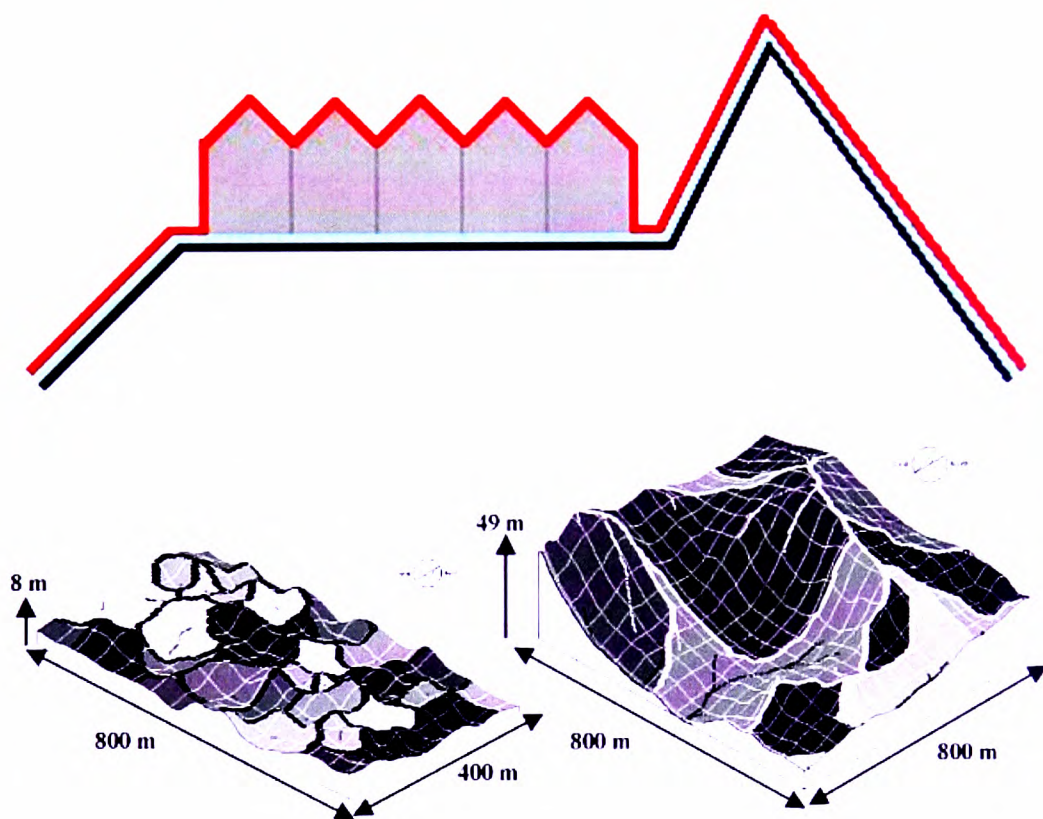
**Figure 2.10: Simple World Map** - Two-dimensional map representing the world's countries and water-bodies (from Wikipedia, accessed 10.02.2012)



## 2. BACKGROUND & MOTIVATION

### 2.3.3.2 2.5D Map Rendering

In this case the map-space, the surface on which the map content is placed, is not 2D anymore. The map-space is a 2.5D element where each x,y-coordinate is linked to exactly one z-value. These structures are normally represented by a grid or a TIN (Triangulated Irregular Network). Very often digital terrain models (DTM) or digital surface models (DSM) are used as map space in this case (see fig. 2.11).



**Figure 2.11: 2.5D-Surfaces as Map-Space** - Top: Schematic depiction of 2.5D Terrain and Surface Model. Digital Surface Model in red, Digital Terrain model in light blue (from Wikipedia, accessed 10.02.2012). Bottom: example of two DTMs modelled as grids (MacMillan et al., 2004)

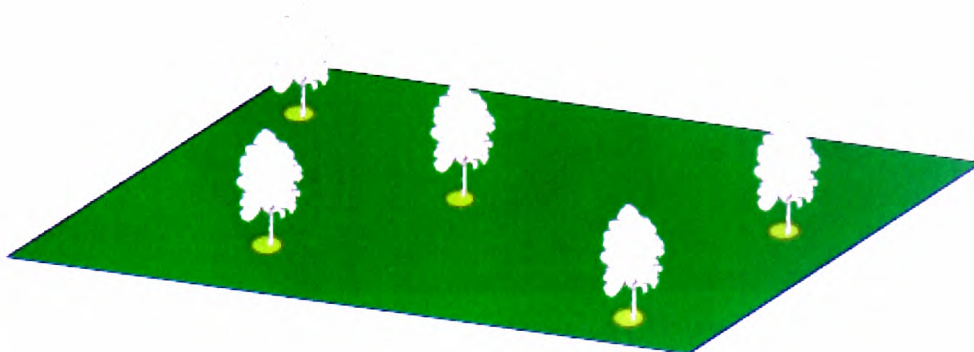
When the 2D map of the earth is projected onto a DTM the map elements are still 2D polygons, only the topography can be better understood because of the 2.5D map surface. The icons (flags) from the previous example are also still 2D as they are mapped onto the DTM as well.

## 2. BACKGROUND & MOTIVATION

---

### 2.3.3.3 3D Map Rendering

Three-dimensional map rendering is similar to 2.5D map visualization. In the case of a 3D map the map-space is often a DTM, or in some cases a flat 2D plane in 3D space. In figure 2.12 an example for the latter case is depicted. The map data is a set of points, which represent the position of trees. The 3D map visualization uses 3D icons at these positions to generate a 3D representation.



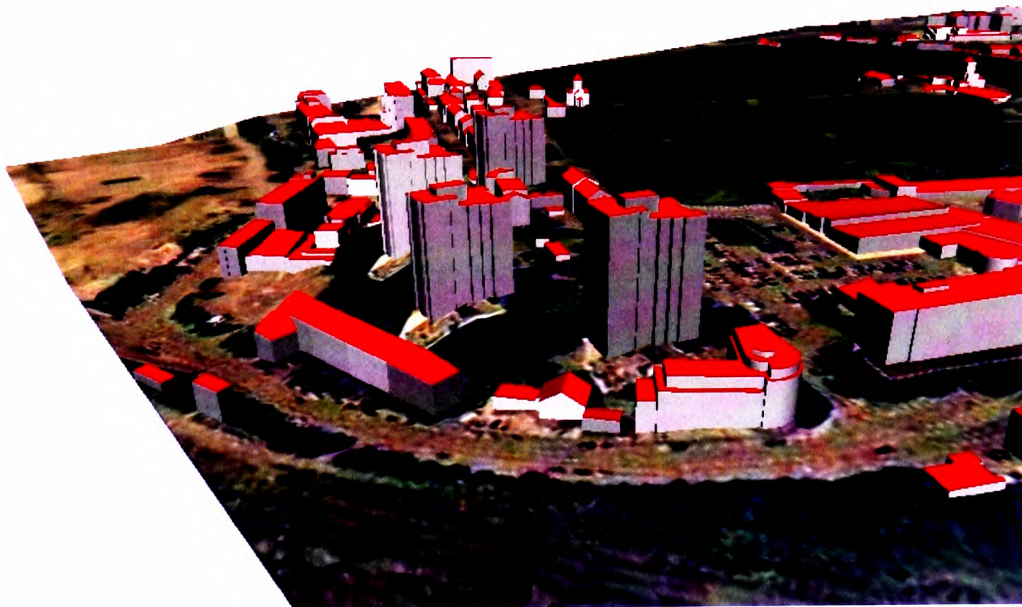
**Figure 2.12: 3D Map with Tree-Icons** - A flat map-space in 3D with tree icons generating a 3D impression

The icons are visual models of trees, no semantics or attached attributes; this is just a visual three-dimensional representation for the real world object. In figure 2.13 there is a DTM used as map-space and the map content is an aerial photo. The 3D buildings are icons that represent the buildings in the map area. Nevertheless, these buildings are just visual icons, like the trees in the previous example, in most cases they do not have any semantic structure or attached attributes, in some cases even the semantic information that the icon is a building is missing. The map-space is still the surface of the DTM and map content can only be placed here. Using photo-textured building icons does not change this situation, because the icons would just look more realistic. However, they would still be icons. The KML standard (Keyhole Markup Language)

## 2. BACKGROUND & MOTIVATION

---

(OGC, 2008b) used for GoogleEarth reflects this concept quite well. Buildings can be modelled as 3D-polygonal geometry inside a 'placemark'-element, hence an icon that 'marks a place'.



**Figure 2.13: 3D Map-Terrain with Building-Icons** - A DTM map-space in 3D with building icons representing real world buildings

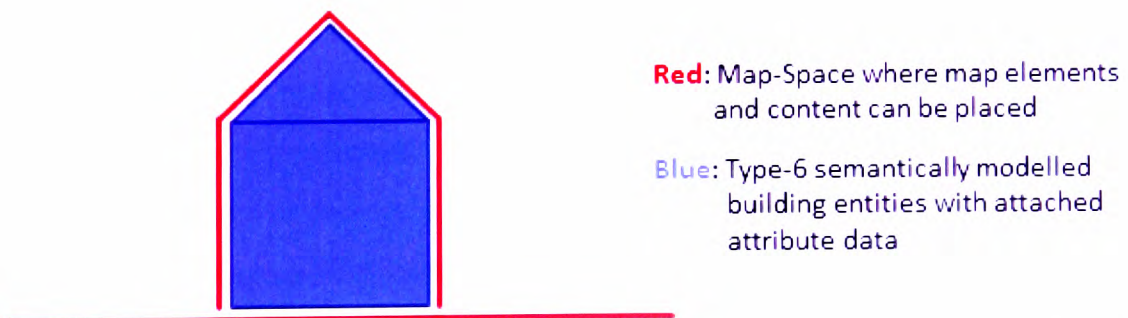
Even when certain semantic elements are modelled, e.g. separate wall and roof geometry these are often just photo-textured or coloured differently. This can be used to represent two attributes at the same time, e.g. year of construction is represented by the roof colour and use of the building by the wall colour. Nevertheless, the actual map-space is still the DTM surface. Further examples for this representation can be found in chapter 2.1.1. In figure 2.1 the map of a navigation system is depicted. It can be seen that the map content with the integrated route visualization is placed onto the terrain, which acts as the actual map-space. The 3D buildings are textured icons in the map, which provide context for the route visualization. In figure 2.2 a similar situation can be observed. The pollution information (map content) is mapped onto the terrain surface (map-space). The grey building blocks are 3D icons helping the observer to relate the map information spatially. In both cases the surface of the buildings is not part of map-space in which the actual map content is placed.

## 2. BACKGROUND & MOTIVATION

---

### 2.3.3.4 Extension of Map-Space

The contribution of this work is to change textures of 3D (building) objects. They should not solely be a container for icon textures (photo-images), which enhance the appearance of 3D icons. When 3D buildings are modelled in type-6 (see chapter 2.2.1) and linked to additional data, they do not act as icons (type-1 model). They are part of the map-data and the information should be integrated into the map illustration. However, currently the building surface cannot act as map-space because there is no way to organize the content of textures in a way map-content is organized (e.g. layers, z-order of layers, separate façade elements, etc.). The contribution of this work is a model to organize texture content and provide basic capabilities to turn them into map-space (fig. 2.14).



**Figure 2.14: Extension of Map-Space** - The map-space in 3D maps should also cover the surface of buildings and other objects in order to visualize object specific map-content

This enhancement of texture content in order to represent map-space (in basic and narrow terms) needs to be achieved by using current computer graphics capabilities and methods (e.g. a programmable Graphics Processing Unit (GPU)) to achieve state-of-the-art map rendering. This would allow interactive and/or situation dependent changes of map content, comparable to currently existing 2D digital maps (e.g. switching different content on/off). But not only is the rendering performance in focus in this work. The presented approach (see chapter 4) will also suggest ways to organize

## 2. BACKGROUND & MOTIVATION

---

texture/façade information in order to enable the management of map-content. One aspect is the introduction of a layer structure for into the texture content model.

### 2.3.4 Level of Detail and Level of Realism

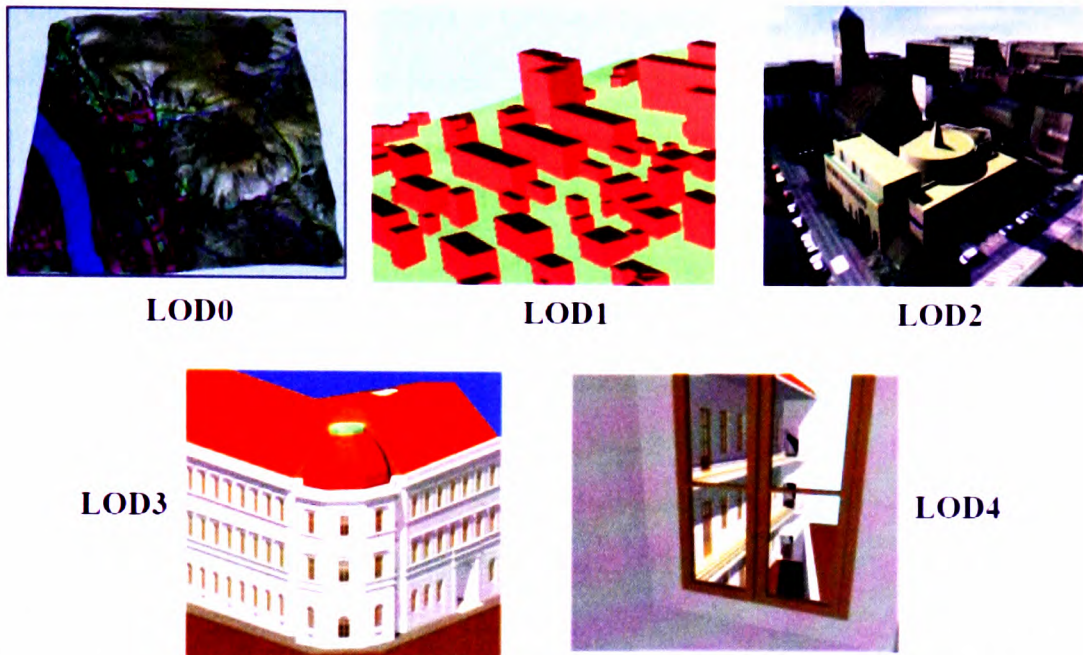
In the previous sections we learned that the surfaces of 3D objects, e.g. buildings, need to be developed into a part of map-space to be able to contain and visualize 'map-content'. But what needs to be achieved? What are the requirements for texture content and what are the capabilities these textures need to provide? In this and in the next section two basic aspects are discussed that should be investigated in terms of texturing for (3D-) map-rendering. This section is going to discuss adaptable detail in (façade-) textures. The next section looks at additional non-photorealistic information that needs to be integrated into 3D model textures.

The need for adaptable detail and the fact that one specific representation cannot be suitable for a multipurpose data model is already identified in the CityGML standard for the aspect of geometrical detail. However, the standard distinguishes different levels of detail only in terms of geometry, not for textures yet. The CityGML standard, which in this work would be regarded as a good practice when it comes to multipurpose 3D city models, defines 4 distinct LoDs for buildings (OGC, 2008a). It therefore argues that for different scenarios in which a model can be used, different geometrical levels of detail can be most appropriate. LoD in this case means one entity/feature/building does not consist of one single geometrical/semantic representation, but at maximum four different ones. They range from LoD1, the extruded footprint up to a certain (building-) height resulting in the approximate building volume, to LoD4, the detailed model of the building and its interiors like rooms, inner walls, internal doors, etc. (figure 2.15)

It can be seen that the LoD can also reflect the capabilities of the model acquisition, which leads to different levels of detail. However, scenario driven aspects do also require the use of a LoD1 model even if the LoD3 geometry would be available in

## 2. BACKGROUND & MOTIVATION

---



**Figure 2.15: CityGML LoD concept** - The five levels of detailed defined by the CityGML OGC standard (OGC, 2008a)

addition. For example, for simulations like noise or mobile phone network distribution, it would be sufficient to use a LoD1 model because the interior elements or detailed roof geometry might not be so much in focus for these scenarios.

Therefore, adapting the appearance, respectively the complexity of the model, to the particular use-case is achieved for geometry by the introduction of the LoD concept. The equivalent concept for textures is missing. In order to have the flexibility to generate purposeful representations adapted to the particular scenario textures, or to be more precise the texture content, should also be able to adapt to user and scenario requirements.

The question is: how can a new texture content model reflect different levels of detail in order to provide multipurpose representations? Hence, how can differently detailed representations for (façade-) textures are achieved in order to reflect the idea of extended map-space (see section 2.3.3.4). When the extended map-space on building surfaces should be able to hold different content in terms of detail, similar to the LoD concept for geometry, the new texture model should integrate this idea.

## 2. BACKGROUND & MOTIVATION

---

Conceptually it needs to be possible to define a façade texture LoD in order to be able to differentiate between diverse façade representations in terms of detail. As the term 'Level of Detail' is already used for the geometrical detail<sup>1</sup>, this work introduces the term Level of Realism (LoR) to reflect the level of detail in textures. The purposes of introducing the LoR concept are quite similar to those for the LoD. On the one hand it should reflect the capabilities of the acquisition method, for example, window extraction from façade photos. This method can extract window positions and the portion of the texture that represents the window element. This information might result in a façade texture with a 'background material' and the window elements. Other features of the façade were not extracted. On the other hand the LoR should also reflect different requirements for different scenarios in terms of visualization, as we learned that photo-realism or naturalism is not always the suitable solution.

An important aspect, for LoD and LoR, that one should definitely recognize, is that both concepts are not changing the 'resolution' of the model, but the 'density' of the information and content. Both concepts are not aiming to simplify the content, but to omit a specific, semantic portion of it.

For the geometrical part of a 3D city model there are approaches to simplify geometry (Kada (2007), in chapter 5) and still preserve the main characteristics of the object, which does not work for many mesh simplification approaches in computer graphics. These approaches generally simplify the geometry of the object; hence try to reduce the number of polygons or triangles (geometric primitives). However, the LoD concept specifies certain features that are present in a higher LoD but not in a lower one (e.g. interiors, roof geometry, etc.). For the LoR concept this should work in a similar way. For a texture not the resolution should be changed (filtered/mip-mapped) to change the 'detail' and e.g. reduce size, but the content of the texture should be adaptable. It should be a matter of which semantic entities (windows, doors, ledges, signs, etc.) are

---

<sup>1</sup>LoD has a further meaning in computer graphics: Here the LoD refers to numbers of triangles in a mesh, respectively the resolution of an object's geometry

## 2. BACKGROUND & MOTIVATION

---

going to be included into the texture. In the presented work the author refers to this aspect as 'information density' or 'density of façade elements'.

But not only is the density of information in a façade texture an aspect that is investigated in this work. As already mentioned it is also important for multipurpose models to be able to provide multipurpose representation. Therefore it is not only interesting how the information density can be adjusted by concepts like the LoR, it is also valid to ask if it is useful to integrate and combine different information in the façade representation (real world façade elements + non-visual attribute data).

### 2.3.5 Additional Information in Façade Textures

At this point we are coming back to the idea of 'purposeful pictures' (see Schirra and Scholz (1998)) that are providing the right information for a specific task that a 3D visualization is used for. The discussion on the LoD/LoR concepts shows that the adjustability of the amount of information can be beneficial in certain cases. This adjustability, however, focuses on naturalistic elements of the façade and how many of them should be present in the façade texture. In the end the user would be able to choose between different depictions like 'full detail', 'wall material plus windows/doors', 'material only', etc.

Nevertheless, in specific scenarios it can be also important to add additional information to the texture, which is not part of the naturalistic appearance, in order to make it visible to the user. 3D city models, especially sophisticated type-6 models, normally provide links to attribute data sets at least at building level. This additional information is one more aspect that differentiates the multipurpose models from purely graphical models. As we find semantic entities with a meaning and a purpose, properties of the semantic entities can be linked even if they are stored in another data set (when sharing a common ID). Visualizing attribute values in the 3D scene is often achieved by colouring the building, or a building's geometrical parts (roof, walls, ground) according



## 2. BACKGROUND & MOTIVATION

---

to a colour scale representing the values the attribute can have. One example would be the year of construction. Buildings can be coloured according to the year they were built and a kind of 3D thematic map is created. This works quite well for a single attribute that refers to the whole building. In principal, this visualization method is based on the idea of using differently coloured icons (see section 2.3.3). In this case, for example, icons for buildings built in 1990 are coloured green, buildings in 1995 in blue, and so on. Hence, the icon can represent one attribute at a time. This approach can only reflect the extension of map-space in a very limited way.

In another example this solution is much more complicated. Having a building with a shop on the ground floor and a flat on the first floor and an attribute data set storing the two types of use it is rather complicated to symbolize this information by colour coding when there is no explicit geometry for building floors. Current models mostly consist of walls representing the building's outer hull. It is quite difficult to colour the upper portion of the wall geometry in one colour and the lower portion in another colour. This becomes even more complicated the more information is intended to be visualized. Therefore as there are no explicit geometrical representations of façade elements and building floors to which the colours can be applied one solution would be to look at the texture in this case as well. One would need to suggest a definition for texture content that allows specifying that the upper half of the texture should have colour A and the other half colour B. When the texture is applied to the wall geometry the colours can represent the two different types of use although the separate floors are not present and there is only a single geometry for the wall. Besides the LoR this is another way to use textures scenario specific and not only for enhancing 'realism'. Managed and modelled in an appropriate way they can also answer questions discussed by Ferwerda (2003) and Schirra and Scholz (1998) mentioned in earlier sections of this chapter.

A further advantage of combining information inside a texture can be very useful for knowledge creation and understanding the links between different information and coming to new conclusions. Combining information and changing the content by switch-

## 2. BACKGROUND & MOTIVATION

---

ing to other combinations in real-time might also be useful in this case. Using the example from above one might want to mix the two colour-coded areas (top residential, bottom for commercial use) with e.g. an electricity consumption graph for the last month and superimpose the two types of information to solve a specific task analysing the situation. This flexibility in terms of content, combination and density of information is already achieved in other disciplines and systems (e.g. digital maps, GIS, CAD, etc.) by using layers. This concept is well known and widely accepted and therefore this work is going to investigate if and how it can be used within the presented texturing model for façades in 3D city models.

### 2.4 Summary

In this chapter the current state-of-the-art in the field of 3D city models was presented by providing examples of their current use and different ways of data acquisition. It becomes obvious that digital 3D urban models are used in various scenarios and in very different forms in terms of semantic detail.

The different forms of manifestation of 3D models was addressed by adopting the six types of models defined by Stadler and Kolbe (2007). The definition of type-1 to type-6 models is used throughout this work in order to refer to the differently modelled information and semantic content of 3D city models and the resulting consequences of visualization.

As type-6 models transport much more information about urban objects they can be used in many more ways than photo-realistic visualization. This chapter described why these models need to be managed in a flexible and efficient way in order to serve many different scenarios and use-cases. Different clients must be able to access the exact right portion of a model and to receive the appropriate information that is necessary to fulfil a specific task. When flexible systems are used to manage the models, these

## 2. BACKGROUND & MOTIVATION

---

models can be integrated into a wider infrastructure in order to server as a 3D data repository in a 3D-SDI and to server in multiple process chains built by distributed systems.

As described in this chapter the type-6 models can be used in multiple-scenarios and for very different purposes, of which photo-realistic representation of the real world is only one possible solution. Visualization of city models in a 3D-map can be realized by 'placing an icon' in the real world position, as a symbol in a 3D map. However, it could be more useful to extend 'map-space' onto the surface of the 3D object by defining a new texture content model, which enhances the capabilities for map-like visualization. Evidence for the need of this type of visualization can also be found in related fields of science, where it is argued that photo-realistic representation is not always the right way to visualize 3D scenes. Hence, the extension of map-space can lead to a more 'scenario/task centred' approach for 3D map-rendering. Extending map-space can be regarded as useful for a very simple reason: the more space is available for map content, the more information can actually be transported by a map. Of course, this assumption is a much reduced view on information visualization as not only the quantity of the information is relevant but also the quality of the information visualization approach. However, it seems sensible that extended map space is beneficial because there is more available space that can be used to place and visualize information. This information still needs to be shown in a useful way, of course.

Two aspects for the new texture content model, hence 3D map-space on building surfaces, were identified and suggested as a first step towards the extended map-space. This chapter discussed issues on Level of Detail/Realism as well as ideas on how additional information (besides realistic façades elements) can be integrated into the model textures. These two aspects as well as other related issues will be investigated and addressed by the texture content model presented in chapter 4.

# Chapter 3

## Methodology

This research focuses on the rendering and visualization of city models as part of 3D maps. In chapter two this work outlined several approaches for map rendering and how 3D building models are used inside 3D maps. Looking at the concept of using 3D building objects as icons appears to be somehow limiting when the actual data set that is referred to as type-6 is taken into account. Therefore it has been outlined that this limitation could be overcome by extending map space to the surface of the 3D building entities. In this chapter ways in which the question of (façade) texturing of 3D city models can be investigated are addressed. Validation of the suggested solution is also considered.

The chapter is subdivided into two major parts. The first part is going to discuss the constructive research methodology and in which way it is used in this PhD research. The second part is going to describe three research methods used in this work in order to perform the six stages defined for the constructive research process.

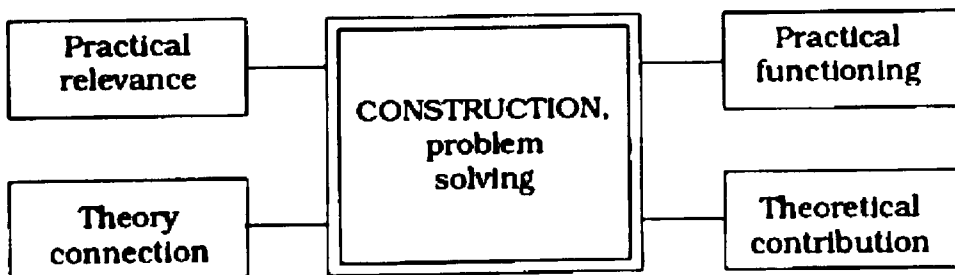
### 3.1 Constructive Research

The constructive approach in this methodology tries to solve problems through the *'construction of models, diagrams, plans, organizations, etc.* Several examples of applied constructive studies are found in technical sciences, in clinical medicine and in operations research. Some are found in management accounting. Mathematical algorithms and new mathematical entities provide theoretical examples of constructions.

### 3. METHODOLOGY

---

*Constructive research can be found even in philosophy in those cases where the world is constructed, step by step, from supposedly basic elements like objects, time-space slices, observations, thoughts or logical relations. Creating an artificial language (e.g. Morse alphabet, Braille's alphabet, computer languages) is an example of a construction at its purest'* (Kasanen et al., 1993). The actual aim of constructive research is to create new knowledge about a certain aspect in a specific domain by constructing an adequate artefact/construct, which solves an existing problem. Therefore, the research is based on a concrete problem that has been identified by the researcher in a specific field of science. The new knowledge is created during/by the construction process of the solution. *'Constructive research gives results which can have both practical and theoretical relevance. The research should solve several related knowledge problems concerning feasibility, improvement and novelty'* (Crnkovic, 2010).



**Figure 3.1: Constructive Research** The idea of constructive research solving practical and theoretical problems. (Kasanen et al., 1993))

The important part about the construction or the constructed artefact is stated in Crnkovic (2010): *'A construction, be it theoretical or a practical one, when it differs profoundly from anything previously existing, constitutes a new reality against which a pre-existing one can be examined and understood, so it has an undeniable epistemological value.'* As stated earlier, the knowledge gain in constructive research takes place during the construction process and by the evaluation with regards to the situation that existed be-

### 3. METHODOLOGY

---

fore the innovation was introduced by the researcher. However, not only the question of solving a real world problem contributes to the assessment of the research work. Also the comparison against the previous situation generates new insights and creates knowledge. Checkland and Holwell (2007) describe the same aspect for the field of action research (AR), a similar research approach: the existing framework of ideas (F), the methodology (M) and the area of interest (A) need to be described in detail, before the concrete action is performed. The *'susceptibility to change F, M and A in research in which the researcher becomes involved in the flux of real-world social situations leads to a (or probably the) most important principle in AR... In keeping your intellectual bearings in a changing situation in which the adequacy of F and M and the appropriateness of A are likely to be tested, it is essential to declare in advance the elements of F, M and A. ... Without that declaration, it is difficult to see how the outcome of AR can be more than anecdotal'* (Checkland and Holwell, 2007). Although AR focuses more on learning from a social situation, whereas constructive research is concerned on the process of creating a novel artefact, the aspect described is also essential for the latter.

#### 3.1.1 Six Stages of Constructive Research

The research process for constructive research is described by the following six stages:

- Find a practically relevant problem which also has research potential.
- Obtain a general and comprehensive understanding of the topic.
- Innovate, i.e., construct a solution idea.
- Demonstrate that the solution works.
- Show the theoretical connections and the research contribution of the solution concept.
- Examine the scope of applicability of the solution.

### 3. METHODOLOGY

---

*'The innovation phase is often heuristic by nature; stricter theoretical Justification and testing of the solution typically come afterwards. The innovation phase is the core element of a successful constructive study'* (Kasanen et al., 1993). Constructive research is not necessarily a linear, but more likely an iterative process in which the working solution (stage 4), the research contributions (stage 5) or new knowledge about applicability (stage 6) lead to a better understanding of the problem (stage 4) and this leads to further innovation.

In figure 3.2 the research architecture in this work is depicted and the separate elements of the process are related to the desired research outcomes. One can see that the results of each stage are the basis for the next element and the process is consequently built on the previous elements, which is related to the structure of the constructive research methodology. In this diagram the elements 'City Models' and 'Current Visualization' covers stages 1 + 2 of the constructive research process, the element 'Texture Model' covers stage 3, 'Proof of Concept' realises stage 4 and the element 'Case Study' covers stages 5 and 6 of the constructive approach.

On the left side the connection between elements that form an iterative process are depicted. As described above it is desired that stages 4, 5 and 6 produce new insights into the area of interest and create new knowledge about the visualization of models and the relevance for research. Therefore they enable the researcher to acquire new knowledge which can lead to further innovation of the constructed artefact.

## 3.2 Realizing the Stages of the Constructive Approach

In order to actually perform the defined stages above this work uses three research methods that appear to be most appropriate to cover the constructive research process:

### 3. METHODOLOGY

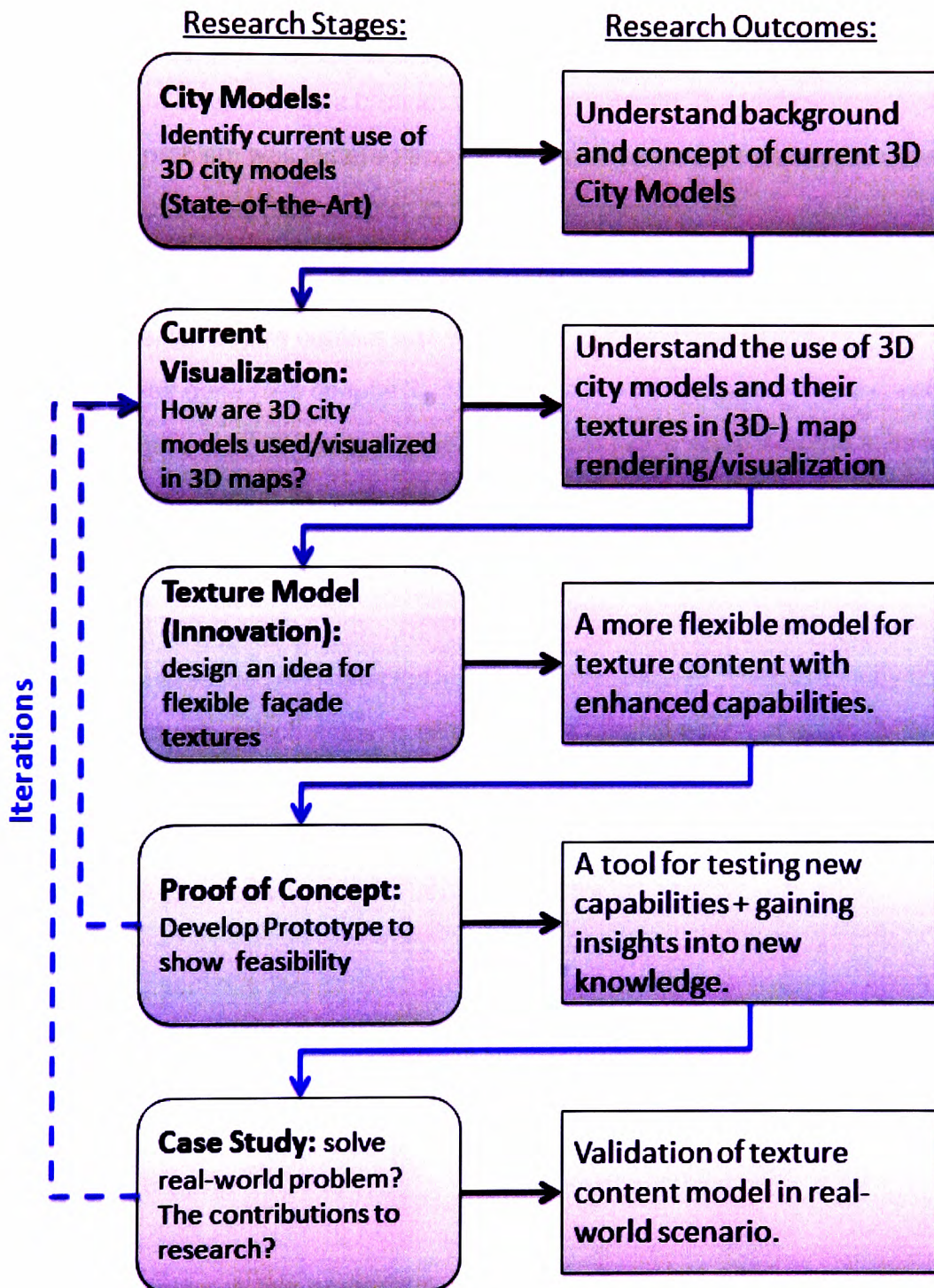


Figure 3.2: Research Stages - The steps taken in this work in order to conduct the research project (schema structure adopted from Wang (2007))



### 3. METHODOLOGY

---

- **'Learning' - phase** (stages 1 + 2): in this phase of the presented research the state-of-the-art is analysed in various respects of the investigated field of science. This is mainly done by a literature review. Closely related disciplines and their findings are included into a broader view of the subject. The understanding of the research problem was fostered and this also reinforced the motivation to change the state-of-the-art (see chapter 2).
- **'Innovation and proof-of-concept'- phase** (stages 3 + 4): In this phase the new model for texture content was developed, hence the construction of the new artefact was done (see chapter 4). In order to perform stage four of the constructive research approach a prototype was developed to show that the solution is feasible (see chapter 5).
- **'Validation' - phase** (stage 5 + 6): In phase research contributions are presented by conducting a case-study. Here the capabilities of the new texture content model are assessed and reflected in a real-world scenario. The case study shows example of use, which helps to estimate the potential scope of applicability and to understand the degree of novelty and feasibility. By generalising the findings in the case-study it is also possible to create new knowledge for the wider research field of 3D city models and information visualization.

In the following sections the three phases of literature review, prototyping and case-study are explained in greater detail.

### 3.3 Literature Review

The literature review in this phase builds the basis for the emergence of the texture content model in this work. A profound knowledge of the structure and use of 3D city models as well as the technical capabilities in terms of rendering 3D scenes is crucial to come to a feasible and useful concept.

### 3. METHODOLOGY

---

*'Presenting what has been researched and written on a subject is one way of showing what needs to be done'* (Murray and Hughes, 2008) and therefore grounding one's own work upon existing work and the gap in the existing research and knowledge. *'By building on the findings of previous studies'* and *'by taking them a step further'* (Murray and Hughes, 2008) the rationale for a research project can be provided. The rationale for this work can be extracted from the presented scenarios in literature in which 3D models are used. Another aspect that can be found in literature is a new concept for modeling 3D city models (type-6 models, see chapter 2), which indicates a need for different visualization forms. Furthermore, meetings in the MoNa3D project also provided certain requirements for the specific use case of pedestrian navigation, which also acts as a case-study in this research. Another important aspect about the literature review was to learn about the existing standards and system interface definitions that are used in the field of 3D city models and the related system infrastructures. Integrating the new texture model into the technical environment and existing standard definitions is an important objective in this work and good knowledge about the related standards and concepts is indispensable. A further aspect where the current state-of-the-art needs to be investigated is the capabilities of current systems for 3D model management and especially the capabilities and algorithms for 3D rendering. An in-depth literature review is also necessary in this field.

In general the literature review should provide an in-depth knowledge about work of other researchers. Research is always conducted in context of previous work and is not started independently from all other research. That means knowing about concepts of others can help to apply concepts from related disciplines to own problems and research tasks, merging approaches and extend algorithms instead of inventing completely new ones, for example.

Murray and Hughes (2008) describe the need of a profound literature review as: *'show where your study fits into the broader scheme of things; how it connects with the existing body of knowledge on the subject or on other related issues. In doing so, it also*

### 3. METHODOLOGY

---

*shows how your own research is original and promises to contribute to that pool of knowledge. In other words, along with the introduction, it helps to contextualize or 'position' your research by placing it within a broader framework. This also helps you to avoid reinventing the wheel by needlessly repeating the work (and mistakes) of others.*

This can lead to efficient and effective processes for generating new knowledge and findings on a reliable basis (previous work). In this work concepts from computer graphics, procedural modeling and rendering techniques are used and extended in order to define a new texturing concept tackling research questions for geovisualization for 3D city model façade visualization. Concepts from computer graphics on task driven non-photorealistic rendering support the idea of flexible façade textures for the aforementioned type-6 models. In this way literature review helped to support the idea of a more flexible façade representation and that it is worth to be investigated and to define a new model for (façade-) texture content.

The literature review and the reviewing of existing systems and other related disciplines also provided concepts and technical approaches that appeared to be very useful for the required flexibility of the new texture concept. One example is the layer concept that is part of the proposed texture concept approach. This method of arranging content is successfully used in other fields and systems and appears to be well understood, which would give a new texture concept a strong and reliable basis in terms of content management.

In general one can say that the literature review lays the basis for defining the new texture content model (Objectives 1 + 2) and also for integrating this concept into 3D-SDI environments (Objective 4). For defining the new texture model as well as the integration into the wider 3D-SDI infrastructure it is essential to know existing concepts and work as well as specifications and technical solutions (e.g. from related disciplines). Besides conferences, workshops and project meetings where these things are discussed, literature review is a very important source of information and knowledge. In addition to these aspects the second function of summarizing relevant work and the

### 3. METHODOLOGY

---

state-of-the-art is to describe the existing situation and reality, in terms of the constructive research approach. As this methodology defines an innovation process as its core element of knowledge creation it is essential to describe the status quo in order to be able to evaluate the innovation and acquire new knowledge by comparing the situation before and after the innovation.

#### 3.4 Proof of Concept by Prototyping

In order to prove the realizability in terms of technical aspects a prototype is produced (chapter 5) that implements the functionalities that are conceptually defined.

*'A prototype is the partial implementation of a system built expressly to learn more about a problem or a solution to a problem... A software prototype implements part of the presumed software requirements to learn more about actual requirements or about alternative designs that could satisfy the requirements'* (Davis, 1992). In this work the main method used is *'evolutionary prototyping'*, which *'builds quality systems from the start - evolutionary prototypes - which are evolved over time'* (Davis, 1992). In contrast to *'a throwaway prototype an evolutionary prototype*

- *is built in a quality manner (including a software-requirements specification, design documentation, and thorough tests)*
- *implements only confirmed requirements (after all, why implement poorly understood requirements when you know you'll probably understand them much better after you build and implement the first prototype?)*
- *is used experimentally*
- *is used to determine what requirements exist that haven't been thought of*

*When the prototyping is complete, the developer modifies the software-requirements specification to incorporate what was learned... To minimize risk, the developer does*

### 3. METHODOLOGY

---

*not implement poorly understood features... In contrast to conventional development, which attempts to build the entire set of requirements, the evolutionary prototyping acknowledges that we do not understand all requirements and builds only those that are well understood' (Davis, 1992).*

As one can see (evolutionary) prototyping in software development allows iteratively developing a system that is not fully implemented and only provides a subset of specific features, however, for the existing features it is fully functional and therefore can be tested against the requirements that are specified for the implemented subset. When the subset implementation is verified and works according to the specification another implementation cycle can be started and further functions can be added. By prioritizing defined requirements in the conceptual phase according to their importance for the envisioned concept it is possible to define exclusion criteria. If one of these criteria is not met, then it is not necessary to start implementing less important or not fully defined functions. Instead it is possible to plan modifications of the concept, change the (hardware-) environment or to define certain preconditions for the feasibility of the proposed concept. Prototyping and an iterative development process allow splitting the overall research task into smaller units, keep track of the findings and related consequences and in case sub-tasks are prioritised it also helps with risk management and counter measures.

Nevertheless, prototyping is not only a measure for the technical development of systems and technological feasibility. It also links back to the conceptual phase. When a prototype shows significant weakness in performance or is not easily extended to the next development level, this also indicates drawbacks of the underlying concept and is an indication that the concept should be reviewed.

One final aspect about prototyping is discussed at this point, which is also beneficial in a research project. Developing a prototype allows implementation of a concept and shows its feasibility and properties as well as its appearance and some basic functionality without the necessity to implement a complete scenario dependant system

### 3. METHODOLOGY

---

providing all end-user functions (e.g. intuitive user interface). This is the case when there are certain time constraints (remaining time of the project or funding) or when specific resources are not available at a specific time. This can happen if previous work is done in a specific programming language and should be used, a prototype in this language can be developed. Porting the prototype and developing it further in a new programming language or environment can be part of future work, when the feasibility is verified. When certain hardware is not available (e.g. smartphones with certain hardware configurations) and the suitable products are just about to enter the market, the prototype can be developed on another hardware (e.g. a PC) that can emulate the intended platform. In this aspect prototyping can also help to 'secure' findings and extend or port them to other scenarios in the future. This was the case for the MoNa3D project, where the required graphics chips were not available at the time of developing the prototype. The project team therefore decided to implement the texture concept prototype based on available graphics hardware for PCs and port the implemented solutions to mobile devices as soon as smartphones with the required graphics hardware enter the market. This enabled the project team to test the concept even before the final target platform was available.

Evolutionary prototyping is used in this work to achieve objective 3 in order to show that the concept can be realized technically. The prototype also proves that content can be changed interactively in real time and allows user interaction or context sensitive changes. This aspect also supports the multipurpose nature of today's models in contrast to pure visualization models where only one texture representation is loaded together with the model and does not change during the visualization.

### 3.5 Validation by Case Study

Case-studies are one method of testing a hypothesis. The case-study tries to evaluate how good a certain concept (the hypothesis) works in a given context and to generalize

### 3. METHODOLOGY

---

the findings to a general problem. That means this method tries to draw conclusions from a significant case and to estimate if this performance is valid generally or in other contexts.

Gillham (2000) defines a case-study as a study *'which investigates' ... a phenomena. ... 'to answer a specific research question. ... and seeks a range of different kinds of evidence, evidence which is there in the case setting, and which has to be abstracted and collated to get the best possible answers.'* The case-study should therefore provide a specific environment for the case that is investigated to ensure its significance.

For Gillham (2000) a case is:

- a unit of human activity embedded in the real world
- can only be studied and understood in context
- exists in the here and now
- merges in with the context so that precise boundaries are difficult to draw.

In the presented work the evaluation in terms of constructive research (stages 5 + 6) is done by a case-study (chapter 6).

The case-study that was chosen for this research is mobile pedestrian navigation. One reason is the participation of the author of this work in the MoNa3D project, which investigated the use of 3D city models for pedestrian navigation. Nevertheless, the scenario of using 3D models for pedestrian navigation support also fits the formulated research problem. As pedestrians navigate urban space, the city as such is part of the problem that needs to be solved and at the same time part of the solution of the task, as pedestrians make use of landmarks within the urban environment. Therefore the 3D model is not only a visual 'background' in order to provide a (photo-)realistic representation of the environment, it is used to provide additional information to the user (landmarks) and the model should be visualized according to the scenario requirements.

The aspects on the integration of the new texture content model into the technical environment (3D-SDIs) that is described in chapter 4 are also an important issue that is rel-

### 3. METHODOLOGY

---

evant in the case-study. The MoNa3D project defines a system architecture based on an open data infrastructure. The capability to integrate the new texture model in such an infrastructure appears as a crucial requirement in the investigated case. Therefore the case-study provides general constraints and requirements for the integration of the concept into the wider technical environment of 3D city models. When real-world applications normally exist in a specific environment a newly developed concept needs to fit into this environment as well, otherwise it would not be accepted in the specific domain.

Evidence for this case-study was acquired by accessing and discussing project outcomes and by attending project meetings. However, the scenario was also investigated by matching the texture model capabilities to requirements that are formulated for pedestrian-navigation in relevant literature (see chapter 6.4).

It was necessary for the presented work to put parts of the case study onto a theoretical level as prototypes of the pedestrian navigation system in the MoNa3D project did not evolve to the stage where they could have been used for user-tests (the future stage of user tests would also be better performed by action research methodology). Prototypes developed in the project showed that existing 3D data sets (provided by one of the project partners), which navigation system manufacturers use in their products can be used for pedestrian navigation systems. Also the textures in the data sets (tile-based) could be used for the zone/layer-based texture approach that is presented in chapter 4. The concept was implemented on a smartphone during the project. However, the reconstruction of the complete façade texture was done in a pre-processing step on client side and not on the graphics chip during rendering. The existing smartphone prototype developed in the MoNa3D project also integrated navigation hints into façades for one test route, however, other capabilities like façade detail adaption, real-time changes of routes and real-time appearance changes of landmarks were not implemented.

The prototype did show some results for specific questions asked in the project, however, it was only just about to reach the level where it would have been possible to



### 3. METHODOLOGY

---

conduct user tests (sending users out into the streets). It was therefore not possible to validate visualization concepts for the given scenario with users in a real-world setting. However, in order to evaluate the fulfillment of requirements for the layer/zone-based textures it is sufficient to present profound reasoning for aspects of the case-study to validate the presented texture model. Real-world observations might support these findings but are not essential to evaluate the façade texture model.

As the case study was based on the MoNa3D project scenario, which defined a system that supports pedestrian mobile navigation by using 3D city models, the investigated case was based on previous work and findings in research in this field. Sources in literature suggest that pedestrians navigate in a different way to cars, e.g. make much more use of landmarks, etc. These findings were incorporated into the project aim and goals were defined for appropriate visualization of landmarks and other buildings to optimally support the navigation scenario. One example for such a goal was to be able to adapt the detail of specific buildings and adding additional information.

Hence, as the scenario requirements are known, in terms of visualization capabilities, they can be checked against the 'proof-of-concept'-prototype. This prototype just visualized one building/landmark and generates a kind of 'laboratory'-test, however, this is sufficient to test if the project requirements are met. As the prototype also implements the reconstruction process inside the rendering pipeline and allows real-time changes of the texture content, it can be estimated if solutions for further requirements found in literature can be met. This part of the validation can be done without a fully functional navigation prototype, because just the capabilities of the new texture model are evaluated taking the requirements defined by the case-study scenario into account.

However, the benefit of a fully functional pedestrian-navigation prototype would be that the concept can be tested in terms of user-friendliness, navigation-support, cognitive load, etc. Nevertheless, as these aspects are outside of the scope of this work it is sufficient to know the visualization-dependent requirements of the scenario and to in-

### 3. METHODOLOGY

---

investigate if the proposed texture model can meet these requirements. A fully functional pedestrian navigation system is not necessary to evaluate these visualization aspects.

By evaluating how well the presented concept can meet the scenario requirements in the case of pedestrian navigation this work attempts to generalize the findings and estimate how the concept would perform generally and how flexible it is (Objective 5). In terms of constructive research the case-study actually tests the scope of applicability and the novelty of the introduced innovation. Regarding knowledge creation the new artifact should be tested against the situation before the innovation was introduced. In the case-study presented in chapter 6, this work compares visualization capabilities of 'traditional texturing' to the new texture content model and how well this model can address requirements formulated in relevant literature on pedestrian navigation. By generalizing the findings for the investigated case it should be possible to estimate the contribution of the new texture content model for the wider field of 3D city model use-cases. Finding general results for the wider research field by looking at a specific case is the actual goal of the case-study.

#### 3.6 Summary

The presented methodology is based on six stages, which are realized in three more general phases in this work.

Stages one and two are addressed by a literature review. An overview about the research field, a description of the actual problem and its research potential as well as the general motivation for this work can be found in chapter 2. This phase of the work is an essential one, as it describes the currently existing reality (and its lack of knowledge). This existing situation needs to be described carefully because it is going to change due to the introduced innovation. In order to be able to assess the innovation

### 3. METHODOLOGY

---

and its contribution to knowledge the current situation before the intervention needs to be declared.

Stage three of the research process, the actual innovation, can be found in chapter 4. Here the novel model for texture content for 3D city models is presented.

Chapter 5 of this work describes the second phase of the work, which actually covers stage four of the constructive research methodology. Here the feasibility of the model is tested by developing a prototype. Proving the feasibility of the innovation is necessary because constructive research also aims on solving a practical problem (map rendering in this work), besides creating new knowledge. Moreover, as the new knowledge is created by constructing a solution for a real-world problem, the new knowledge can only emerge when a functional solution is generated. Hence, proving the constructs feasibility is part of the knowledge creation process.

The third phase of the work covers stages five and six where research contributions are extracted and where the scope of applicability is assessed. In this work a case-study is performed and general findings are deduced concerning texturing/visualization of 3D models in a pedestrian navigation scenario (see chapter 6). By generalizing the findings of the case-study it is also possible to formulate general findings for the wide field of 3D city modeling and visualization.

# Chapter 4

## Texture Content Model

In chapter 2 we learned that building objects are often used as icons in 3D map illustrations. They provide one fixed visual representation of the object in order to generate a three-dimensional visualization that is very often focused on photo-realism. In some cases one can find coloured icons to represent a specific attribute for the buildings, e.g. red icons for residential buildings, etc. Looking at textured icons using standard photo-image textures, the textures can add missing (façade-) elements visually. However, this is only one specific, hardly changeable, photo-realistic way. As described in chapter 2 it would be beneficial to use buildings not only as icons because type-6 city models provide much more information about urban objects and urban space. Therefore, in order to visualize this information included in type-6 models it would be useful to extend map-space to the surface of these objects (compare chapter 2)

In this chapter a new model for texture content in 3D city models is suggested. This new model tries to provide capabilities in order to represent content in a way comparable to a digital map, where content can be combined and ordered according to specific scenario requirements. At the same time as 'map-issues' are addressed, the new model still keeps rendering issues in mind as well as the nature of textures in 3D computer graphics and tries to find joint solutions. In this chapter and the remainder of this thesis the focus is put on building façades as they appear to be the most appropriate and prominent surfaces for extra map-space inside urban models. However, the new content model should be valid for other surfaces as well and should be able to be applied accordingly.

#### 4. TEXTURE CONTENT MODEL

---

The presented concept is built upon the construction/synthesis of textures and it uses three elements, which are necessary to generate the final texture representation. The three elements are:

- **The description** - defines the structure as well as elements and additional information that are to be included into the façade texture. The description in general terms defines the appearance and the content of the façade texture. Furthermore the description can also be a specialised and extended specification of content that can be used in an optimized way by a reconstruction implementation on a specific platform.
- **The texture tiles** - hold the actual image information. Normally they contain the visual representation of a semantically defined element (e.g. a window). But texture tiles can also be used for other purposes besides the standard use, for example, integrating a thermal image into the façade texture as a single tile in a separate layer.
- **The reconstruction implementation** - this is the process that constructs the final façade texture using the description and the texture tiles. This process can run on different levels of complex system architectures (e.g. client-server), in different scenarios and on different hardware platforms.

The chapter starts with some definitions of basic elements (image, texture, video, texture atlas) that are needed in order to build the new model on top, thus understand the new content model. It is important to understand these basic elements and the fact that this work does not want to change their nature, e.g. an image/texture being a pixel matrix. However, the new model tries to change the way texture content is defined and managed for the field of 3D city models. After the description of the basic elements the chapter presents elements and the structure of the new texture content model (zones, layers, etc.) and how they are organized. As zone-, layer- and other elements only

## 4. TEXTURE CONTENT MODEL

---

describe the skeleton of the content the chapter also provides a section on how this frame is filled with actual content, e.g. how texture tiles are applied to zones.

The final part of the chapter looks at the environment (see chapter 2.2) in which 3D city models often exist, how the texture content model fits into this environment and how it can contribute to flexible and effective solutions in client-server scenarios within (3D-) SDIs.

### 4.1 The Image

This work often uses the notion 'image' (the term 'texture', which is also used regularly in this work is explained in a later section). This term is mainly used for digital images that are generally used in the field of information technology. Images can be produced by digital cameras, airborne camera systems, satellites, apparatus in medicine (like a computed tomography), etc. In this work images are often referred to photo-images taken from a façade in order to generate a texture of it, but also other sources of image data will be described.

A general definition of the properties of a digital image is given in Poynton (2003):

*'A digital image is represented by a rectangular array (matrix) of picture elements (pels, or pixels). In a grayscale system, each pixel comprises a single component whose value is related to what is loosely called brightness. In a color system, each pixel comprises several components usually three whose values are closely related to human color perception.'*

Looking at a digital image in more detail it is a '2-dimensional, regular matrix of values. In a more formal way, a digital image  $I$  is a two-dimensional function of coordinates  $\mathbb{N} \times \mathbb{N}$  in a set of image values  $P$ , which is:'

$$I(u, v) \in P \text{ with } u, v \in \mathbb{N}$$

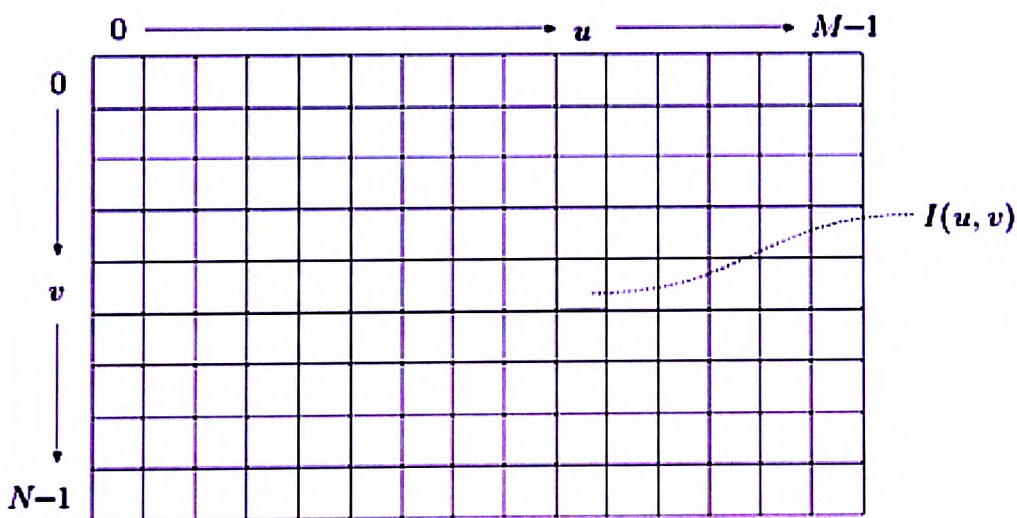
#### 4. TEXTURE CONTENT MODEL

---

*'The size of the image is therefore defined by its width  $M$  and height  $N$  of the image matrix  $I$ . The resolution of the image is specified by its spatial extent in the real world, e.g. 'dots per inch'(dpi) or 'lines per inch' (lpi)... or in kilometres per pixel for satellite images. In most cases it is assumed that the resolution of an image is identical for each direction, hence the image elements are square. This is not necessarily true as most video cameras show non-square image elements' (Burger and Burge, 2005).*

Defining a position in an image is done by specifying the coordinates of the pixel in the matrix coordinate system. For digital images this coordinate system varies from the mathematical one.

*'In order to define a position in an image a coordinate system is needed (fig. 4.1). For digital images the general convention is to place the origin in the upper left corner. The x-axis is the normal horizontal axis but the y-axis is inverted and runs from the top of the image to the bottom' (Burger and Burge, 2005).*



**Figure 4.1: Image coordinate system** - The image coordinate system with width  $M$  and height  $N$  (Burger and Burge, 2005)

One image element can hold data according to the image format. The pixel element consists of *'binary expressions of length  $k$  and therefore it can represent  $2^k$  different values. Here  $k$  is referred to as the depth of the image.'* (Burger and Burge, 2005).

## 4. TEXTURE CONTENT MODEL

---

The type of data that is stored in a pixel depends on the image type. In this work mostly RGBA images are used, providing one channel for each of the colours red, green and blue, plus one channel for alpha. Thus, a value of a pixel in a RGBA image can therefore be described as:

$$RGBA = f(u, v)$$

In the data texture (see chapter 5.5) the channels can hold float values with 32-bit depth. This float texture is used to encode information that is later needed for the texture synthesis process implemented in a shader programme for the implemented prototype presented in the next chapter. A detailed description of the encoded information will be presented and discussed.

### 4.2 Video

A video is a sequence of digital images. *'A sequence of still pictures captured and displayed at a sufficiently high rate - typically between 24 and 60 pictures per second - can create the illusion of motion'* (Poynton, 2003)

The number of pictures per second is influenced by a number of aspects regarding the display and the environment in which it is used. Effects like flicker have an effect on the viewing experience.

*'Many displays for moving images emit light for just a fraction of the frame time: The display is black for a certain duty cycle. If the flash rate - or refresh rate - is too low, flicker is perceived. The flicker sensitivity of vision is dependent upon the viewing environment: The brighter the environment and the larger the angle subtended by the picture, the higher the flash rate must be to avoid flicker. Because picture angle influences flicker, flicker depends upon viewing distance.'* (Poynton, 2003)



#### 4. TEXTURE CONTENT MODEL

---

Therefore, *'using displays that are designed in way that the user observes very concentrated in high brightness and the image occupies a large horizontal visual angle (e.g. computer screens), the image refresh rate should be higher than 70 Hz'* (Schmidt, 2009).

For the transmission of video sequences the frames are not handled as rectangular matrices but serialized into a continuous one dimensional stream of information:

*'The brightness values that are produced by the optical transformation are spatially and temporally discretized and are available as information for the particular pixel in reduced form, but still simultaneously in vast numbers' (the pixel matrix) '... The parallel transmission of all information would be very inefficient. An essential idea is therefore to transmit the pixel information in serial rather than in parallel way. ... When the scanning of the image, the transformation and the reconstruction during the replay are fast enough, the human eye perceives a complete image, although at a discrete point in time only one pixel is transmitted'* (Schmidt, 2009).

The scanning of the pixel matrix in Poynton (2003) is generally described as:

*'Video scanning represents pixels in sequential order, so as to acquire, convey, process, or display every pixel during the fixed time interval associated with each frame. In analog video, information in the image plane is scanned left to right at a uniform rate during a fixed, short interval of time the active line time. Scanning establishes a fixed relationship between a position in the image and a time instant in the signal. Successive lines are scanned at a uniform rate from the top to the bottom of the image, so there is also a fixed relationship between vertical position and time.'*

For aspects that are investigated in this work it is sufficient to regard videos as a set of frames arranged along a time axis. Each frame can be consequently treated as a digital image where all image information is parallel present. The serial data arrangement for transmitting and visualizing video frames is not taken into consideration here.

## 4. TEXTURE CONTENT MODEL

---

Therefore, based on the definition of an image and the information that can be received from a certain element within an image the pixel information for a given frame  $t$  is:

$$RGBA_{video} = f(t, u, v)$$

Consequently, for each frame in the video sequence there is an image, which pixels can be accessed using the  $u,v$ -coordinates. In the chapter on the implemented prototype it can be seen that video frames representing real-time information are handled as textures, hence, extracted frames are regarded as images and processed in the same way.

### 4.3 The Texture

A texture is a digital image that is applied to geometry in a 3D model. Textures are normally used to increase the realistic appearance of otherwise 'flat' rendered polygonal geometries. In Heckbert (1986) this is described as:

*'one of the most frequent criticisms of early synthesized raster images was the extreme smoothness of surfaces they showed no texture, bumps, scratches, dirt or fingerprints. Realism demands complexity, or at least the appearance of complexity. Texture mapping is a relatively efficient means to create the appearance of complexity without the tedium of modelling and rendering every 3-D detail of a surface.'*

The process of applying the image to the geometry of the digital model is texture mapping (Blinn and Newell, 1976). In this process *'the source image (texture) is mapped onto a surface in 3-D object space, which is then mapped to the destination image (screen) by the viewing projection. . . . The mapping from texture space to screen space is split into two phases. . . . First is the surface parameterization that maps texture space to object space, followed by the standard modelling and viewing transformations that*

#### 4. TEXTURE CONTENT MODEL

---

*map object space to screen space, typically with a perspective projection.*' (Heckbert, 1986)

In order to draw a textured surface onto the screen, there are several possibilities to determine the particular pixel colour on screen: *'a scan in screen space, a scan in texture space, and two-pass methods. ... Screen order'* (scan in screen space), *'sometimes called inverse mapping, is the most common method. For each pixel in screen space, the pre-image of the pixel in texture space is found and this area is filtered. This method is preferable when the screen must be written sequentially.'* (Heckbert, 1986)

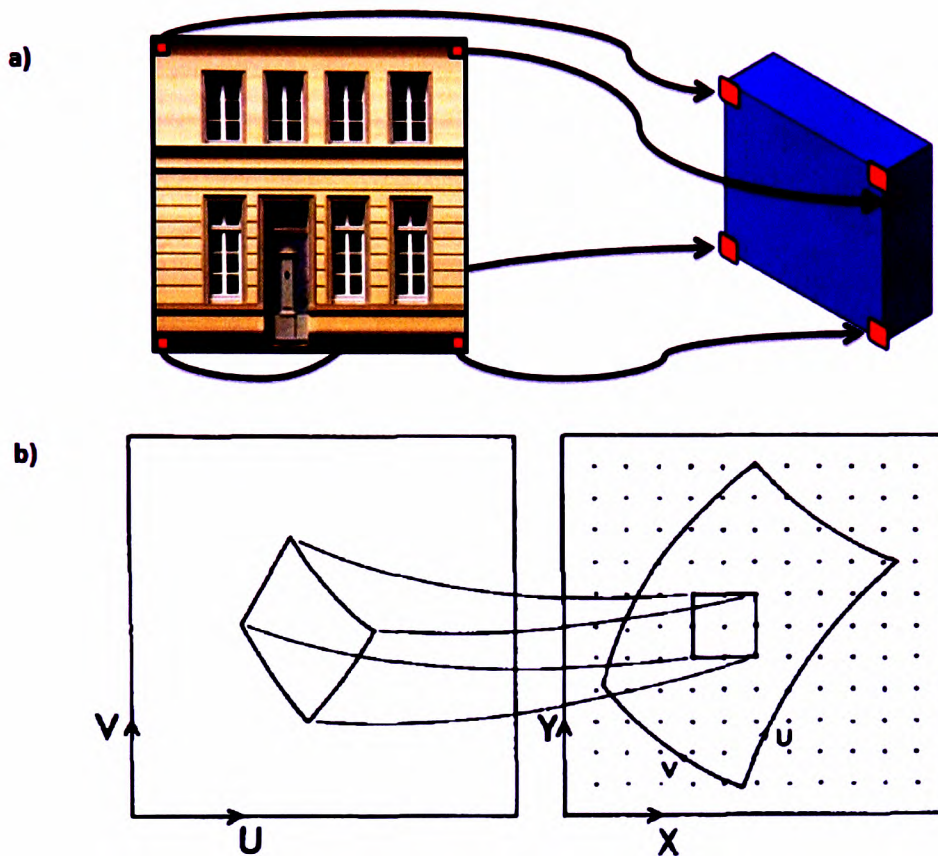
**Listings 4.1:** Pseudo code for screen scanning (Heckbert, 1986)

```
for y
  for x
    compute  $u(x,y)$  and  $v(x,y)$ 
    copy  $TEX[u,v]$  to  $SCR[x,y]$ 
```

In general the process of texture mapping can be regarded as applying a portion of an image onto the geometry of an object in a 3D scene (which is later rendered onto a 2D screen).

The texturing approach in this work, or to be more precise, the approach for texture content generation is based on an empty texture. The texture coordinates, hence, the way the empty texture is applied to geometry, are already defined. Nevertheless, the content of the texture is still undefined. The empty texture that needs to be filled with the appropriate content is called null-texture in this work. None of the pixels is filled with any values. The state of the null-texture should not be misinterpreted by a texture that is completely coloured in black (all pixel values set to 0) or a completely transparent texture that is invisible, because these two states might be intentionally chosen for specific representation purposes. The null-texture is empty or rather its content is 'none-existent' and it needs to be filled.

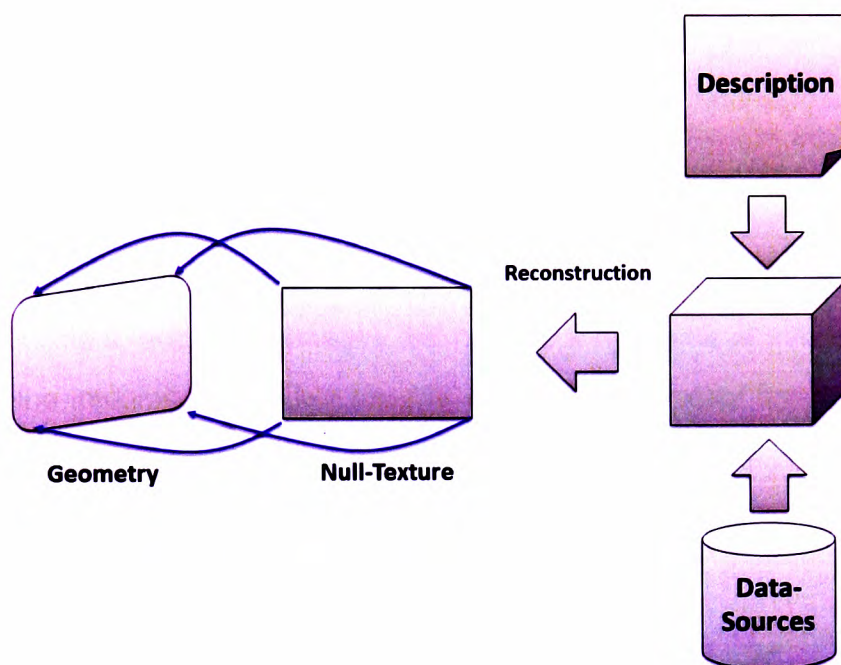
## 4. TEXTURE CONTENT MODEL



**Figure 4.2: Texture Mapping** - a) texture coordinates to assign texture corners to vertices (own depiction) b) Region of texture pattern corresponding to picture element: left-hand side shows texture; right-hand side shows image (the image is the screen content in this case). (Blinn and Newell, 1976)

The reconstruction of the façade texture, hence filling the null texture, is based on a description that integrates data from different sources into the texture and on the tiles mentioned in the introduction of this chapter. These tiles need to be managed in a texture atlas for performance reasons. Therefore the data sources for generating textures are described next, followed by the concept of the texture atlas and the new texture content model.

## 4. TEXTURE CONTENT MODEL



**Figure 4.3: Null-Texture Concept** - Null texture concept: The re-creation of the façade texture. The null-texture mapping to geometry is known, but the content needs to be generated.

### 4.4 Data Sources for Textures

Textures, respectively the images that are used as textures, can be filled by using specific inputs. In this work mainly façade textures are in focus, which can be generated, processed and applied in very different ways.

Kada et al. (2005) present an approach for texture mapping using geo-referenced real world images that are automatically orientated and mapped onto geometry. Frueh et al. (2004) present a method for texture mapping using oblique imagery for façades. And there are a lot more examples for texture generation for 3D city models. Images are relatively easy to use as sources for textures because only the mapping of the image to the geometry needs to be defined. Images need some kind of post-processing before they can be used as textures (rectification, etc.). However, the content is implicitly created through the process of photography using an image sensor (digital photography). The post-processing is dependent on the way the photos are created: oblique airborne

#### 4. TEXTURE CONTENT MODEL

---

images need a different type of treatment than terrestrial images (see aforementioned literature). Photo-Images are not limited to 'standard photography' as well. Other sensors, e.g. for thermal images, can produce digital information that can also be used as a texture.

Textures in the presented research are intended to transport other information than 'naturalistic content' and photo-images are not the only source of information. One example is the integration of video streams into a 3D city model texture. This data changes very fast and needs to be updated at a very high frequency. The concept of video that is used in this work, already explained in an earlier section, regards a video as an arbitrary number of frames (digital images) arranged on a time scale. Therefore single frames can be used as content for a texture and mapped to geometry or used to fill a part of a texture. The concept is comparable to normal images in terms of mapping. However, the time dimension requires the texture content to change at the video frame rate. These fast changes are realized in a special layer that is described in section 4.6.3.

Other types of data can be inserted into textures as well after they are converted into an image matrix. Metering data can be converted into a chart (e.g. line chart) and used as a texture. There are several chart-Tools available (e.g. Google Chart API<sup>1</sup>, JFreeChart<sup>2</sup>, etc.), which create charts for a given data set. The output of the tool is a digital image that can instantly be used as a texture for geometry. One example would be electricity consumption data that can be visualized as a line chart and applied to the according building geometry. In this way the metering information, the spatial position of the consumer and the building type can be visually integrated in the 3D city model. These are just examples for possible texture information sources and many more could be defined. By specifying the transformation process from the input data to the digital image matrix, any information could be represented through textures. This work will

---

<sup>1</sup>[code.google.com/apis/chart/](http://code.google.com/apis/chart/)

<sup>2</sup><http://www.jfree.org/jfreechart/>

## 4. TEXTURE CONTENT MODEL

---

especially investigate how to integrate different sources of information in one texture and not just use the texture 'channel' for one specific information stream (e.g. metering data → chart).

### 4.5 Texture Atlas

A texture atlas is a special structure to manage textures before they are applied to geometry. The atlas is a collection of (smaller) textures that are arranged in an optimized way, so that the atlas can hold a maximum of textures. According to Wloka (2005) the activation and deactivation of textures, the so-called texture switch, has got a high performance cost and need to be minimized in order to achieve acceptable frame rates. The solution that is presented is a texture atlas, which is also called texture page. The packing of several textures into an atlas reduces the need for texture switches; however *'models using these atlases need to remap their texture coordinates to access the relevant sub-rectangles out of the texture atlas'* (Wloka, 2005).

The generation of the texture atlas can be achieved by different algorithms and approaches. Heuristic algorithms like the one presented in Igarashi and Cosgrove (2001), which is also used by Buchholz (2006) for his multi-resolution texture atlas concept, can be used to generate texture atlases as well as data structures like Binary Space Partition Trees (BSPTrees).

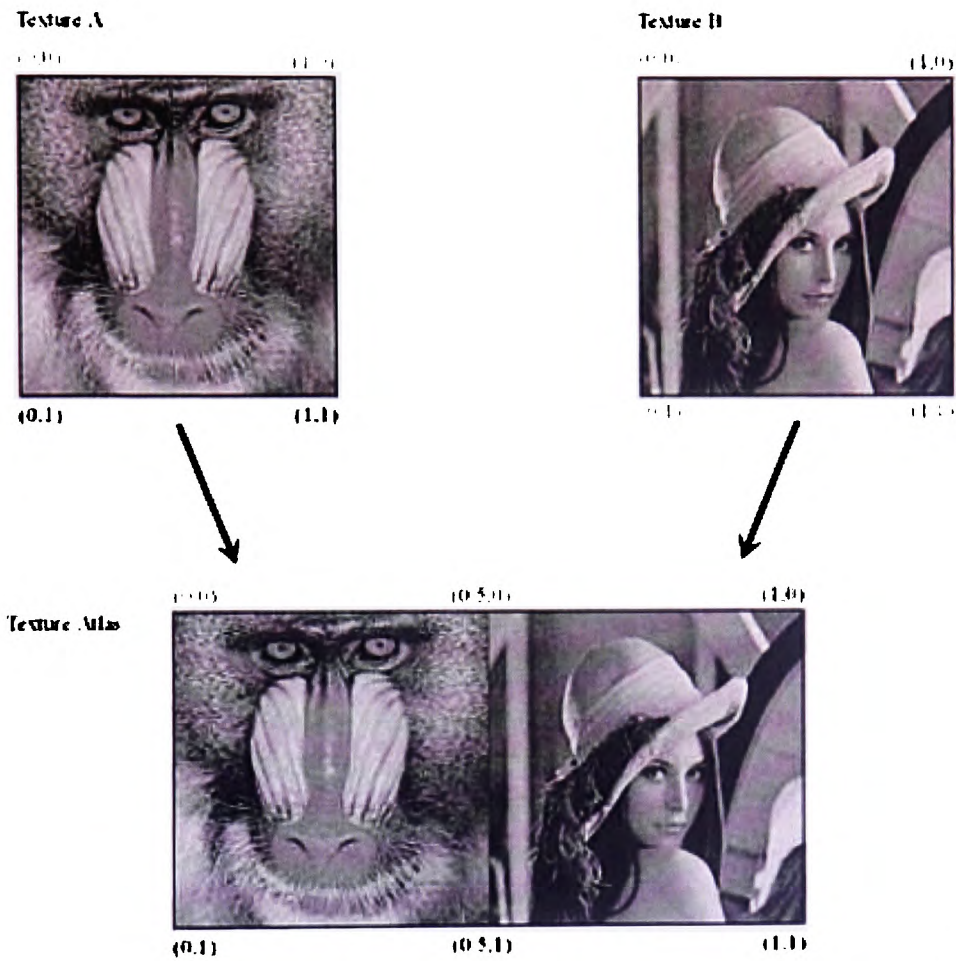
For texture atlas creation using a BSP approach the overall texture space is divided into half-spaces defining free regions where textures can still be placed inside the atlas and occupied regions where other textures were already placed. When inserting a texture into the atlas the splits of space are placed at the borders of the inserted textures in order to define empty and occupied half-spaces<sup>1</sup> (fig 4.5a). An example for a resulting texture atlas for façade tiles can be seen in fig 4.5b.

---

<sup>1</sup><http://www.blackpawn.com/texts/lightmaps/default.html> (last accessed 02. June 2011)

#### 4. TEXTURE CONTENT MODEL

---

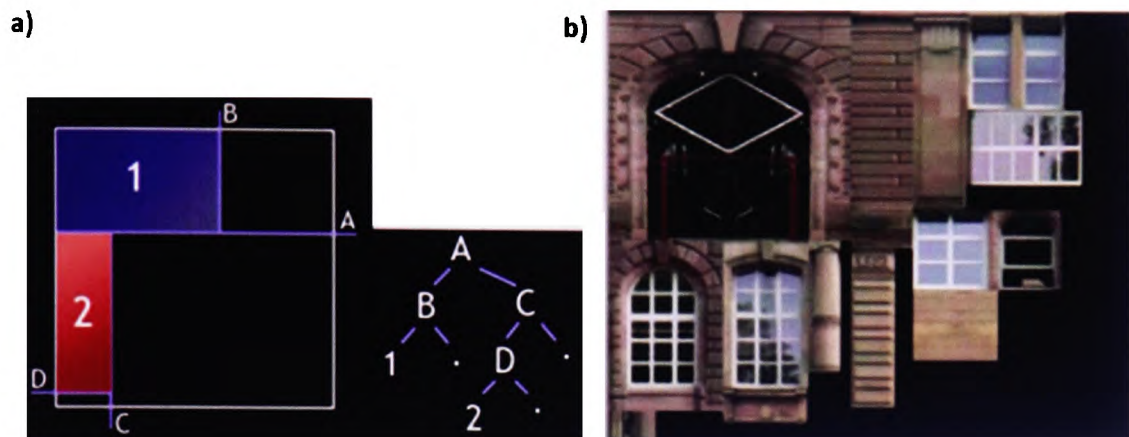


**Figure 4.4: Texture Atlas Concept** - Combining textures into an atlas. The texture coordinates for accessing data out of an atlas are adjusted according to where the original texture is in the atlas. (Wloka, 2005)

In the presented work the use of texture atlases for façade textures is necessary because the presented approach for texture synthesis is based on small texture tiles (see fig 4.5b). Therefore the number of textures for a single façade would be unacceptably high when representing each tile as a separate texture, increasing the aforementioned problem of texture switches even more. Hence, the use of texture atlases to pack the tiles for one façade into a single texture is inevitable.



## 4. TEXTURE CONTENT MODEL



**Figure 4.5: Using BSP to create texture atlas** - a) example for a BSP tree for texture atlas creation (from <http://www.blackpawn.com/texts/lightmaps/default.html>) b) Example texture atlas from the prototype implementation

## 4.6 Modelling of Null-Texture Content

### 4.6.1 Zones

The content for the null-texture is flexible and varies according to the request of the user or according to definitions of the system or the scenario. Therefore it is a result of a process that combines arbitrary content to produce the final façade representation.

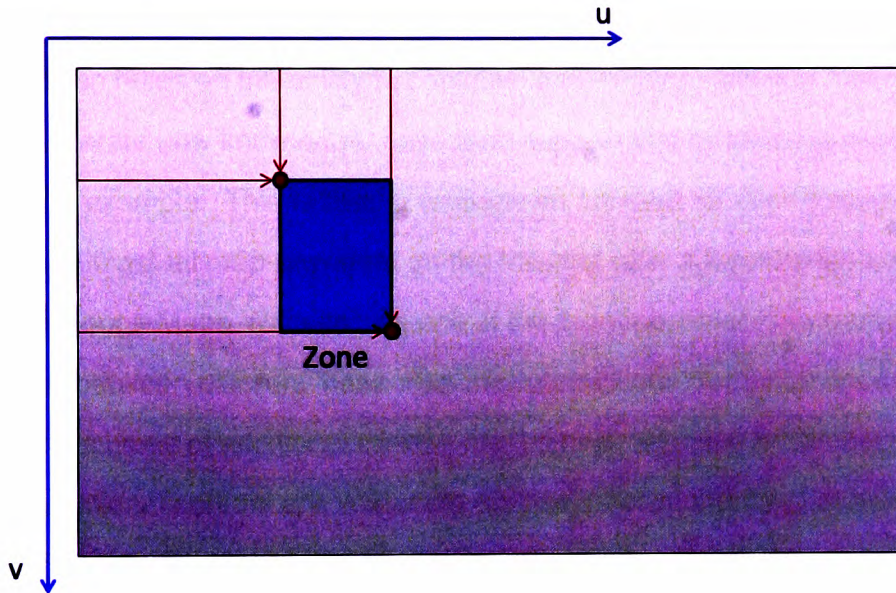
The basic element of the texture content model is a zone. A zone is a closed polygonal area of any shape. This zone defines an explicit number of pixels inside the null-texture that are about to be set by a specific procedure. The procedure can be simply a texture look-up that defines the colours of the pixels in the specified zone. However, any other algorithm can be used to produce the content for a zone, and the process is not limited to applying a texture to the zone.

In this research the investigated forms are restricted to rectangular zones. The zones are described by the coordinates of the upper-left and lower-right corners of the zone. The coordinates of the zones are defined in the texture-coordinate system of the null-texture in the range  $[0, 1]$ .

## 4. TEXTURE CONTENT MODEL

---

The zones in this concept define the spatial extent and the position of an element inside the null-texture in texture space. The content is added to the pixels in the area by the reconstruction process, which reads the information how the zone should be filled from the description. In order to manage and arrange zones with content of similar kind layers are introduced.



**Figure 4.6: Defining a Zone** - zone corners in u,v of null texture

### 4.6.2 Layer

A layer in the presented concept is a set of zones that share a common property. Normally one would include all zones representing one type of window in a layer, or all doors, all shop signs, etc. In this way content can be grouped and displayed (or omitted) together. This and similar concepts of layers are comparable to GIS, CAD and online map applications.

Using a well-known and accepted method, which the layer concept certainly is, would allow having flexible content for façades and other planar surfaces of buildings in 3D city models. Data providers as well as users are familiar with using layers. Therefore

## 4. TEXTURE CONTENT MODEL

---

this concept would support model creators to organize the acquired façade information in a meaningful way. They would be able to categorize the elements and put them into the correct layer, generating a classified set of extracted elements, which can be used very well for interactive visualization and flexible content combination. This would support approaches in knowledge construction through 3D maps and geovisualization, for example (compare chapter 2.3).

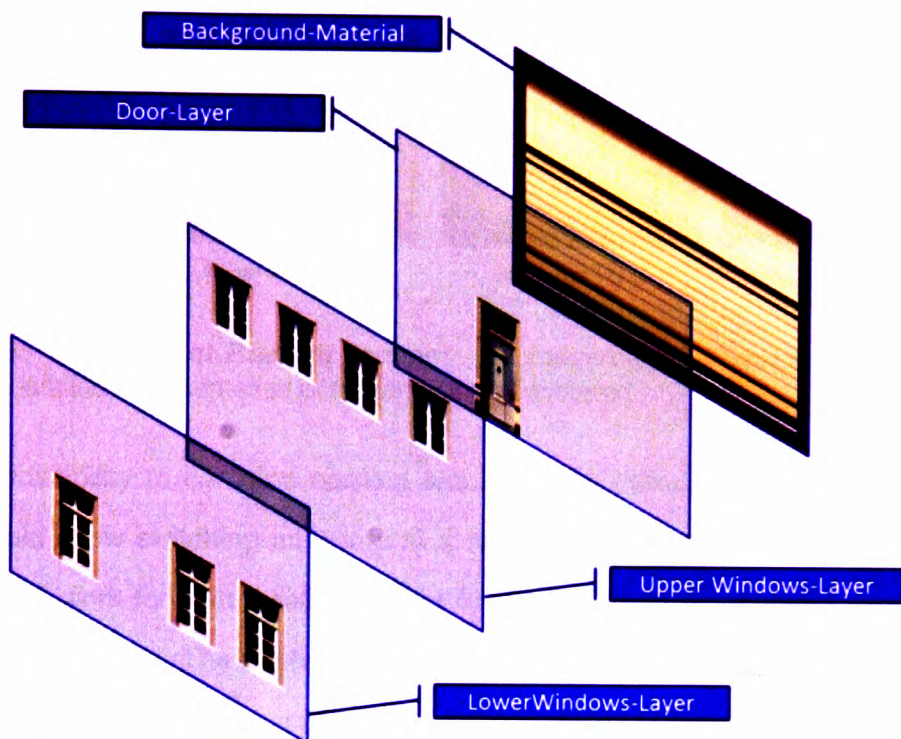
Geovisualization relies on the ability to combine specific data sets in a meaningful way in order to generate new knowledge. Layers do support this by allowing switching content on/off, for example. This is also a well-known concept for (non-) expert users of digital maps. Almost all map-providers on the Internet offer a layer based content concept. On the GoogleMaps-Website<sup>1</sup>, as one of the prominent map providers, it is possible to choose between different 'base-map' representations (terrain, map, satellite). In addition various types of additional content can be overlaid using additional layers. Examples for additional content are Wikipedia articles, photos, points-of-interest (labels), etc. Hence, to introduce a layer concept in terms of textures for façades in 3D city models seems to be promising in terms of usability and acceptance both in the expert community and for data providers as well as with less experienced users, because the basic concept is already well understood.

This work would therefore define a layer as a collection of non-overlapping zones that can be used to arrange and manage content in a meaningful way. It is important to notice that zones in one layer cannot overlap by definition (section 4.6.2.1), whereas zones of different layers can overlap. That is why operations need to be defined how to handle overlapping content of different layers when the null-texture is filled (section 4.6.2.3). Layers can be linked to a set of texture tiles (1 . . . n), which will be applied to the zones of the layer (or one colour is defined that fills all zones). There are two ways of attaching meaning to the layer in addition to the 'classification' of content due to its character (window, door, etc.): the first one is the introduction of the LoR concept

---

<sup>1</sup><http://maps.google.com> (last accessed 25.08.2011)

## 4. TEXTURE CONTENT MODEL



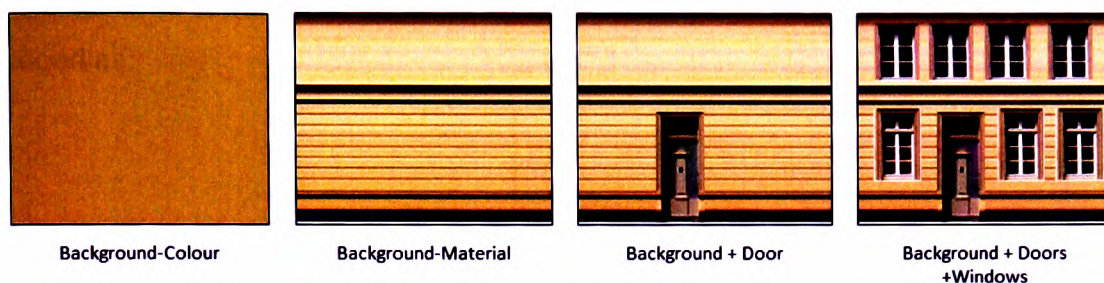
**Figure 4.7: The Layer Concept** - Content/Zones can be arranged in different layers

(see chapter 2.3.4). This attribute of layers allows defining distinct levels of realism and which type of content of the façade belongs to which level of realism. By assigning a LoR to each layer, the content 'density' of the final texture can be modified by the application quite easily. Selecting a specific LoR would switch all layers on that have a LoR value equal or smaller than the selected value. All layers holding elements of higher detail are switched off.

In figure 4.8 the combination of pulse layers forming distinct LoRs is shown. Higher LoR can also contain ledges, signs, ornaments, fire ladders, etc. LoRs cannot only be used to visualize models in lower detail for specific use-cases; moreover a low LoR is more appropriate when adding thematic content (non-realistic). The lower level of façade detail would distract the user less when observing integrated thematic information.

In this way content based control of detail can be achieved for façade textures, comparable to the LoD concept for 3D city model geometry specified in the CityGML standard (OGC, 2008a).

## 4. TEXTURE CONTENT MODEL



**Figure 4.8: The Level of Realism concept** - By assigning a LoR-value to each layer differently detailed representation of façades can be achieved

A second possibility to combine content would be the introduction of a layerSet-ID. This ID would allow switching all layers of a set on, and all other layers off. The parameter would work for 'naturalistic' content as well as 'thematic' content, and for any combination of the two. The useful aspect for data providers is the possibility to select specific content and group it to a useful pre-defined visualization, which would support specific user groups, especially non-expert users, with a standard selection of useful representations. For more experienced users, who create their own combinations of content, the layer set can store a previously defined visualization for later use or for exchanging specific visualization settings with others. A further well-known concept in regards to layers is the z-order. Layers can be placed on top of each other and specific information can be in the foreground or act as the background for other information. This capability of layer-based content fundamentally adds to the flexibility of content that wants to be achieved for façade textures.

### 4.6.2.1 Description - The Pulse Function

As we have the possibility to define the size of single zones in the null-texture as well as aggregate zones in layers, it needs to be outlined how the position of the zones is defined. In this work the concept of pulses that follow each other along the two axes of the null-texture is used.

#### 4. TEXTURE CONTENT MODEL

---

The pulse function (or rectangle function) defining one pulse is mathematically described as:

$$\Pi(x) = \begin{cases} 0, & |x| > 1/2 \\ 1/2, & |x| = 1/2 \\ 1, & |x| < 1/2 \end{cases}$$

For each axis of the null-texture (u, v) there is a pulse-function defined which can contain multiple pulses. The distance between two pulses is not unified as well as the size of the pulse. For the presented use of the pulse concept the definition of 1/2 for  $|x| = 1/2$  is also not necessary. The required function to determine if a texel of the null-texture is inside a pulse for one of the axes can be formulated by:

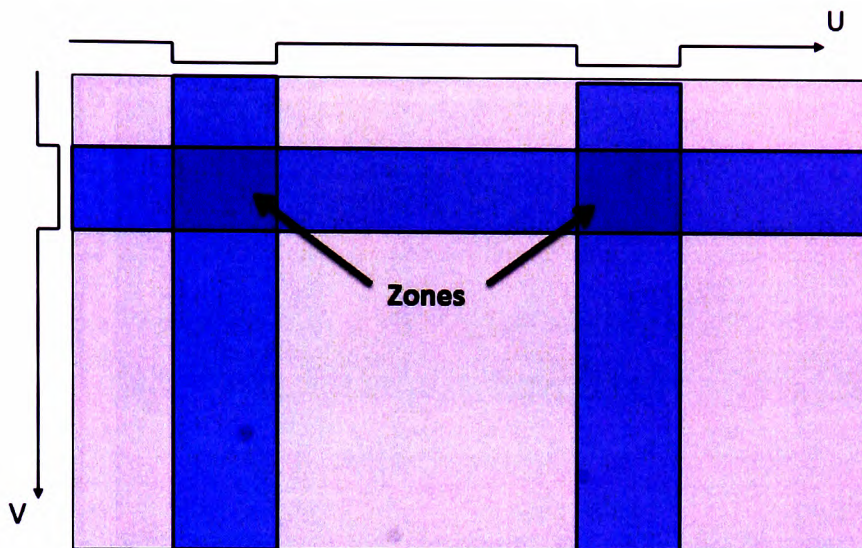
$$a_u = ([u > start_{pulse_1}] \wedge [u < stop_{pulse_1}]) \vee ([u > start_{pulse_2}] \wedge [u < stop_{pulse_2}]) \vee \dots \vee ([u > start_{pulse_N}] \wedge [u < stop_{pulse_N}])$$

*with  $start_{pulse_1} < stop_{pulse_1} < \dots < start_{pulse_N} < stop_{pulse_N}$*

A layer is defined by two pulse functions, the u-function and the v-function. By defining pulses in u- and v-direction a structure is built where zones become active when both functions equal to 1 (true). Graphically the concept can be depicted as in figure 4.9. A similar concept can also be found in Loya et al. (2008), although their work assumes periodic appearance of façade elements for each layer.

Each pulse in the u-function defines a vertical stripe over the façade area, defining a façade as a rectangle covering the area of the associated wall (even if the wall is non-rectangular, e.g. L-Form). The v-pulses define horizontal stripes accordingly. Where these stripes overlap they generate an active zone where a related action should be performed. In most cases the action is applying a texture tile, which is linked to this pulse function/layer. Nevertheless, any other action can be defined as well. The definition of action is not limited to applying a texture. It is also possible to assign a RGB colour, etc. Thinking about future developments and research it should also be investigated if any arbitrary algorithm could be associated with an active zone. This might

## 4. TEXTURE CONTENT MODEL

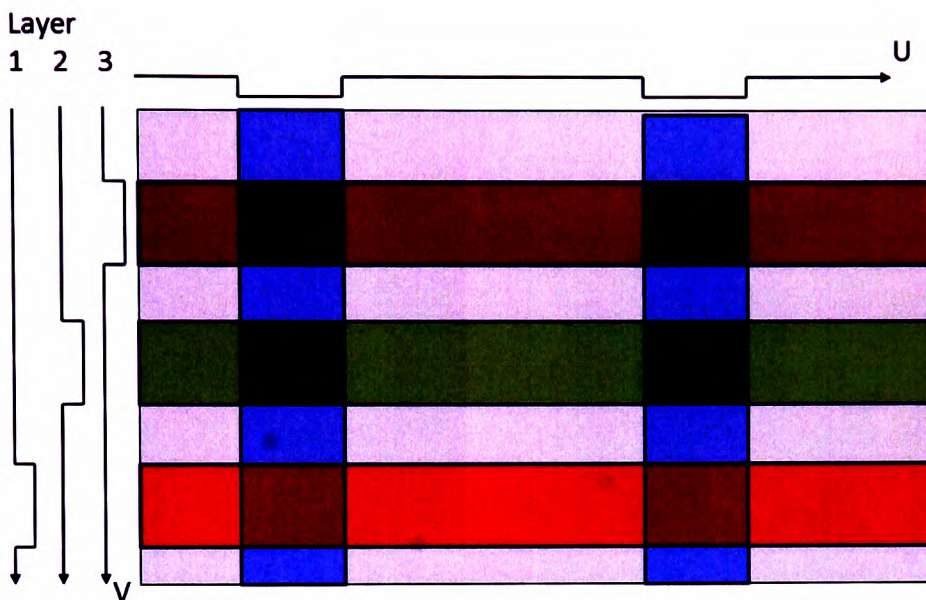


**Figure 4.9: Pulse functions overlap** - When pulses overlap they define zones for which an action should be performed

lead to scenarios where any algorithm or process can produce content for the active zone according to a set of input parameters, for example, visibility calculations, shadow calculations, heat loss etc. This information can be arranged in additional layers which would allow to limit the output of the associated algorithms to specific 'zones' and preserve the rest of the façade in a naturalistic appearance. In this research work actions are limited to texture tiles and fixed colour values for thematic colouring in order to fill zones. More complex algorithms and concepts are left aside and are considered as subject to future research.

One benefit of using pulse functions in order to describe façade structures is the reusability of functions in one direction (e.g. a u-Function) and to combine them with more than one function of the other direction (v-Functions). Because of the specific structure of façades and their nature of mainly consisting of horizontally and vertically aligned elements (Royan et al., 2007), the use of one u-function would be possible for several content layers. A layer actually holds a u- and v-function for one specific element of the façade (see section 4.6.1). The concept of reusing and referencing one function in multiple layers is shown in figure 4.10 for three types of windows for separate floors of the building.

## 4. TEXTURE CONTENT MODEL



**Figure 4.10: Multiple use of Pulses** - reuse of functions in multiple layers. The three layers might define different window types for different floors that are located directly below each other, so the u-function can be reused

On specific hardware platforms, especially mobile devices, it is important to minimize data size in terms of data transmission, memory capacity as well as storage capabilities. Modelling information only once and referencing it when needed multiple times helps to reduce data size. Here in this example in terms of the façade description. As mentioned already, this might be more important on specific platforms and neglectable on others, but the concept of pulse functions allows minimizing data size in terms of the façade description.

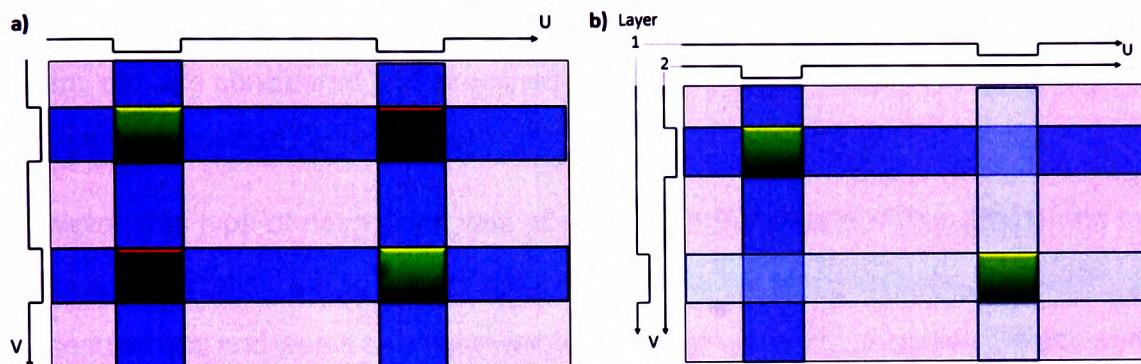
However, the pulse function approach also has disadvantages that result from the horizontal and vertical stripes running over the wall area: the stripes cannot be cut off; they run until they reach the opposite boundary of the façade area. A situation where this is a disadvantage is depicted in figure 4.11 a).

When certain elements in the façade are of the same type but do not follow the 'chessboard-pattern', the pulse functions generate elements that are not present, respectively they are created unintentionally (elements in red in fig. 4.11). As one would regard this case as very infrequent for most real world façades, the pulse function ap-



## 4. TEXTURE CONTENT MODEL

proach is still feasible. For those cases generating false active zones it is necessary to split elements up into two separate layers (figure 4.11 b)).



**Figure 4.11: False Zones** - a) green are intended zones, red are wrong zones that are falsely defined. b) Locating pulses into different layers solves this problem.

The concept of zones also allows using the 'negative' of one pulse layer, the combination of a u- and v-function combination (see next section for details), as the information for the programme that reconstructs the façade texture. By defining active zones when both functions equal '1' (true), these functions implicitly define an inactive zone where both functions equal to '0' (false). This information can be used by a programme to apply window tiles to the zones. Instead of defining further functions/layers for the (background) material of the wall, the programme could use the inactive areas to apply the wall material tiles there. This would reduce the data size of the description because two types of information are encoded into one layer.

The pulse function approach, which can also be found in Parish and Müller (2001) used for their procedural approach for generating 3D city models, is certainly not the only possible way to describe zones for synthesised texture content. A more detailed discussion on the approach and use in Parish and Müller (2001) will be provided in the discussion at the end of this chapter. The next section will discuss an alternative concept for this description.

## 4. TEXTURE CONTENT MODEL

---

### 4.6.2.2 Alternative Descriptions

The presented concept for the façade description is based on pulse functions in order to describe what is called zones in this work. Benefits like reusing functions in multiple layers, etc. are considered and examined and the prototype implementation using this approach on the GPU is presented.

However, this type of description was adopted from Parish and Müller (2001) and reflects a visualization solution. It provides benefits like the aforementioned reusability of pulse functions and works as a light-weight model for rendering purposes. As this work focuses on the use of 3D building surfaces as information space for 'map-like content' the pulse function approach works very well.

Nevertheless, in terms of data modelling and management alternative descriptions might need to be considered. To store the zones implicitly as overlaps of the two pulse functions is useful when storing content for visualization. The concept is rather compact and as the implemented prototype uses the same structure for the visualization process there is no structural conversion necessary between the data model for data management and the GPU reconstruction programme. Only the format needs to change in order to be readable by the shader programme. The concept of the description stays the same. Nevertheless, as a pre-processing step has to be performed anyway before the description can be used on the GPU it could be investigated if an alternative description concept would be more useful on the data management and storage side. When storing the zones using overlapping pulse functions these zones do not explicitly exist as own entities. This makes it rather complicated to assign attributes to them or link additional data. Texture tiles are therefore assigned to pulse layers as a sequence, meaning that the programme needs to be told to assign the tile to all zones or apply them in a certain order (first zone → first tile, second zone → second tile, etc.). Hence, there is no direct reference between the zone and the tile, as the zone is no explicit entity of the description. This is a disadvantage in terms

## 4. TEXTURE CONTENT MODEL

---

of storing/linking additional information to it. As the zones describe a feature of the façade, e.g. a window, one would want to store more information about it. When the 3D model, respectively the façade description, is not only used for flexible visualization but also for extracting information about the façade as input for other processes (e.g. window/wall-ratio), the additional information about the façade features is important. To be able to link attributes to the zones becomes fundamental.

For example, when estimating heating energy demands for buildings in a city the window/wall-ratio of the buildings' façades needs to be known. The extracted window information from a photo-image could be transformed into the presented model and the ratio can be calculated. In this way the description of façade elements would not only be used as visualization information but to hold data about the elements in the façade, e.g. if these elements are not modelled in the geometry of the building object. Additional attributes for these elements like glazing type, window type, heat transfer coefficient, etc. can only be assigned to all elements in the entire layer as the single elements in the pulse function concept are not explicitly modelled. It might therefore be more useful to store the elements with their rectangular bounding box as specific entities and translate them into pulse function representation in a pre-processing step. In that way the façade description can also be used for other use-cases than visualization. This example shows that façade descriptions are not only important for visualization purposes and it needs to be investigated how they can be used for a variety of scenarios.

### 4.6.2.3 Operations

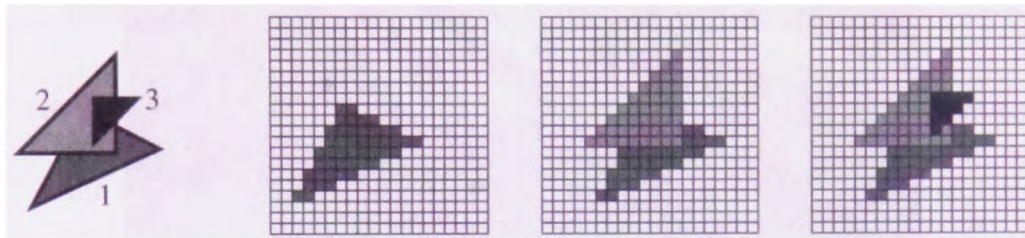
As layers are arranged to cover the whole of a given façade and in a specific z-order the content will certainly overlap in certain situations. Here the question is how the construction process for filling the null-texture should apply the content and handle overlapping areas.

## 4. TEXTURE CONTENT MODEL

---

Some examples for possible layer operations are given in the following list:

- **Painter's algorithm (Priority fill)** - As the layers are arranged in a specific z-order the 'painter's algorithm' (see de Berg et al. (2000)) can be used to subsequently draw the layers to the null-texture. Starting with the layer farthest away and then drawing subsequent layers over them (fig.4.12).



**Figure 4.12: Painter's Algorithm** - The concept of the Painter's Algorithm (from de Berg et al. (2000))

Another way to implement the effect of painting over the existing content is to find the closest layer for which the specific pixel is inside a zone and apply the colour to this pixel. For this solution the layers are arranged in opposite order (the nearest is first in the list). The current pixel that is processed is checked against the zones of the layers. If it is inside a zone the corresponding action is conducted and a colour is assigned to that pixel. Subsequent layers are not checked anymore, because if the pixel is inside another zone the colour value must not change as the other layer is behind the first one.

- **Blending/Transparency** - another possibility is to define a transparency value for certain layers, so that the content behind can shine through the layer on top.
- **Erase zone** - if the zone of one layer overlaps with a zone in another layer an algorithm could be used to erase zones according to predefined rules or a specific parameter. One parameter for example could be the z-order or depth to erase the zone of the layer in the background or the zone of the foreground layer. A priority attribute that can be freely set could change this behaviour and a zone is erased according to its significance (e.g. a door will always erase a window). This type

## 4. TEXTURE CONTENT MODEL

---

of layer operation gives a lot of control about the content as the interaction of semantic elements can be influenced by the data provider or the user.

These are just a few examples for operations and there are certainly others to be added in specific scenarios or use-cases. The examples for different operations which are combining layers are another aspect of adaptability and flexibility of the presented texturing approach.

### 4.6.3 The Real-Time Layer

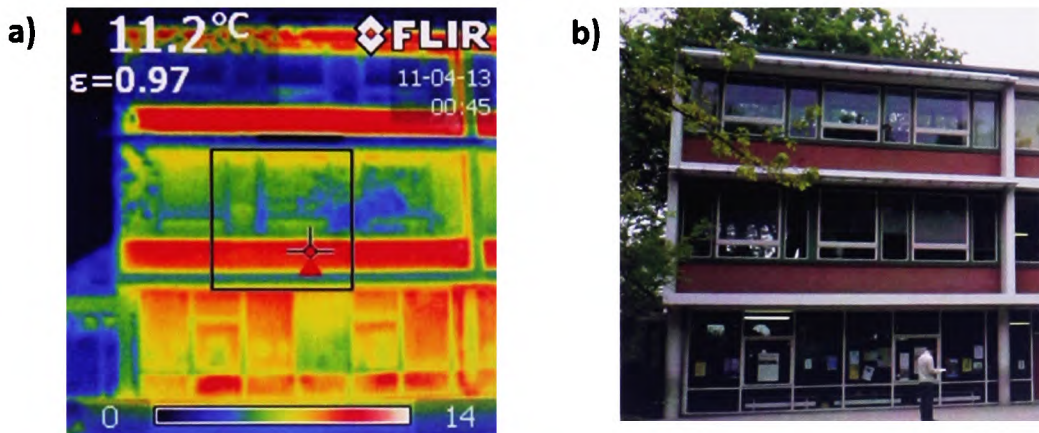
When looking at the presented concept of using pulse functions and the definition of zones that are rectangular in this approach, the concept is focusing on the horizontally and vertically arranged structures in the majority of building façades. Doors, windows, signs, etc. can be modelled as rectangular areas in a façade that normally do not overlap. Here the use of pulse functions can be regarded as a useful description for such structures, as presented earlier in this chapter (reuse of pulse functions).

However, there is also information that one would not want to subdivide into rectangular zones and add it as a layer to the façade description. This information does actually not include features that can be extracted and described as rectangular texture tiles. Data like a thermal image, a noise pollution map of the façade, etc. includes information in the form of a digital image that needs to be integrated as a whole and not as subdivided zones. The thermal image, for example, cannot be tiled by extracting façade elements. It rather contains continuous data about a surface property, stored in a regular 2D matrix. This type of data will be called 'thematic image data' in the remainder of the thesis. Another issue would be the integration of a video stream, which also includes continuous information that can not necessarily be subdivided into zones in order to be represented by texture tiles, as tiles normally contain the visual information of a specific façade element. What is even more important in case of a video is the fact that

#### 4. TEXTURE CONTENT MODEL

---

the content changes very fast and the update of a tiled video image would probably take too long to serve video frames at an acceptable rate.



**Figure 4.13: Thermal Image Content vs Façade Structure** - a) The continuous thermal image information has the character of an evenly distributed surface property in contrast to b) identifiable objects of a façade structure (copyright HFT Stuttgart, project Enff:Stadt Ludwigsburg)

In order to integrate information like videos and thermal images the layer-based texture approach needs to be extended and slightly modified. There are actually two ways of integrating the aforementioned types of information that cannot be tiled properly. The first option is to use the existing structures that the concept provides and use them as a frame for the 'thematic image data'. Hence, it is possible to create another pulse layer and name it appropriately as 'thermal image'. The pulse function would define a single pulse for the entire image. Normally the whole wall area would be covered and the thermal image would act as the texture tile that is assigned to the resulting zone. By using the z-order of the layer other content can be placed in front of the thermal image or behind it. This concept would work when the 'thermal information' is not changing frequently, although it would misuse the zone concept. The thermal image would be integrated into the texture atlas holding the other façade tiles, which would make the use of the atlas less efficient. If the information would be a video captured by a thermal camera the content would need to change very fast. Using the existing

## 4. TEXTURE CONTENT MODEL

---

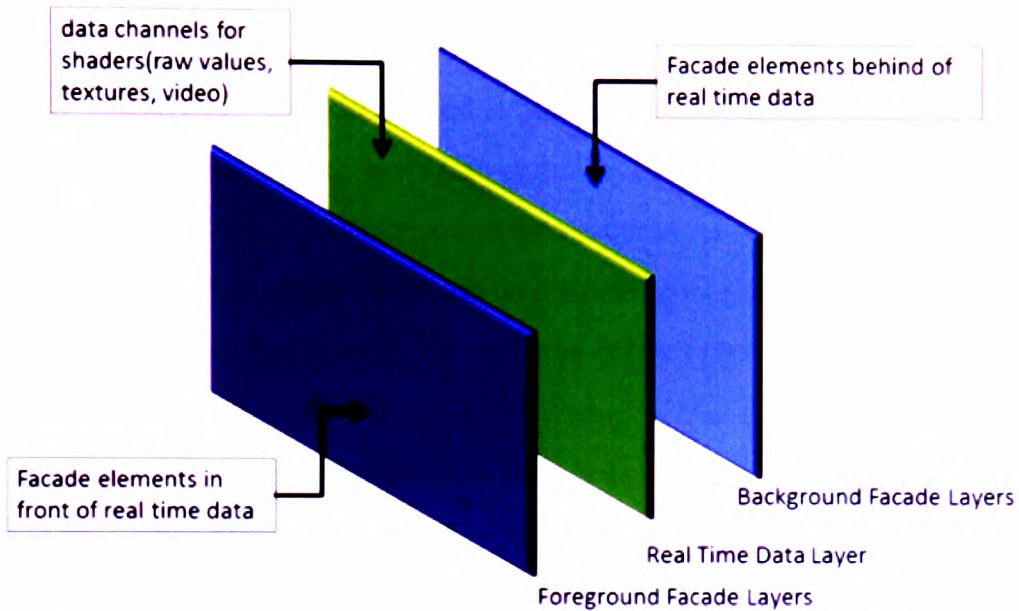
approach as it is described in the previous sections each content change would require the reconstruction of the entire façade texture at the same rate of the video stream (besides the fast changes of the video frames the concept of packing the tiles into a texture atlas can hardly be integrated with video content). The reconstruction process would need to be started even though the façade content (the selected elements that should be in the façade texture) might stay unchanged and only the video content is going to change. For this kind of (near) real-time data there needs to be a different concept, which is going to be presented in this section.

### 4.6.3.1 Real-Time Content Integration

Real-time content in terms of information like noise, thermal data, videos, etc. is normally represented as a sequence of images (compare next section on mapping information), where images and content change very fast. In terms of a video the single images change so fast that the human eye cannot distinguish single frames and perceives a continuous motion. As videos are integrated into 3D scenes as textures, the content of the texture needs to be loaded and changed very fast. As mentioned already, doing this by using an additional texture layer and re-filling the null-texture at each frame of the video is an overhead as windows, doors and other content might remain unchanged for the majority of video frames. Besides this aspect the content of the video cannot be integrated into the texture atlas for a façade and would need to be handled in parallel, which makes the zone/layer-concept inconsistent. For describing the alternative concept using a separate layer the case of a video will be used although the content that can be integrated is not limited to a video/movie.

As normal layers cannot be used to integrate real-time video frames an additional fixed layer is included into the façade texture content. This fixed layer can handle real-time 2D image data and is called 'real-time layer' (fig 4.14).

## 4. TEXTURE CONTENT MODEL



**Figure 4.14: Real-Time Layer Concept** - The position of the real-time layer

The real-time layer is based between two 'containers' for pulse layers: the foreground and the background container. Basically there are two scenarios existing for the texture synthesis process. In the first scenario there is no real-time content present for the particular façade texture, therefore there is no real-time layer. In this case there is only one null-texture that will be filled with pulse layers. The concept of filling the null-texture is described in previous sections. In the second scenario the real-time content is present and therefore there is a real-time layer and an additional background null-texture.

As we learned layers do have a z-order or depth. The z-level parameter makes it possible to position content in front or behind other content. By allowing negative values for the z-order attribute it is possible to put content behind the real-time layer, into the background container, and content with positive z-level in front of the real-time layer (foreground container). Consequently, the real-time layer has the z-level zero. By filling the foreground and the background containers two separate null-textures are filled and there is not only one texture for each façade, instead there are 1) a foreground texture, 2) a real-time texture (with content of the real-time layer) and 3) a background



## 4. TEXTURE CONTENT MODEL

---

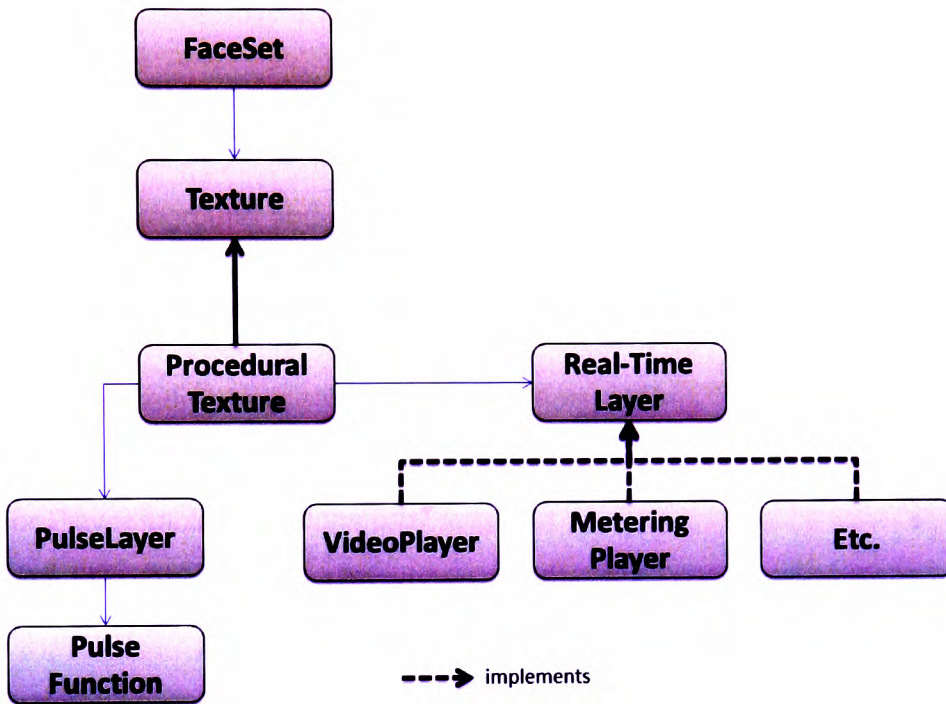
texture. These textures need to be combined during the rendering process by suitable multi-texturing concept. A possible solution would be to paint the textures on top of each other starting with the background texture, and then paint the real-time layer. In case the real-time layer only covers a part of the overall façade the rest of the real-time texture needs to be transparent. On top of the real-time layer the foreground texture is painted. For the foreground all areas that do not belong to a zone need to be transparent. By using separate null-textures the real-time content can change independently of the façade content. New video frames can be loaded without changing the front and the back texture. Furthermore, the video can be put further into the fore- or background, although the real-time layer is set to be at the z-level zero. By transferring more layers into the foreground null-texture the video relatively moves more into the background and vice versa when transferring more layers into the background null-texture.

Content for the real-time layer is not limited to video frames. Any data that can be represented as a texture/image can actually be transferred into the real-time layer. For example, real time electricity consumption metering data can be used to generate a line chart of the last 20 measured values that is provided as an image. For each new measurement (e.g. 10 seconds interval) a new chart image is created and replaces the old chart in the real-time layer. In that way the real-time metering data is updated independently from the façade content although it is integrated in the façade representation.

### 4.6.3.2 The Player Components

In order to support different real-time content inside the flexible façade texture representation an interface needs to be defined that describes the connection point for this content. In the presented concept this interface is the 'RealTimeLayer' which is implemented by the 'Player' components (fig. 4.15).

#### 4. TEXTURE CONTENT MODEL



**Figure 4.15: Real-Time Layer Implementation Concept** - The Player Components that act as the Real-Time Layer content providers

A player provides an image (a frame) that can be used as input for the real-time layer. The interface also provides functions to check if a new frame was created by the Player and is ready to replace the current content of the real-time layer. The rendering process can check in each display-cycle if the player provides new content and if it needs to load the new frame in order to render updated real-time information. Players that implement the Real-time layer interface would normally run in a separate thread<sup>1</sup> and implement the logic of transforming external real-time data streams into a 2D image matrix. Examples will be provided in the next chapter presenting the prototype implementation for the façade texture synthesis.

<sup>1</sup>a thread is a piece of code or a function that can be executed parallel to the main process of a computer programme

## 4. TEXTURE CONTENT MODEL

---

### 4.6.3.3 Summary

The concept for real-time data integration is a straight forward extension of the presented layered texture synthesis approach for façades in 3D city models. It uses two null-textures for foreground and background layers and embeds the real-time content in the middle. The z-order parameter is used to define which layer is part of the background and which are applied in front of the real-time content. By providing separate 'containers' for façade representation and real-time content these two types of information can change independently, which also enables a better performance. Two examples for Player implementations providing real-time content will be described in the next chapter about the prototype implementation.

### 4.6.4 Mapping of Information to Zones/Real-time layer

After defining the zones, their position and extent, by the pulse functions and the combination of two functions to form a layer, the next question is how to fill the zones with content. The information that is needed to fill the zones of a layer can come from different sources (see section 4.4) and needs to be translated into a texture. In terms of zones that represent real world façade elements like windows, etc. the information is represented by texture tiles.

#### 4.6.4.1 Image to Zone

Appropriate content holding the visual information for the zones needs to be provided. This content can be delivered in form of texture tiles, small rectangular pieces of the overall façade typically holding the visual information for one façade element. Tiles can be cut from the real-world photo image or gathered in any other way. It would also be possible, for example, to read tiles from a database of standard elements in order to fill specific zones.

#### 4. TEXTURE CONTENT MODEL

---

The information that is normally used is the naturalistic representation of the real world element, e.g. a window. An element can be regarded as 'atomic' or 'not dividable into smaller elements'. These tiles are pieces of the real world façade that are meant to be applied to the zones. The tiles deliver the visual content for the semantic structure. When a layer has two pulse functions describing the positions of the windows in a façade, for example, the texture tiles would actually hold the visual characteristics of this type of window. Another layer could also form a window layer describing the position for a second type of window, with a different texture tile associated to it, forming the information for this second type. However, for one specific pulse layer the texture tile concept also has an advantage in terms of recreating a photo-realistic impression, if this is the aim of the visualization. Using one specific tile for all windows of one type, for example, the resulting 'photo-realistic/naturalistic' façade might look too synthetic, because all elements of one type would look exactly identical. By defining not only one specific tile for one layer, but a sequence of tiles, variations of the same 'window' can be applied to the zones of a layer.

The texture tile can be applied to a zone in different ways, similar to textures that are mapped to geometry. As a tile is not necessarily the same size as the zone there are several possibilities to handle this problem:

- **Scale** - the texture is scaled to the extent of the zone. If the texture is smaller than the zone it is stretched to fit the zone. If the texture is larger it is scaled down.
- **Repeat** the tile is repeated until the end of the zone. This can be controlled separately for each axis. It is possible to define a repeat only on one axis. In this case the repeat is performed only on the specified axis and a scale on the other.

The mapping of image information to a zone can be regarded as a painter process, taking into account the settings for scale/repeat, filling the zone with the associated texture tile information.

## 4. TEXTURE CONTENT MODEL

---

### 4.6.4.2 Measurement Set / Attributes to Zone

Attributes are often represented by false-colours in 3D city models. Value ranges of attributes are assigned to a specific colour scale and geometry is coloured accordingly. In this way it is possible to represent numerical, attributive data in a 3D city model visually. The zone in this case is just filled with the corresponding colour; hence the same colour is assigned to all texels inside the zone. There are ways to represent single building attributes (see chapter 2), but it would also be possible to integrate much more information. When considering a system controlling the blinds of windows, the 3D city model could use that information and integrate it into the visualization. The window zones could be coloured green when the blinds are open and red when they are closed. When the frequency of changes is low enough this information can be represented through the zone colour. However, if this information is likely to change very often it might be considered to integrate this information into the 'real-time layer', because the change in the zones would require a reconstruction of the entire façade texture.

### 4.6.4.3 Real-Time Content to Real-time Layer

#### Video

As described in the section on the real-time layer the concept of the additional 'channel' is to integrate real-time information in form of an additional 'texture'. Hence, real time information needs to be transformed into a digital image that can be used to fill this real-time layer. For videos this is a quite simple step. By using appropriate software tools to access each frame of the video separately it is possible to extract single frames as images. Each frame-image is loaded into the 'real-time' texture. This process needs to be realized in parallel to the normal rendering process, suitably realized in a separate thread of the application. When a new frame is extracted from the video

## 4. TEXTURE CONTENT MODEL

---

the 3D visualization application needs to load it into the assigned texture object for the real-time layer.

### **Metering**

For other data that is not image based the process of generating the required (near) real-time texture information is a little bit more costly. Information like metering data, for example, needs to be transformed into a texture, respectively an image. One possibility would be to define a colour scale (green → orange → red) in order to visualize the currently measured value and the colour would change according to increasing or decreasing measurements. Another possibility would be to actually generate an image that contains the current value as text. In that way it would be possible to represent the value as such.

A further approach, which is also implemented in the prototype, is to convert the measured data into a chart that illustrates measured values over time. There are a number of libraries for different programming languages and systems to generate charts for a given data set. The result of the process is a digital image containing the requested chart type representing the provided data set. This digital image can be used as a texture for the real-time layer. Changes in the data set would result in re-rendering of the chart image, which would then replace the texture of the real-time layer.

### **4.7 The Reconstruction Implementation**

The reconstruction stands for any piece of software or implementation of an algorithm that actually performs the construction of the final façade texture. The input for the programme is the description providing the structure and semantics of the façade representation and the texture tiles, which provide the visual content. Programmes would normally differ depending on the platform they are implemented for, because they include the logic of how to translate the information of the description and the tiles into

#### 4. TEXTURE CONTENT MODEL

---

the most appropriate form to be utilized on the specific platform. As it will be described in chapters 5.5 and 5.7, the description of the façade can actually include platform-dependent additional information, which the associate programme can use to perform its task in an efficient way. The generation of this platform-specific information can be part of the actual reconstruction process of the final texture, an initialization step or can be implemented as an independent pre-processing step. The latter case will be described in the chapter on the rendering process of the implemented prototype. The prototype implementation takes a pre-processing step in order to generate a platform-dependent description containing additional information extracted from the general façade structure description. Besides the fact that the programme for the reconstruction process can be implemented platform independent on various systems using different programming languages and on different devices, it is also true that the pre-processing step and the actual programme might need to be realized on different platforms using different concepts and hardware. This is especially the case when working in a client-server environment, e.g. a web-application or with a mobile device. In such a scenario it is quite likely that the server and the client system are different. The server might also need to serve multiple clients of different type. It is possible that the server needs to implement multiple pre-processing steps for each specific client in order to support each client in an optimal way. This and other issues on SDIs will be discussed in detail in the next section of this chapter, looking at client-server scenarios in detail.

Nevertheless, also on one single device the pre-processing step and the actual reconstruction process might need to be implemented based on different technologies due to the underlying system architecture. In the next chapter on the implemented prototype, the reconstruction is done on the GPU of the system, using the programmable parts of the rendering pipeline. In order to achieve this goal a Java based prototype was implemented. This 3D viewer application reads the 'general' façade description (see section description) and transforms it into a texture (see section data texture). This tex-

## 4. TEXTURE CONTENT MODEL

---

ture is loaded onto the graphics chip using the OpenGL-API (Application Programming Interface), where the programme implementation uses it to perform the reconstruction of the final façade texture. In this case the texture generation procedure includes two steps using different technologies on the same platform. However, having the ability to split the two steps and perform them on different platforms and distributed systems is especially beneficial in client-server environments (see chapter 4.8). Therefore the presented approach for synthesized façade textures is not only variable in terms of content and visualization; it is also adaptable to in terms of implementing it in different system scenarios.

### 4.8 Integration into Spatial Data Infrastructures

In chapter 2 it is outlined that type-6 city models very often exist in an SDI-environment in order to be used in different scenarios and for different purposes. They are often managed in GIS-like systems and are provided to various client systems via standardized open interfaces, e.g. from the OGC. In chapter 2.2 WFS and W3DS example workflows in a 3D management system were described briefly. These services are the relevant OGC interfaces for the scenarios in this chapter as well. The WFS provides access to the raw model data, which can be further processed by the client, whereas the W3DS provides scenegraph structures that are used for 3D visualization (see OGC (2010*b*) and OGC (2010*a*)).

This part of the chapter investigates how the presented texture content model can be integrated into the SDI context. As the model itself provides a considerable degree of flexibility the integration can be done in various ways. Therefore, three example scenarios are investigated in the following section, to show approaches and possible benefits of layer/zone-based textures in an SDI-environment.



## 4. TEXTURE CONTENT MODEL

---

### 4.8.1 Flexible Integration into Client-Server Process Chains

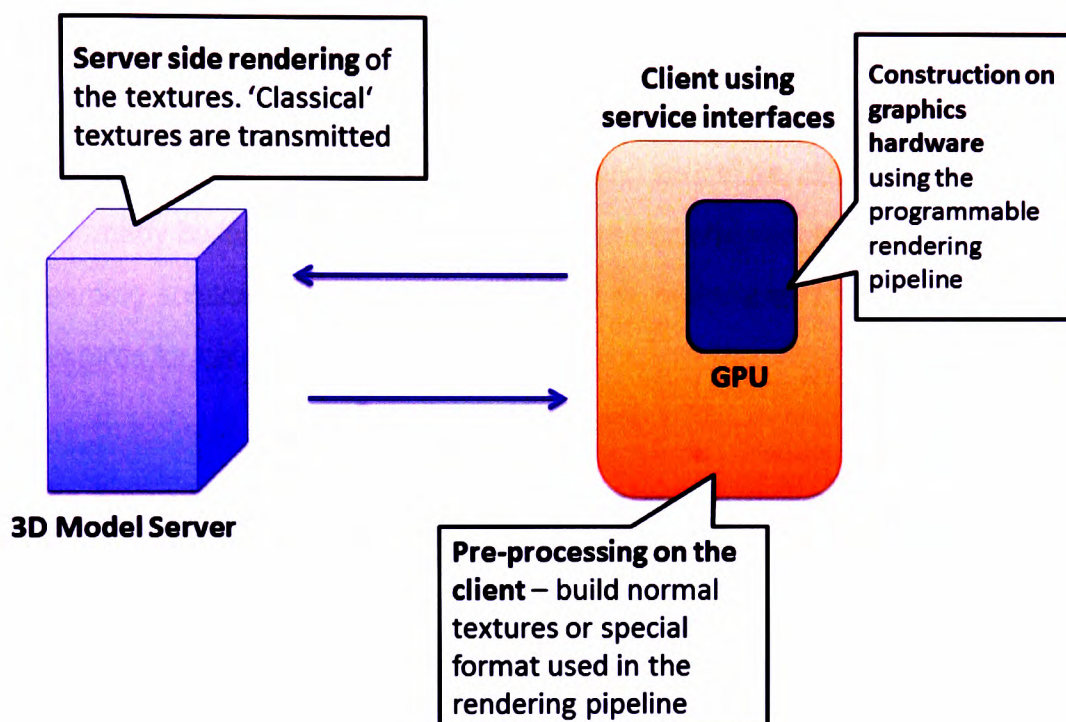
The flexibility of services in an SDI depends on their capability of providing data in just the right way for specific scenarios and client requirements (see section 2.2). The integration of the flexible texture content into the various processes, which depends on the client capabilities, scenario requirements and on the user request, can be realized in various ways and on very different levels of the server-client process chain. Depending on the intended process the client can request data in order to process it itself or delegate the majority of tasks to the server side. This would require that the related functionalities are present on the server. For example, the server needs certain basic capabilities to produce pre-processed output, e.g. generate a scenegraph from raw model data when it wants to support the W3DS interface. This task involves applying a certain user requested style to the model before the output is generated. This capability is necessary to support the defined parameters in the OGC standard.

In terms of layer/zone-based texture content there are several scenarios that need to be supported. Three of them are discussed here as examples for integrating the flexible texture content concept, which this work is investigating. There are several points for the texture content to be created from the zone/layer description. At which point the reconstruction takes place depends on the scenario requirements but for the main part it depends on client capabilities and how the zone/layer information can be handled by the client. Different scenarios regarding these requirements are presented in the next three sections. The different levels at which the façade representation can be generated in a client-server scenario are depicted in figure 4.16.

### 4.8.2 Scenario 1: Server Side Texture reconstruction

The client in this scenario can only perform the traditional texture mapping and is too 'thin' to handle the construction process in addition to its general tasks. However, the scenario requires different representations for building façades, which should be able

## 4. TEXTURE CONTENT MODEL



**Figure 4.16: Possible reconstruction levels for texture content** - The possible stages of the client-server environment, which can be used to construct the texture content

to be requested from the server according to the user input (user can switch layers on/off). Here the construction of the texture would be done on server side, querying the façade layers that are needed for the requested representation from the database and the final façade texture would be generated by the management system. The model can then be sent to the client in the usual visualization format. The rendering and texture mapping is done in the normal way. In this case there is also no need to send tiles packed into a texture atlas, as the tiles are processed on the server side and the final texture is generated before the transmission to the client. Of course, the server interfaces have to support ways to influence the output façade content by providing appropriate parameter sets, e.g. add additional parameters to the W3DS interface.

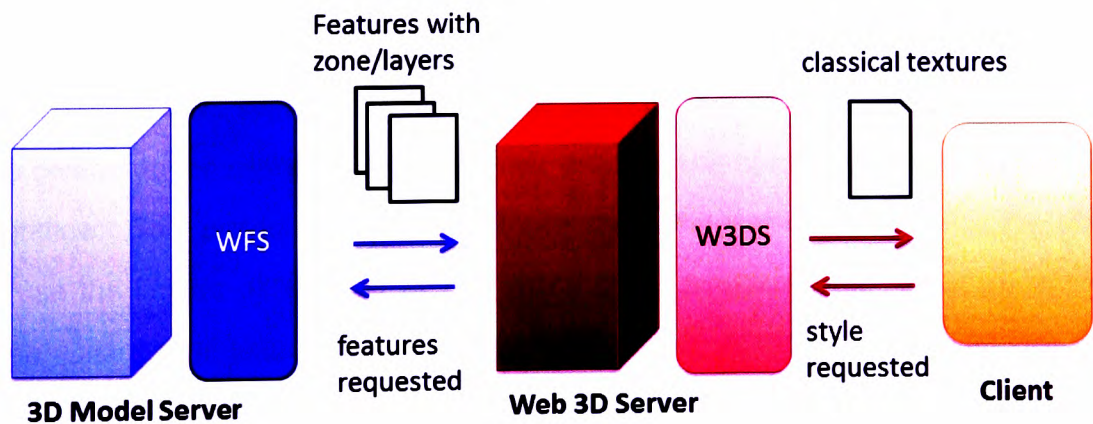
The benefit of this solution for thin clients is that they can still benefit from the layer-based texture concept in terms of flexible appearance of the 3D model although they

## 4. TEXTURE CONTENT MODEL

cannot handle the new texture concept themselves. However, a drawback is, when content changes are required, it is necessary to request the new representation from the server, because the 'raw' texture content information is not present on the client. On the server new textures are pre-generated and sent to the client. If this needs to be done for many buildings the response times can become very long, unless some kind of streaming solution or transmission strategy is implemented for this scenario (e.g. new textures for nearby objects first, etc.).

### 4.8.3 Scenario 2: Client Side Reconstruction and Processing

In this scenario the client's performance is high enough to handle the construction process of the texture content. The client in this case is not necessarily an end-user application (for end-user example see chapter 6.3.3). It can also be another service. Here at this point the use-case includes an external W3DS that queries the 3D model through an WFS interface and produces an appropriate scenegraph for its own client.



**Figure 4.17: Portrayal W3DS** - Portrayal W3DS querying features with layer/zone based texture content and creating classical textures for its client

In this scenario the client of the 3D management system is another service that intends to receive the original model information in order to process it in the appropriate way.

#### 4. TEXTURE CONTENT MODEL

---

Here the zones, layers and tiles are provided in their original form, e.g. by a CityGML ADE (Application Domain Extension). The information about the façade content can be used by the client (in this case the external W3DS) to generate the required façade representation according to the request the W3DS received. For the 3D management system the task in this scenario does not involve any pre-processing of the texture content, it just needs to generate the requested data format encoding and deliver the information.

In the discussed scenario the WFS of the management system provides the genuine model data to a W3DS in order to be processed further. However, when implementing the WFS it is unknown if a client already supports the layer/zone-based façade content. Furthermore the new texture content model allows that building façades within one model can either have normal textures or layer based texture content (the layer-based texture can be implemented as a sub-form of a standard texture, see chapter 5.3). Therefore the request to the WFS needs appropriate filters to tell the system if the output should contain layer based texture content or only normal image textures. In this way also W3DS services and other clients that are 'not aware' of the layer based texture concept can still use the WFS in the 'classical' way.

The management system in this scenario provides the genuine layer description of the content to the client, which can further process and use the information in its own workflow. This provides more control over the content and also enables the client to use the façade description for non-visualization purposes, e.g. calculate window area. Changes of texture content are easier to realize as the description and the tiles are already on the client side. The reconstruction of texture content does not include reloading information from the server. However, the client needs to have the required capabilities and needs to be 'aware' of the concept in order to handle the texture information. By using appropriate filters it might also be possible to query specific texture layers through the WFS interface. In that way only specific layers are transmitted that are relevant for the client's scenario and not the full set of texture content has to be

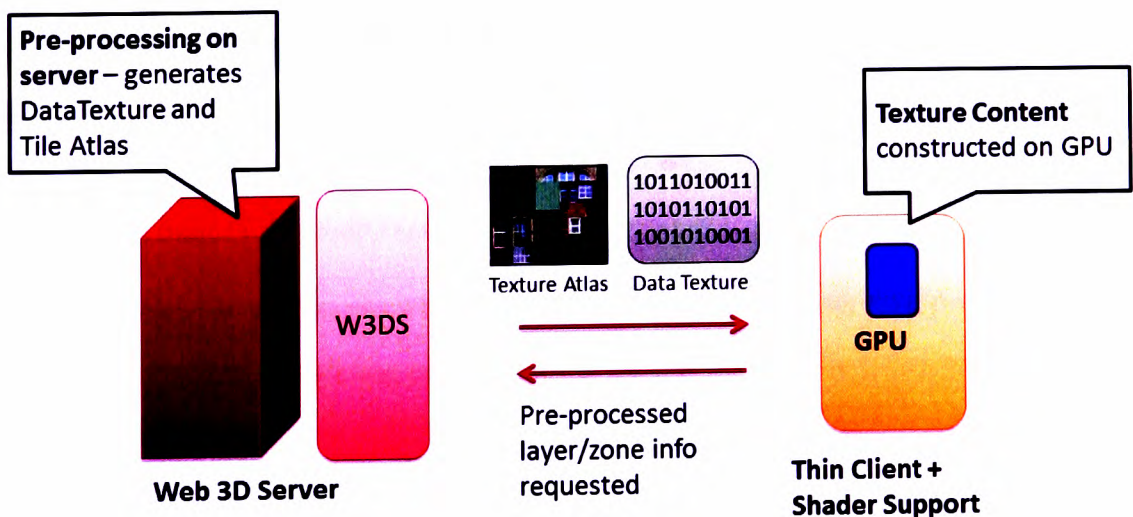
## 4. TEXTURE CONTENT MODEL

---

exchanged. However, this possibility using the WFS interface and the relevant filter definitions needs to be investigated further, as it has not been tested in the scope of this work.

### 4.8.4 Scenario 3: Pre-processing for Clients with Special Requirements

In this scenario the 'raw' texture layer and tile/zone information is pre-processed but no final façade texture is generated. The description of the texture content is just transferred into a suitable form so it can be better used by specialized clients. One example for such a client is the prototype implemented for this work presented in chapter 5. It uses the programmable rendering pipeline of the graphics hardware in order to construct the required texture representation. In order to do this it needs the layer description as well as the pulses, which describe the zones, encoded into a data texture (see chapter 5.5). This processing step, as well as the creation of the tile atlas, can be delegated to the server for thin clients.



**Figure 4.18: Thin Client using Shader on GPU** - The construction of the texture content is done on the GPU. This thin client requests the layer/zone information from the server instantly encoded into textures

#### 4. TEXTURE CONTENT MODEL

---

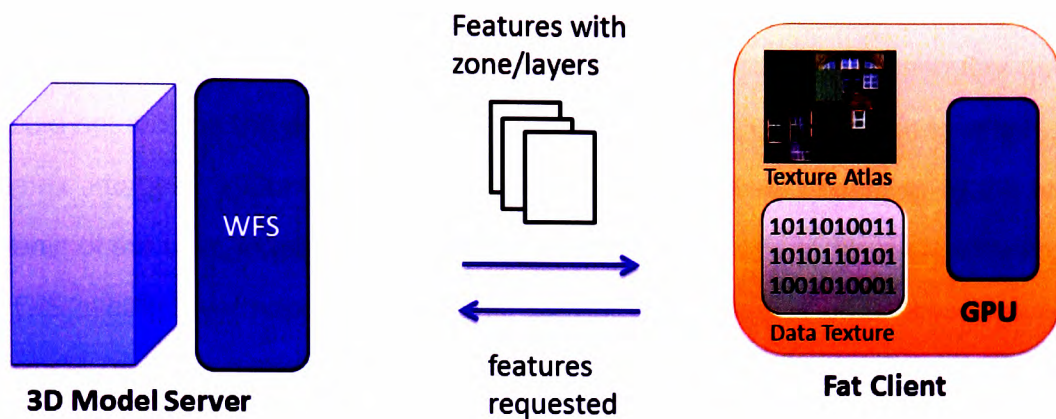
The server in this case would produce the tile atlas and the data texture and include it into the output, e.g. into an X3D or VRML scenegraph. The use of X3D/VRML, hence using the W3DS interface, seems more suitable in this case, as these output formats support elements to represent shader programmes. In this way it is also possible to transfer the required shader code for the construction of the texture from the server to the client. However, it would still need to be investigated how the 2-step rendering process described for the prototype in chapter 5 can be realized in this way, receiving the shader code from the server. At the moment it seems to be more feasible to assume that a specialized client is using an existing rendering concept in a shader programme that already exists on the client side and the client receives the appropriate pre-processed data texture and the tile atlas from the server.

As already explained the server (the 3D management system) provides specialized output in this scenario, which allows using the latest graphics hardware capabilities. By reconstructing the texture on the client's graphics hardware during rendering, hence at the very end of the process chain, the data size in terms of textures is kept small in comparison to full façade textures. During transmission and in client memory only texture tiles are used (packed into the texture atlas), which are significantly smaller than normal façade textures. Streaming content based on small portions of the texture is also proposed in Ricard et al. (2008) as a useful function. The reduction of data size is beneficial in terms of SDIs as data is transmitted over a network. However, data size also plays a major role when mobile clients are part of the application scenario.

For scenarios in which the client utilizes the programmable rendering pipeline a second solution is possible depending on the client's capabilities. In the previous solution the W3DS interface would be used to query a scenegraph including the two 'information textures' (data texture and tile atlas) for each layer-based façade texture for immediate rendering (type-3 model received from the server). Depending on the client it would also be possible to shift the processing step of generating the information textures to the client side (fig. 4.19).

## 4. TEXTURE CONTENT MODEL

---



**Figure 4.19: Fat Client using Shader on GPU** - The pre-processing (generating dataTexture and tileAtlas) can be done on client side for fat clients

In this case the client can work with the full type-6 model and still use the shader based reconstruction/visualization concept for the façade textures. This approach can be useful for 3D-GIS systems, for example. These 'fat clients' can access the full information (type-6) and use the features with their semantics and additional information for their own processes and workflows. However, they can at the same time visualize the flexible texture content and use the rendering pipeline for the texture reconstruction process. In this case the description needs to be transformed on the client to be loaded onto the graphics hardware.

This scenario, as well as the other two, shows that separate processing steps can be flexibly shared between client and server involving the two service interfaces WFS and W3DS. How the processes are actually shared depends on the scenario and the capabilities of the involved systems.

### 4.9 Discussion

After presenting the texturing approach for 3D city models the consequent question would be: what is the advantage of a layered approach in comparison to simple 'pixel-matrix' images? Content can be arranged and combined according to the requirements of the user, in real time if necessary. That is also one of the main advantages of a GIS system over paper based maps. Layers can be turned on/off and specific content can be pushed into the background or pulled into the foreground, depending on the importance for the user. When this flexibility should be achieved for building façades in 3D city models by standard image textures, it would be necessary to generate the required textures for each and every content combination in a pre-processing step and store them in a database. These textures would then be applied to the model according to user requests. This approach seems to be difficult to realize, not only because of the issues about data size and storage of all the relevant textures, but also because of the fact that the data provider would need to envision all possible combinations of content in order to create the required image files beforehand. The presented concept of flexible texturing for 3D city models in this thesis aims for the possibility to provide content layers especially for buildings' façades, which can be changed interactively and in real time.

#### 4.9.1 Pulse Function Approach

A similar approach using a pulse function concept can be found in Parish and Müller (2001) where it is used for generating valid façade representations for rule based generated cityscapes. The concept of arranging pulses in order to define façade element positions and group them in layers is comparable to the presented layer/zone-approach. The work of Parish and Müller (2001) supports the feasibility of the texture content model presented in this chapter and is an additional example for dynamic façade content creation. However, the intention is different. Parish and Müller (2001)



## 4. TEXTURE CONTENT MODEL

---

use pulses that are equidistant, arranged in layers and which describe valid naturalistic façade structures. They also use operations among layers to generate valid content avoiding overlaps between elements of different layers. Operations described in this chapter, like 'erase zone', can be linked to this approach, which automatically adapts content to make it more valid/realistic, e.g. erase a window that overlaps with a door. Although in this work the procedural modelling of arbitrary valid façades is not investigated, the example shows that appropriate operations can be implemented for other domains (and the texturing approach is influenced by other domains) and the layered façade content is very flexible and provides sufficient adaptability.

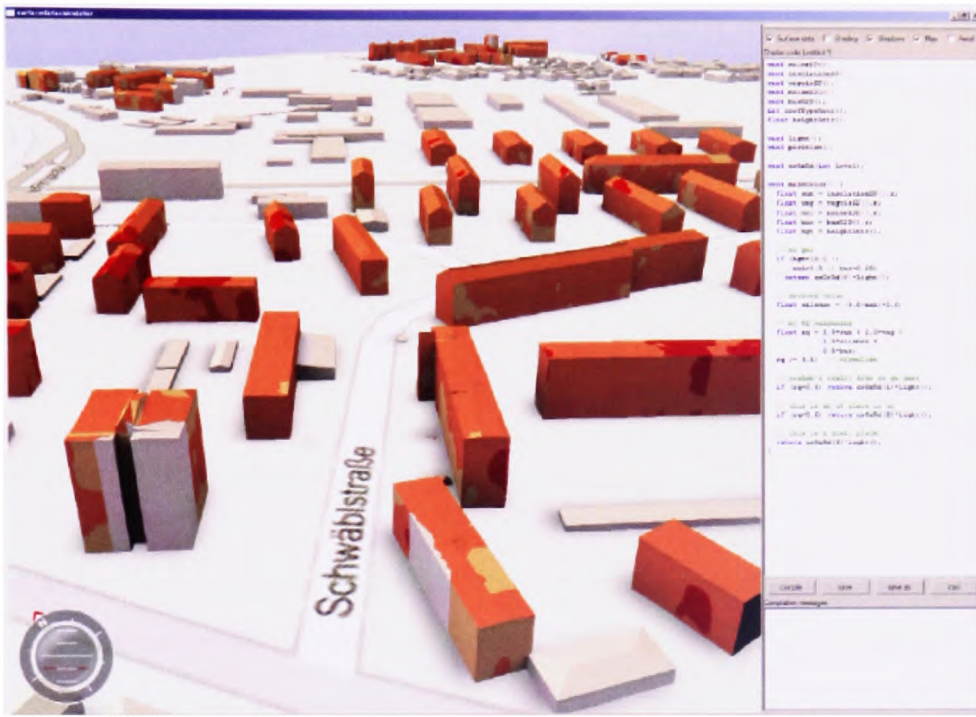
A difference of the presented approach compared to Parish and Müller (2001) is that the constructed façade structures in their work do not necessarily have a real world equivalent. In their paper they describe the construction of valid façade textures to be used for their procedurally generated building geometries. The adaption of façade content and the integration of additional content into real-world façades in terms of information visualization are not described.

### 4.9.2 A Similar Approach for Thematic Visualization

As discussed in chapter 2 the aim of this work is to use textures of buildings (façade textures) and other urban objects differently. These textures should not only act as containers for real-world photo images but the surface of the buildings should be able to act as 'map-space' in order to present various types of information. Thus, 3D city model textures need to be regarded as versatile displays for information visualization rather than restricting them to be a kind of photo-realistic wallpaper. In chapter 2 the work of Lorenz and Döllner (2010) was identified as relevant work, which shows that this is a sensible direction to take and that there are attempts to achieve this goal.

So in general the work of Lorenz and Döllner (2010) supports the idea of this work and shows that textures can be used as information channels and not only to apply

## 4. TEXTURE CONTENT MODEL



**Figure 4.20: Surface Properties Tool** - Residential quality visualization using surface properties. Users can create their own weighting and color mapping from all available data sources using small code fragments. Here, red tones denote suitable places and light orange tones unsuitable places (Lorenz and Döllner, 2010)

photo-images (see 4.20). However, their approach differs in terms of information management. In their case information can be merged by arbitrary operations. Having the ability to change the shader code directly provides a maximum of flexibility and seems to be appropriate for expert users. However, there is no clear way how to organize surface property information and to merge it with façade elements like windows, for example. This work argues that introducing a layer-based concept for textures can help to organize content. The layer concept is well-understood and functions like switching layers on/off are very familiar to both expert and non-expert users.

Looking at the approach presented in this work, it aims to provide a solution that is closer to existing 2D digital maps where layers are already a common element of map-content management. In terms of the map-use-cube defined by MacEachren and Kraak (1997) this solution might be seen in the synthesis and presentation area. The approach of Lorenz and Döllner (2010) might be considered as a tool for analysis and

## 4. TEXTURE CONTENT MODEL

---

exploration, which is more of an expert tool. Nevertheless, both approaches work as a new way to use object surfaces as information displays and it would be very interesting to see a combination of the surface properties approach and the layer/zone-based textures. It might be possible to integrate a calculated residential quality property, for example, with a window layer to be able to relate the calculated surface property to separate floors/flats of the building.

### 4.9.3 Real-Time Layer

Looking at the real-time layer as an extension to the basic layered texture content approach it describes an additional layer with a fixed z-position. The other façade content is arranged around this layer using two null-textures, one in front and one behind the real-time content. The question about this extension is, if it would be possible to integrate real-time texture content by using the basic pulse function/layer concept with one null-texture. This needs to be investigated in future work, as it implies a need to define a model to separate normal tile information that normally goes into the texture atlas, and the real-time content. The real-time content would have to be linked to a zone and it needs to be defined, that 1) the content is neither a primitive colour code (attribute representation) nor 2) a texture tile (from the atlas), but 3) a continuous stream of images, respectively a periodically changing image. This approach would need to be investigated, implemented and tested in the future as it changes the general assumption that zone information is either a simple colour or a texture tile from the atlas. The feasibility of the presented real-time layer approach is tested by implementing a prototype as a proof of concept. This implementation is going to be presented in the next chapter.

## 4. TEXTURE CONTENT MODEL

---

### 4.9.4 Integration into (3D-) SDIs

In terms of using the new texture content model in client-server scenarios within (3D-) SDIs the presented approach can be regarded as useful and practicable. The reconstruction of the final texture can take place either on the server or the client, hence on different stages of the process, according to the capabilities of the involved systems. The texture model shows sufficient flexibility and provides a structure that allows content to be delivered according to user requests and scenario requirements. An aspect for future work is the investigation of existing interface standards. The relevant standards need to be extended or adapted in order to expose the new capabilities of texture content definition. A further task for future work could be the definition of a CityGML ADE for zone/layer-based textures. This ADE would make it possible to exchange the raw texture content information together with the 3D city model between 3D management systems in a standardized way. These adaptations need to be examined and might also be introduced into the standardization processes of the OGC in the future.

# Chapter 5

## Proof of Concept

In the previous chapter the model for façade texture content for 3D city models was presented: flexible content based on zones, layers and texture tiles that form the final texture according to user requirements or scenario specific settings. As it is outlined in chapter 4.8 the texture content approach can be integrated in very different scenarios and hardware environments. In order to show that the concept can be implemented and is feasible in a technical sense this chapter presents a prototype that realizes the concept on a specific platform. The chapter is going to present an implementation that shows how modern graphics hardware with a programmable rendering pipeline can be used to realize the new approach for façade texturing. The construction of the texture content in this case is done directly on the graphics hardware, which implies that all processes before rendering can work with texture tiles instead of complete façade textures. This reduces data size for storage and transmission and allows changing the content of the façade in an interactive way and relatively fast because the construction of new content is part of the rendering process.

Integrating the concept into the rendering process is not absolutely necessary. Nevertheless, one major part of the concept is to be able to change content in real-time or with acceptable response times. This capability allows combining different information and switching between different representations. This also allows context sensitive changes of the representation of certain city model objects, which can be triggered by the application/system. Therefore the integration of the reconstruction process into the rendering pipeline allows changing texture content very quickly because this process is managed by the dedicated graphics hardware.

## 5. PROOF OF CONCEPT

---

This chapter will introduce properties of the platform on which the prototype is implemented as well as special aspects of the programmable rendering pipeline. It will also describe the developed pre-processing step that is needed to prepare the required information to be loaded onto the graphics hardware. The rendering process as such and certain particularities and algorithms are provided as well as a discussion on performance and scalability.

### 5.1 The Rendering Pipeline

In the presented prototype the null-texture is filled directly on the graphics hardware during the rendering process. This rendering process is normally realized in the rendering pipeline of the graphics hardware, respectively the Graphics Processing Unit. A common standard interface to the rendering processes is OpenGL.

*'OpenGL is an industry-standard, cross-platform Application Programming Interface (API). ... Its intention is to provide access to graphics hardware capabilities at the lowest possible level that still provides hardware independence. It is designed to be the lowest-level interface for accessing graphics hardware. ... The OpenGL API is focused on drawing graphics into frame buffer memory and, to a lesser extent, in reading back values stored in that frame buffer. It is somewhat unique in that its design includes support for drawing three-dimensional geometry (such as points, lines and polygons, collectively referred to as PRIMITIVES) as well as for drawing images and bitmaps. ... The fundamental purpose for OpenGL is to transform data provided by an application into something that is visible on the display screen. This process is often referred to as RENDERING' (Rost et al., 2006).*

The process of rendering includes several steps to transform the three-dimensional primitives, which are viewed from a certain perspective into a two-dimensional image

## 5. PROOF OF CONCEPT

that is written to the frame buffer in order to be visualized on the display screen. The rendering operations that are performed in the OpenGL process are depicted in figure 5.1.

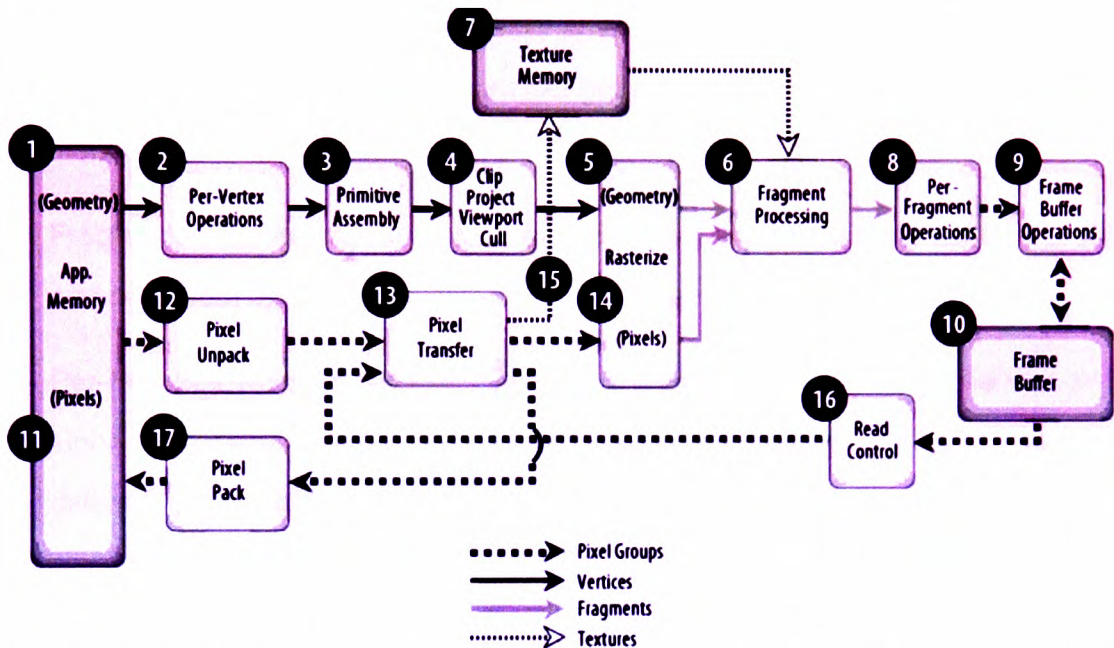


Figure 5.1: OpenGL Operations - Overview of OpenGL operation (Rost et al. (2006))

The major steps of the process are (detailed description in Rost et al. (2006)):

- **Per-Vertex-Operations** (2): include transformation of vertex positions by the ModelView-Matrix, colour and material are applied, texture coordinates are calculated if required and lighting calculations are performed. *'Because the most important things that occur in this stage are transformations and lighting, the vertex processing stage is sometimes called T&L* (Rost et al., 2006).
- **Primitive-Processing** (4): *'actually consist of several distinct steps that have been combined into a single box to simplify the diagram'* (Rost et al., 2006). This step includes clipping, perspective projection, viewport transformation and culling.
- **Rasterization** (5): primitives are decomposed into smaller units which match the pixels in the frame buffer. *'Each of these smaller units generated by rasterization*

## 5. PROOF OF CONCEPT

---

*is referred to as a fragment. ... A fragment comprises a window coordinate and depth and other associated attributes such as color, texture coordinates and so on. The values for each of these attributes are determined by interpolation between values specified (or computed) at the vertices of the primitive'* (Rost et al., 2006). The fact that each fragment has got a texture coordinate is used to implement the content model presented in chapter 4.

- **Fragment Processing** (6): texture mapping happens in this stage, as well as fog and color sum operations.
- **Per-Fragment-Operations** (8): this set of operations includes the scissor test, alpha test, depth test, alpha test, etc. There are also operations for blending different colours and dithering.
- **Frame Buffer Operations** (9): frame buffer operations determine into which part of the frame buffer is written. For example, it is possible to write either in the front or back buffer, respectively the one that is not used for screen display. When the content is written completely to the back buffer it becomes visible and the front buffer is used for rendering new content to the frame buffer (buffer swap).

The developed prototype is a 3D Viewer Java application which implements the texturing approach and capabilities presented in chapter 4. Access to the OpenGL API is achieved by the Java Bindings for OpenGL (JOGL)<sup>1</sup>, which implements the OpenGL API for Java. The 3D model data is provided by the 3D management framework 'CAT3D', which is developed at HFT Stuttgart by the author of this thesis<sup>2</sup>(see also chapter 2). This framework is presented in Bogdahn et al. (2007) and in relation to the presented texturing approach in Bogdahn and Coors (2009a). The link between the CAT3D framework and the developed prototype will be described in more detail in subsequent sections.

---

<sup>1</sup><http://jogamp.org/jogl/www/>, last accessed August 2011

<sup>2</sup>Developed for the authors Diploma Thesis (unpublished)(Bogdahn (2006))



## 5. PROOF OF CONCEPT

---

However, as the prototype intends to implement a texturing approach different to the fixed-function pipeline it depends on programmable parts of the pipeline where custom code can be executed.

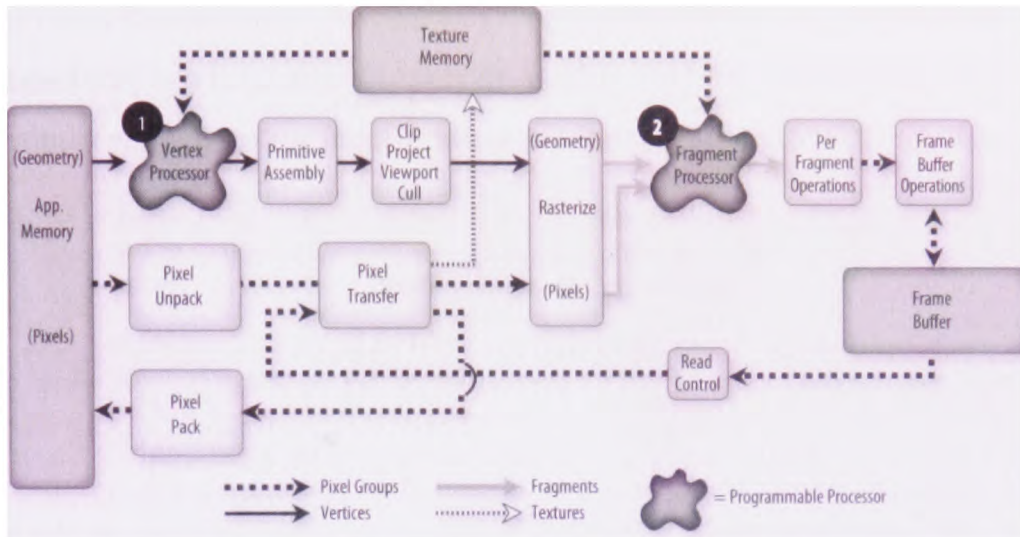
### 5.2 The Programmable Elements of the Rendering Pipeline

The elements referred to in the previous section describe the fixed function rendering pipeline that OpenGL defines. The processes that are conducted inside these elements can be controlled and adapted by using different parameters that can be set through the OpenGL API. One *'could think of OpenGL as a sequence of operations that occurred on geometry or image data as it was sent through the graphics hardware to be displayed on screen. Various parameters of these pipeline stages could be altered to select variations on the processing that occurred for that pipeline stage. But neither the fundamental operation of the OpenGL graphics pipeline nor the order of operations could be changed through the OpenGL API'*. (Rost et al., 2006)

By using the functions and parameters provided by OpenGL a certain degree of control over the actual process can be achieved. However, for many applications this is not sufficient. Looking at the method for filling the null-texture in this work, this concept needs much more flexibility as it needs to implement a new algorithm that needs to be processed on the graphics hardware.

Since OpenGL 2.0 the rendering pipeline provides programmable processors for vertices and fragments, which can execute so called 'shader programmes' or 'shaders'. Figure 5.2 *'shows the OpenGL processing pipeline when programmable processors are active. In this case, the fixed functionality vertex and fragment processors are replaced by programmable vertex and fragment processors. ... All other parts of the OpenGL processing pipeline remain the same'* (Rost et al., 2006).

## 5. PROOF OF CONCEPT



**Figure 5.2: OpenGL Programmable Processors** - OpenGL logical diagram showing the programmable processors for vertex and fragment shaders rather than fixed functionality (Rost et al., 2006)

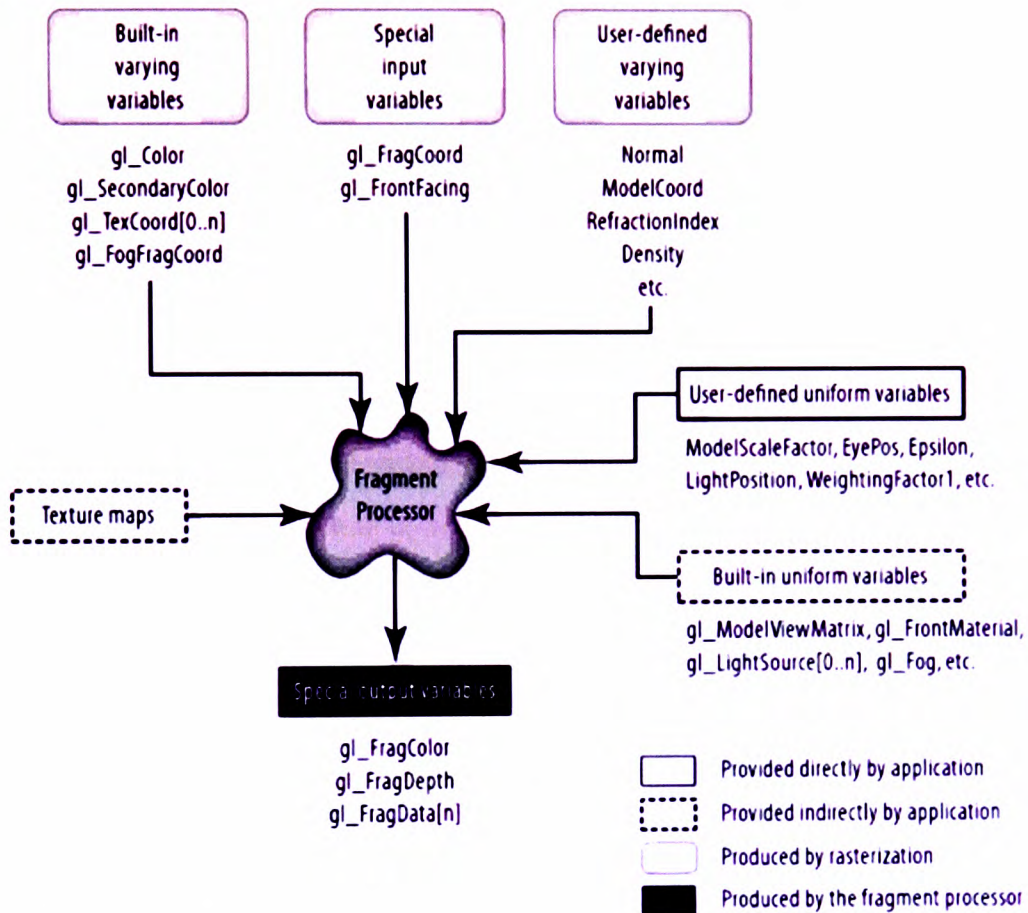
As this work will primarily use the capabilities of the fragment processor, the programmable vertex element will not be discussed here. In this research work a program for the fragment processor is developed that implements the logic and algorithm of filling the null-texture and constructing the requested façade representation. The fragment processor as a unit in the rendering pipeline *'performs traditional graphics operations such as the following:*

- *Operations on interpolated values*
- *Texture access*
- *Texture application*
- *Fog*
- *Color sum*

*A wide variety of other computations can be performed on this processor'* (Rost et al., 2006). However, the fragment shader programmes need to be written in a way, which performs computations only for a single fragment at a time. As fragment processing

## 5. PROOF OF CONCEPT

can be highly parallelized on hardware the fragment processor does not know anything about neighbouring fragments and only works on a single fragment at once. The inputs and outputs of the fragment processor are depicted in figure 5.3.



**Figure 5.3: Fragment processor inputs and outputs** - OpenGL inputs and outputs for the programmable fragment processor (Rost et al. (2006))

As one can see the fragment processor is responsible for texture access and for applying the texture information to fragments. Therefore it is the suitable place for working on the construction of the requested façade texture representation. The fragment processor also has access to texture maps (texture objects), can read information from them and incorporate this information into the fragment computations.

As we will learn in subsequent sections the prototype uses textures to transfer required information for the construction of the façade texture (tiles and description) instead of using uniform variables. One reason for this is that the fragment processor is the ded-

## 5. PROOF OF CONCEPT

---

icated element to process texture maps and incorporate them into fragment calculations. The second reason is that through textures more information can be transported than by uniform variables (depending on the hardware). The third reason is that textures might be easier to transmit by standard data formats, like for example VRML. These formats normally know how to handle a texture or a multi-texture. So if the construction information for filling the null-texture can be encoded into textures, it is easier to integrate them into external data formats for transmission in a client-server environment than a relatively big set of integer or double values.

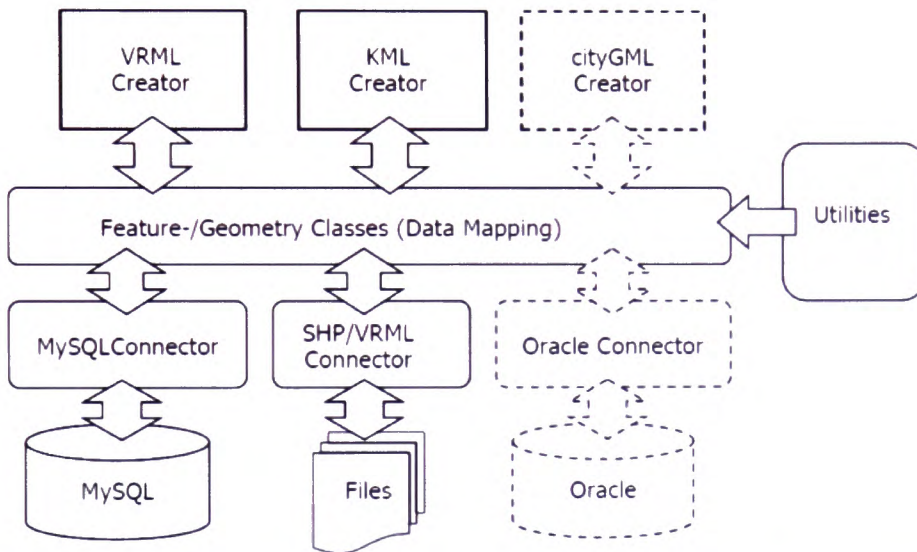
However, the description of the zones and layers and the structure of the façade will certainly not be stored in a database or in the file system in form of textures. Instead, one could think of an extension to the CityGML standard in order to describe the zone/layer-based texture content or a special database schema. The prototype specific encoding of this type of information into textures needs to be generated beforehand, in order to be used directly on the graphics hardware. Therefore, encoding the information into a texture needs to be done in a pre-processing step before the façade content can be generated directly during the rendering process.

### 5.3 Data Management and 3D Viewer Integration

As pointed out in the previous section the information that needs to be transmitted to the fragment processor in order to construct the visual façade representation has to be transformed into a texture, which can be accessed by the fragment shader program. The description and the texture tile information are not in this form on the application side and therefore needs to be pre-processed. The presented prototype is a Java application that uses the Standard Widget Toolkit (SWT) and JOGL in order to implement a 3D Viewer that is capable of handling the developed façade texture description and to visualize it. The 3D city model as such, as well as the information needed for the new texture content approach, is modelled as Java classes. The geometrical and semantic

## 5. PROOF OF CONCEPT

model, as it is described in chapter 2.2.3, for type-6 city models as well as the standard texturing approach, is included in the internal data model of CAT3D. This framework was developed in order to manage, to process and to provide 3D city models (see Bogdahn (2006)). The general architecture of the framework is depicted in fig. 5.4.



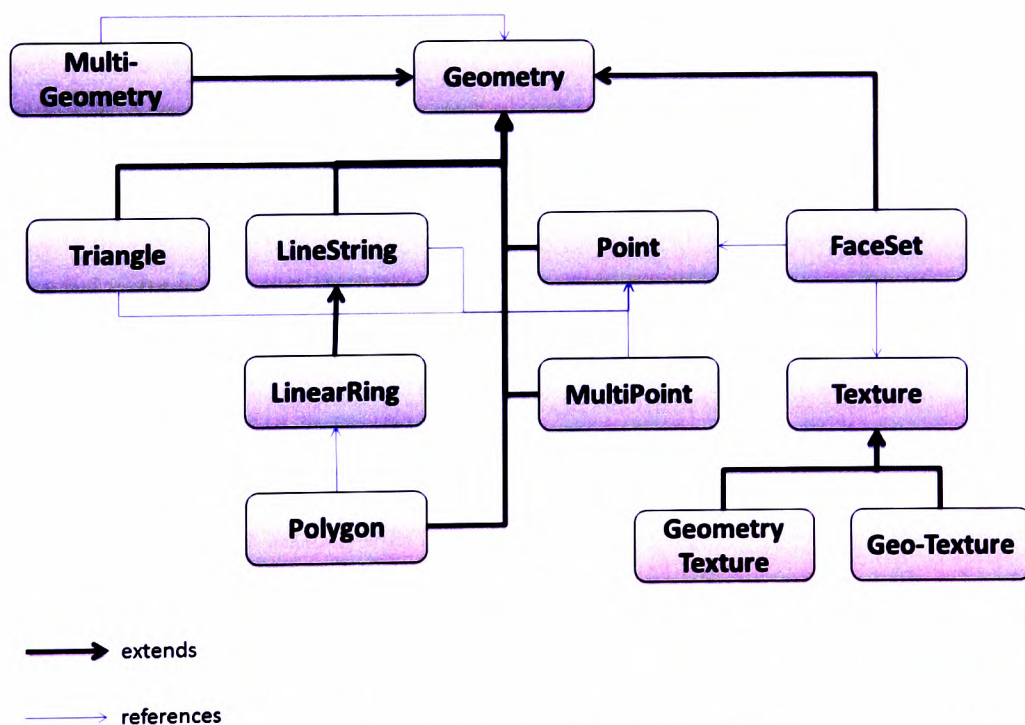
**Figure 5.4: The CAT3D Architecture** - CAT3D framework - components diagram (own depiction)

Briefly recapitulating what was already described in chapter 2.2.3 this framework is modelled by three layers: the data-access layer, the internal data layer (mapping layer) and the format-creator-layer (output layer). By using specific components of the framework one can generate a work-flow for a particular scenario that the system should realize. For example, a web-service implementation could use a database access module to read data from a spatial database and convert it into the internal representation. Utilities could be used to transform the spatial reference system into the one that was requested by the user. The appropriate format-creator would transform the data from the internal data representation into the requested output format. As the single modules can be combined relatively independently it is rather easy to change the output that the web-service produces. The module for data output can be exchanged without the need to change the rest of the process (data access and coordinate transformation) because other format-creator modules also work on the internal data representation.

## 5. PROOF OF CONCEPT

The only part of the process that changes is the translation from the data mapping layer into the new output format.

The presented prototype uses only parts of the framework and integrates it into the 3D viewer application on the data mapping layer. The data-access-layer and the internal data layer are used to access a 3D city model in an external format (e.g. a CityGML-file or a VRML file). The external format is then translated into the internal representation (fig 5.5). As the prototype application has now access to the model data (vertices, polygons, textures, etc.) and access to the OpenGL API through JOGL the model information can be translated into rendering instructions for the graphics hardware. In this scenario no format-creator modules are needed, because the output that the application needs is the rendering command that it sends to the graphics card.

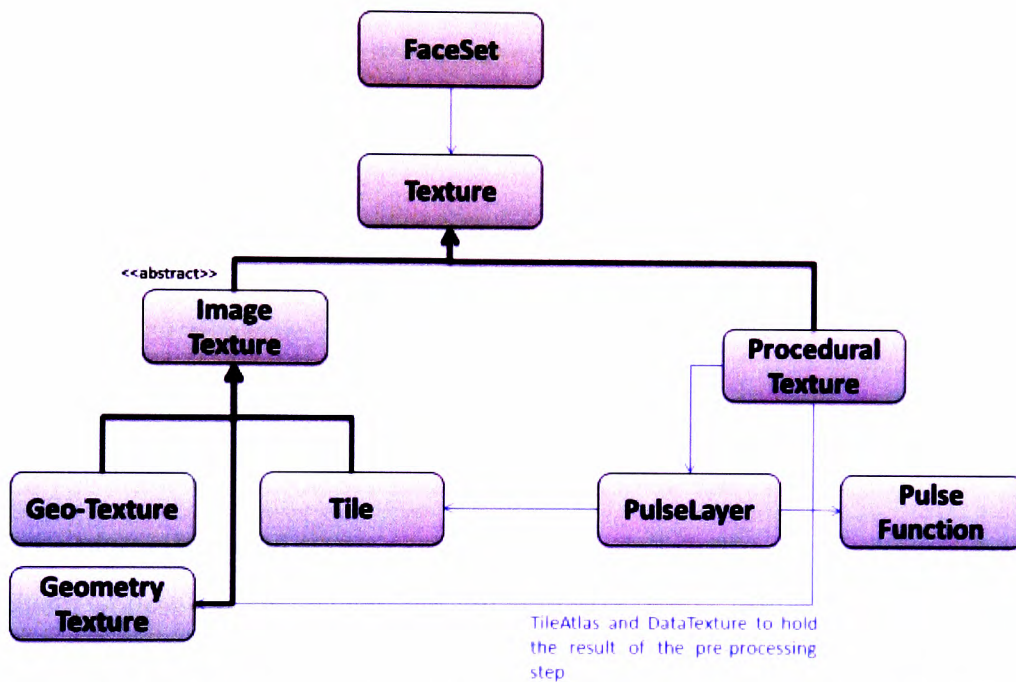


**Figure 5.5: The CAT3D Data Model** - Internal data representation of the CAT3D Framework (Bogdahn, 2006)

The CAT3D framework, however, only modelled standard rendering approaches, therefore only the standard texturing concept as well. One can see in figure 5.5 the texture

## 5. PROOF OF CONCEPT

object that is linked to a FaceSet is a reference to an image file and a set of texture coordinates. Therefore the texture model needs to be extended in order to hold the zone/layer/tile information for a façade texture. The diagram for the part of the extended texture model including pulse-functions, layers, etc. can be found in figure 5.6.



**Figure 5.6: The Extended Texture Model** - Extensions made to the existing texture model of CAT3D in order to represent the new texturing approach

The new texture element is actually an implementation of the existing Java interface (Texture) that is already present in the CAT3D framework. Hence, a model can either have a standard image texture that is handled in the traditional texture mapping way or it can have a texture that has layer-based content (Procedural Texture). In that way it is not necessary for a city model designer to provide the new texture content type for each building in the model. The flexible content can just be provided for buildings where it is necessary (e.g. landmarks) and other buildings can be textured in the traditional way, if required in the specific use-case.

In chapter 4.8 the different stages at which the façade texture can be constructed was discussed in detail. At this point one can see that the construction of the texture

## 5. PROOF OF CONCEPT

---

by analysing the zones, layer, etc. could be done on 'Java-side' (by the application) and a complete texture could be provided for the rendering process. Nevertheless, the implemented prototype intends to show that the reconstruction can also be done directly on the graphics hardware. In order to do this, the Java-based representation of the pulses, layers and tiles needs to be transformed into a form that can be accessed by the programmable fragment processor in the rendering pipeline. This conversion is going to be described in the next section.

### 5.4 The Pre-Processing Step implemented as Utility Functions

The CAT3D framework provides another element in its architecture that is not part of the three aforementioned layers of the system architecture. The Utility-classes allow specifying worker-classes, which perform specific tasks on the internal data representation inside the framework. These Utility-classes implement things like: extrusion of footprints to build block models, perform a Delaunay Triangulation on a set of terrain points or, as mentioned in the previous section, calculate a coordinate transformation from one spatial reference system into another. The Utility-classes and their functions can be used independently of the input format because they work only on the internal data representation.

One function that is added into the utility set of the framework generates a texture that holds the description of the visual façade content. This function encodes the layer information and the zones, respectively the pulse functions that describe the zones, into this texture. The particular encoding into the texels will be described in the next section. The tiles that are used as the visual representation of a specific façade element and are applied to the appropriate zones are managed in a different way. As they can hardly be transmitted to the graphics hardware and used by the fragment processor



## 5. PROOF OF CONCEPT

---

as separate texture objects, they are packed into a texture atlas, or in this case: a tile atlas. The principal functionality of a texture atlas is introduced in chapter 4 and the implementation used for the prototype is described in the following sections.

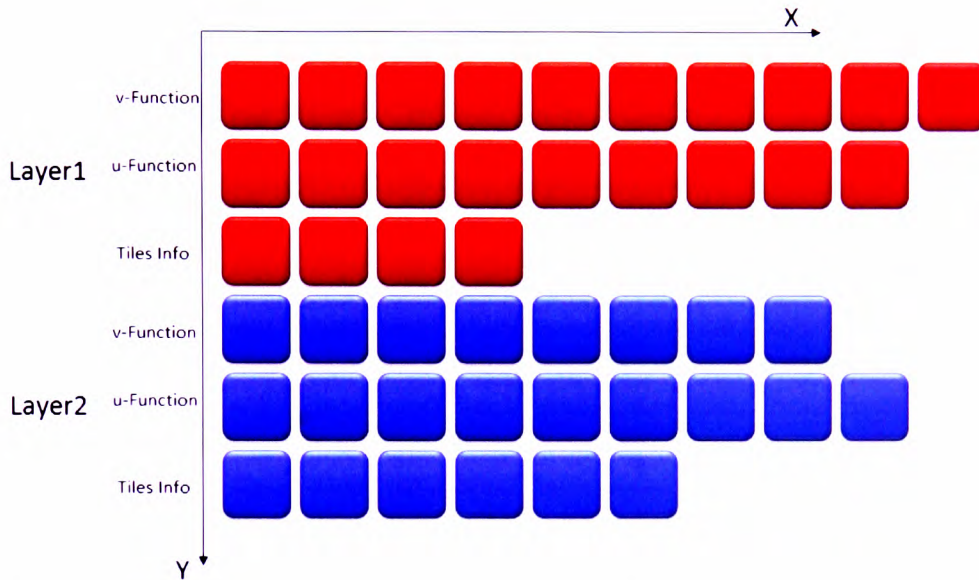
### 5.5 Data Texture

One of the aforementioned utility functions implements the pre-processing by encoding the façade description into a texture, which will be called data texture throughout the remaining part of the thesis. As the façade description includes a significant amount of information that needs to be transferred to and managed by the graphics hardware, a texture is regarded to be the best way to handle this. The data texture is a 32-bit floating point texture, which means it can store 4 floating point values per pixel in its RGBA-channels. In the remaining section the special structure of the data texture will be presented showing where and how the façade structure information is stored.

The main element of the data texture is the layer covering three rows of the texture. The z-order priority of the layer is reflected by the order of the layers inside the texture. Here the first layer is the topmost, the second layer is one level below, and so on (see fig. 5.7). The utility function performing the pre-processing actually orders the layers based on their priority attribute on Java side before generating the texture.

Arranging the layers in their final z-order inside the data texture also supports the reconstruction process in the fragment processor. When the corresponding zone is looked up for the currently processed fragment it is not necessary to go through the entire layer list. If one layer is identified as the one with the valid zone it is irrelevant if there is a zone in another layer containing the same fragment, because the second layer would be behind the first one.

## 5. PROOF OF CONCEPT



**Figure 5.7: Layer encoding into Data Texture** - Layer structure in the data texture. Each colour represents one layer (red/blue).

A layer itself is again subdivided into three rows, each row holding specific information about this layer.

- Row 1: the v-Function
- Row 2: the u-Function
- Row 3: the texture tile information (sequence)

Arranging the three rows in this way (v, u, tiles) has also to do with the search strategy for the zone for a specific fragment. Having the u- and v-function and allowing the programme to iterate through all rows of data texture sequentially it would always analyse the v-pulses first. If it finds a v-pulse that is valid it would look for a u-pulse to validate that the fragment is inside a zone. Assuming that there are more buildings having more window/element rows than number of floors, it is more efficient to check the v-pulses first. If there is no match the numerous u-pulses do not need to be checked at all. This assumption is especially true for windows but might also be valid for other façade elements having more horizontal structures than vertical repetitions. As windows are likely to be the most frequent elements in number, this search strategy is appears most

## 5. PROOF OF CONCEPT

---

effective for this element type. To flip u- and v-functions for other elements appearing fewer times should have no negative effect on the search result in their cases and therefore the concept can be very effective for the overall reconstruction process. Additional search strategies and details on the reconstruction process can be found in a later section in this chapter.

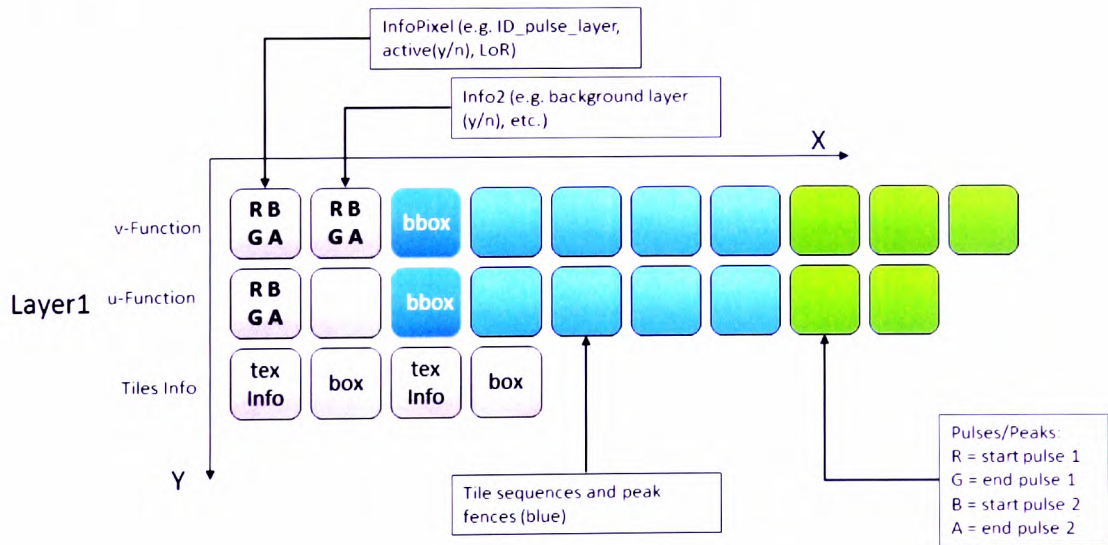
The rows for u- and v-functions are structured differently than the tile information. The function rows start with an info-section: Pixels 0 and 1 holding information about the layer as well as some implementation specific values:

- Pixel 0:
  - **Red**: combine this function with function in row number X - in this implementation the row is always combined with the function in current row + 1.
  - **Green**: length of this function row
  - **Blue**: layer active (0/1 → false/true)
  - **Alpha**: LoR value of this layer
- Pixel 1:
  - **Red**: layer ID
  - **Green**: wrapping type for tiles to zone mapping for all zones in the layer
  - **Blue**: layer operation (not used in this implementation)
  - **Alpha**:intoBackground (true/false → 1/0), should this layer be in the background texture when the real-time layer is active

The second section (pixel 4-6) stores the 'fences' of the pulse function. This is implementation specific information that supports the search strategy of the reconstruction process. In order to find the zone for a given fragment in the final texture the process needs to check all the pulses in u- and v-direction. To minimize the search effort the fences are stored. If the current fragment's u-position is 0.365 in the null-texture, consequently, only the pulses between 0.3 and 0.4 need to be checked. The fence information pixels in the data texture store the pixel positions where the pulses for the range

## 5. PROOF OF CONCEPT

0.3 to 0.4 can be found. With this information from pre-processing the reconstruction programme does not need to check all pulses of a function but only the pulses for the given range. If there are no fences for this range the process can skip this layer, as there are no pulses for the given fragment, and go on with the next layer. Between the info pixels and the fences there is the sequence information for this layer (pixel 3). In this part of the function rows the index of the associated tiles is stored. In this way a layer can be linked to a list of texture tiles and the sequence describes in which way the texture tiles will be applied to the zones of the layer. A sequence of 1, 2, 3 defines that tile 1 is applied to first zone in the content description, tile 2 to the second and tile 3 to the third zone. This pattern is repeated for subsequent zones of the layer. In pixel 2 the layer bounding box is stored, which will be explained later.



**Figure 5.8: Pulse Rows in DataTexture** - The structure of the pulse function rows in the DataTexture encoding

After the sequence information there is an arbitrary number of pixels holding pulse information for the specific pulse function (starting with pixel 7). Each pixel can store the start and the end values for two pulses:

## 5. PROOF OF CONCEPT

---

- Pixel Pulse:
  - **Red**: start of pulse 1
  - **Green**: stop of pulse 1
  - **Blue**: start of pulse 2
  - **Alpha**: stop of pulse 2

This information is actually the 'body' of the pulse function row.

The third row of each layer section is the texture tile information. Each tile is encoded into two pixels:

- Pixel 0:
  - **Red**: void not in use at the moment
  - **Green**: ID of texture unit of the tile atlas (not in use in this prototype).
  - **Blue**: type: 0 tile, 1 bump map, 2 shadow map, etc. (not in use in this prototype)
  - **Alpha**: rotated tile
- Pixel 1:
  - **Red**: x-coordinate of upper left corner
  - **Green**: y-coordinate of upper left corner
  - **Blue**: x-coordinate of lower right corner
  - **Alpha**: y-coordinate of lower right corner

In this row the index of the tile in the list of texture tiles is referenced by the tile sequence in the two function rows. Hence it is possible to reference the according tile in the texture atlas (see next section).

The data texture is one result of the pre-processing step implemented for the presented prototype for layer/zone-based façade texturing. It is a specialized output in terms of the data structure including additional information supporting the search strategy. And it is also in a special format, generated as a texture, in order to transport a large data set onto the specific platform (the GPU in this scenario).

### 5.6 The Texture Atlas

The second output of the pre-processing step is a texture atlas combining all tiles of a zone/layer-based texture or the tiles for all textures for one building, depending on the structure of the building's geometrical and semantic model. A building having several wall geometries, linking a separate layer/zone-based texture for each wall, can have one texture atlas per wall. If the `GL_MAX_TEXTURE_SIZE`<sup>1</sup> of the graphics hardware makes it possible, the tiles for all walls/all textures in one building can be combined into one atlas.

As presented in chapter 4 the mapping of the single elements of a texture atlas onto the geometry is achieved by transforming the original texture coordinates to the position in the texture atlas. Texture coordinates normally define how a texture (coordinate range [0, 1] in x and y) is stretched onto the geometry. In an atlas the original texture is a sub-part of the atlas texture and the texture coordinates need to be transformed to match the [0, 1] range of the texture atlas. As the atlas coordinates are stored in the data texture for each tile, it is possible to find the tile in the atlas and retrieve the texel colour for the specific zone in order to fill the null-texture. The detailed description for the rendering process can be found in the next section of this chapter. The construction of the texture atlas is done using a standard data structure: a Binary Space Partitioning Tree (BSPTree)<sup>2</sup>. This tree structure divides space in half spaces in order to place elements and find the next available half space to add additional objects (see chapter 4).

In the implementation for the presented prototype the tiles are ordered according to their size and added into an approximately estimated initial size atlas. Tiles can also be rotated by 90 ° to fit into an area before searching further into the tree to find a suitable space. Rotated tiles are marked with a flag in order to indicate how the tile

---

<sup>1</sup>OpenGL parameter `GL_MAX_TEXTURE_SIZE` returns the maximum data size for a texture for the specific hardware

<sup>2</sup>see <http://www.blackpawn.com/texts/lightmaps/default.html>

## 5. PROOF OF CONCEPT

---

needs to be read from the atlas to provide the correct colour value for the particular texel. In the rare case the initial size of the atlas is too small the atlas will grow in both dimensions and the BSPTree is generated again.

There are well known issues about texture atlases in computer graphics, e.g. 'colour bleeding'. These artefacts mainly result from filtering operations during texture look-ups for a specific texel and mipmap generation. Because tiles in a texture atlas are normally positioned next to each other texture look-ups on the edge of a tile can be influenced by neighbouring sub-textures. This can lead to artefacts in the final façade texture. There are ways to avoid this behaviour, like one pixel buffers around each tile or by choosing a specific structure to arrange tiles inside the atlas, so that filtering for mipmaps is not a problem. The texture atlas generation for the presented prototype does not support concepts in order to prevent artefacts resulting from colour bleeding and other effects, as the atlas is only one part of the proof-of-concept for flexible texturing of 3D city models. Nevertheless, findings in other disciplines in terms of optimizing texture atlas utilization can be included into the pre-processing step in future work. Investigating optimization of the texture atlas concept, however, is not part of this work.

### 5.7 The Rendering Concept

In the two previous sections the input data for the fragment shader programme is described. The necessary information about the pulse functions, the layers as well as the texture tiles is present and accessible for the fragment processor. As can be seen in figure 5.3 there is also the possibility to set user-defined 'uniform variables'. These variables can be set directly by the application to be used as input for the shader programme. Uniform variables in the presented prototype are used as parameters for the façade texture construction process. In this way the application, hence the user, can control the construction, and thereby the content of the texture. One uniform variable that is implemented is the LoR, described earlier in the chapter on the texturing con-

## 5. PROOF OF CONCEPT

---

cept. By setting the appropriate LoR value for this uniform variable, it can be influenced which layers are turned on/off.

At this point we have all the necessary information at hand inside the fragment processor (data texture and tile atlas) and control over the construction process by using the provided uniform variables. In the following part the rendering process as such will be explained in detail.

### 5.7.1 The Construction of Façade Texture Content

The presented texture approach in this work is based on the null-texture that is applied to the associated wall geometry, if the building is constructed of semantically classified sub-geometries, or the whole building geometry, when the model is generated by individual buildings having a single overall geometry. The concept works with an empty texture (the null-texture) applied to the geometry by using texture coordinates, which would also be present when using the standard texturing concept. This empty texture matrix is filled by the developed fragment shader programme implementing the reconstruction process for the façade content.

As we learned at the beginning of this chapter each fragment has interpolated texture coordinates generated during the rasterization process. The pulse-functions and the thereby defined zones are also based on texture coordinates. The task for the fragment shader programme is now to analyse the texture coordinates for the currently processed fragment (its position in the null-texture) and to find the associated pulses (the zone) in which the fragment is located. If the zone is found the appropriate action needs to be triggered. In most cases a texture look-up in the linked texture tile is necessary, but other actions are also possible.

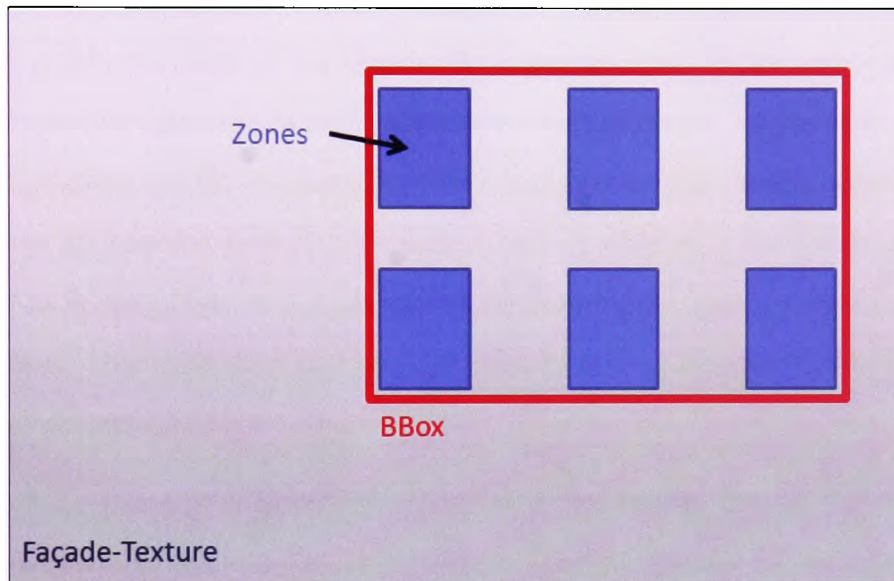
On order to find the zone the 'reconstruction shader' iterates through all rows of the data texture. It starts with the v-function of the first/topmost layer. Firstly the programme needs to find out if the layer is active and belongs to the currently set LoR. If the layer



## 5. PROOF OF CONCEPT

---

does not match the current settings it can be skipped. One implementation-specific piece of information that is also added to the description during the pre-processing is the minimum bounding rectangle (bbox) of the layer content. The bbox contains the minimum and maximum coordinates of all zones in a layer (fig. 5.9).



**Figure 5.9: Bounding Box for Zones in one Layer** - The Bbox for all zones in a layer is stored in order to determine if a specific texel is probably inside a zone

The bounding box speeds up the search process by excluding layers from the further look-up process. When a fragment is located outside of the bbox of the checked layer this layer is skipped consequently and the further steps described below do not have to be performed.

When the fragment is situated inside the layer bbox the next available information is read: the fences (see section 5.5). This information is used to determine the pixels in the row, which have to be checked for the currently processed fragment. If the fragment position is larger than 0.3 and the fence information tells the programme that pulses larger than 0.3 start with pixel 13 in the data texture, for example, the shader starts searching for the matching pulse at that pixel, saving the effort for looking up the pulse

## 5. PROOF OF CONCEPT

---

information in previous pixels. Having found a pulse containing the current fragment the shader does the same for the u-function.

If the layer in which the identified zone is situated defines a colour look-up in the associated texture tile the fragment shader needs to read the texture atlas coordinates (the tile bbox) and look-up the colour inside the tile. By using the information in the sequence pixels the index of the texture tile is determined. At the moment of writing the prototype only uses the first tile associated with the layer, hence only sequences with a single entry can be used and therefore only one tile can be associated per layer. By defining appropriate look-up concepts it can be defined if the tile is going to be stretched to fit the pulse, or if it should be repeated in x- and/or y-direction until the zone is filled. This behaviour can be controlled by setting the description's wrapping-parameter accordingly.

As the look-up process is rather computationally expensive, because it includes numerous texture look-ups into the data texture to read the description, as well as various if-else branches, which is an issue on some of today's graphics hardware, it cannot be performed for each and every frame of the rendering process. However, reconstructing the texture each frame would be meaningless because the content of the façade texture would not change in every frame. If performed for each frame the frame rate would drop unnecessarily although the reconstruction would produce identical results for the majority of the rendered frames. Therefore the actual rendering process is split into two parts. For each part of the process there is a separate fragment shader programme. One shader is implementing the standard texturing, which uses the final texture produced by the second shader, the reconstruction shader. That means, if there is no reconstructed façade content present for a null-texture, the reconstruction shader is called to produce it according to the description in the data texture and the settings for LoR, etc. The reconstruction shader actually renders the final texture into a Frame Buffer Object (FBO). This technology is provided by newer graphics hardware and offers additional frame buffers to which the graphics pipeline can render. These

## 5. PROOF OF CONCEPT

---

additional buffers are different to the screen buffer and therefore the rendered content cannot be seen on screen. This process runs in the background. The frame buffer to which the reconstruction buffer is rendering to is linked to a texture object. Hereby the result of the reconstruction process is stored in a texture object in graphics memory and this texture object can be used by the 'standard' fragment shader and be applied to the geometry in the standard way.

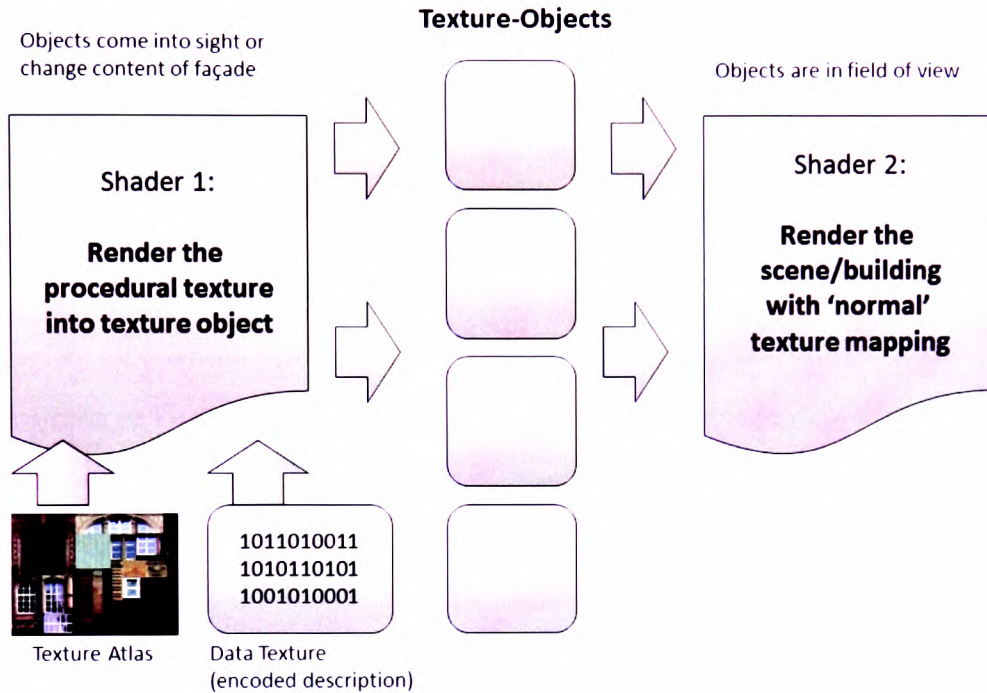
The reconstruction shader is only called when the content of the façade needs to change. A change of content is triggered by:

- Layer(s) change(s) z-order
- Layers are switched on/off
- Change of set LoR
- Change of layer set
- etc.

In these cases the existing texture object is deleted and the reconstruction shader generates a new representation for the façade according to the new settings made by the user or events generated by the application.

By using the 'general' texturing approach for the majority of the frames the workload and rendering time per frame is in the range of 'normally' rendered 3D city models. Strategies for performance optimization, like distance dependant details or concepts for loading/streaming new content into the scene can be applied as for 'standard' 3D city models. One example is the use of Quad-Trees for loading detailed terrain objects near the viewpoint and less detailed terrain that is located in greater distance (Kersting, 2002). The reconstruction process needs much more 'computations' and the frame time for this part is subsequently higher, therefore it is consequent to avoid this computation and only trigger it when necessary. As content can also change for

## 5. PROOF OF CONCEPT



**Figure 5.10: Two level rendering** - The 2 level rendering process.

objects that are in sight of the user the reconstruction cannot always be done 'in the background' or when the building is outside the view volume. The reconstruction for nearby objects in sight should be done first. The normal strategy for the reconstruction would be to create the façade texture when the object is just about to come into sight but is still not visible to the user. As the content of the façade does not have to stay constant in the presented approach changes can happen any time and for any building as they are also triggered by the user. Therefore the user would want reconstruction to happen relatively fast after the change was requested. Prioritization of the reconstruction for nearby objects needs to be realized so the user can actually see them almost instantly whereas changes to objects in the background are allowed to take longer. This concept still needs to be implemented for the presented prototype but is regarded as very useful for visualizing very large 3D city models to optimize the user experience. Additional issues on performance and scalability will be discussed at the end of the chapter.

## 5. PROOF OF CONCEPT

---

### 5.7.2 The Real-Time Content

In chapter 4 an additional layer for real-time content was introduced. This layer allows changing the real-time content inside a texture independently from the façade content. The concept is based on two null-textures that hold the façade elements for the foreground and the background of the real-time layer.

The content for the real-time layer needs to be provided as a 2D digital image in order to be incorporated into the façade texture. In order to manage this data that is part of the façade texture content a new interface is introduced in the Java class hierarchy. Each layer/zone-based texture can have one real-time layer. The RealTimeLayer-Interface defines how real-time content can be accessed.

The interface defines the following functions, which need to be implemented by classes that provide real time data:

- **getFrame()/isFrameUpdated()**: the getFrame()-function returns the generated image/frame that can be converted and load as a texture to the graphics hardware. The isFrameUpdated()-function returns a Boolean value. It indicates if a new frame is available or not. The 3D viewer checks this flag in each rendering cycle and it only loads the new frame to graphics memory when the content has changed, respectively the player has created a new frame.
- **start()/stop()**: these two functions start/stop the player. These commands are very obvious for a video, but in general the two functions start or stop the process of producing new frames that are going to be visualized in the real-time layer. Therefore they are not exactly the implementation of a video player start/stop function in a strict sense.
- **activate()/isActive()**: the activate function actually tells the rendering process if the real-time layer should be used and if a background null-texture is part of the rendering process. This state is independent from the start/stop functionality,

## 5. PROOF OF CONCEPT

---

because a video for example can be stopped (paused) and started again, while the real-time layer is still active during the pause phase. The function `isActive()` can be used to check the current state of the layer.

- **`getWidth()/getHeight()`**: returns the width/height of the frame that the player produces. Different players can generate differently sized output frames. Using the two functions the dimensions of the output frame can be requested.
- **`getPulses()`**: as it will be described in one of the next sections the real-time content does not necessarily need to cover the whole façade texture area. It can be reduced to a certain part of the final façade texture. The (real-time) pulses define a box within the null-texture where the real-time content should be applied.

All implementations of the `RealTimeLayer-Interface` need to implement the logic of transforming a data stream of arbitrary type into a 2D image because this is the only way that is available at the moment to integrate the real-time data. The implementations that are realized up to this moment are called 'Player'. Two of these players are presented at this point: The `VideoPlayer` and the `Pachube2LineChartPlayer`.

### 5.7.3 Video

Video in this scenario does not involve much transformation. As described in chapter 4 videos can be regarded as a sequence of images arranged on a time axis. The task is to access each single frame, convert it into a texture and apply it to the real-time layer. However, video can be regarded as an 'acid test', because the content changes at a very high frequency.

The developed `VideoPlayer` class implements the `RealTimeLayer` interface (fig. 5.11). It uses components of the Java Media Framework<sup>1</sup> (JMF) in order to play the video

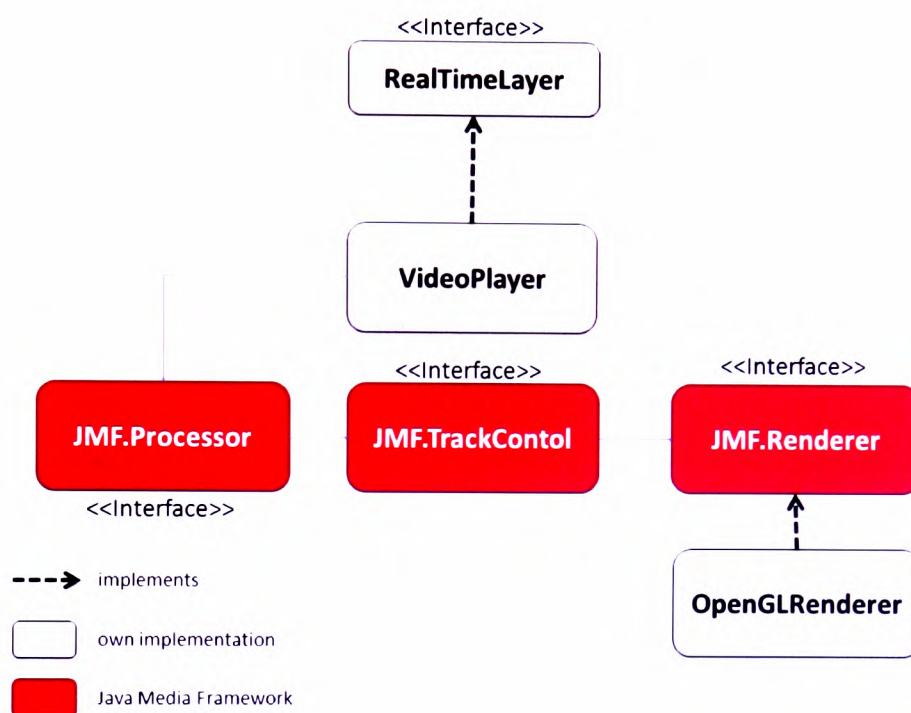
---

<sup>1</sup><http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html> (accessed 13.06.2011)

## 5. PROOF OF CONCEPT

---

and access the video frames that need to be loaded as an OpenGL texture into the graphics memory.



**Figure 5.11: Video-Player implementation** - Video-Player implementation for the real-time layer

Within the JMF there is a processor defined that can load a video file and play it. By assigning specific renderers to the video and/or audio track the content can be processed in particular ways. In the presented prototype the audio track is processed with a standard renderer so it can be heard when the video is visualized inside the façade texture. For the video track a custom renderer is implemented that renders each frame into a `BufferedImage`. The `BufferedImage` is then provided to the 3D viewer by the `getFrame()` function of the `RealTimeLayer` interface. The custom renderer instantly produces a power-of-two texture and flips the image vertically, so it can be directly loaded as a texture by OpenGL.

As the JMF processor runs in a separate thread it produces new real-time data frames independently of the rendering cycles of the 3D scene. The independence of the two

## 5. PROOF OF CONCEPT

---

tasks is achieved by the JMF processor. For other players an additional thread needs to be implemented and controlled separately. The JMF video processor is started by the start/stop functions defined in the RealTimeLayer interface, which forwards the commands to the JMF processor.

### 5.7.4 Metering Data

For providing an example for real-time metering data the prototype implements the class PatchubeToLineChartPlayer. This player requests metering data from the patchube website<sup>1</sup>. This platform enables the publication of own metering data on the web. The data streams can be accessed via a specific URL and can be requested in different formats. The player implementation reads the data from the specific URL for XML output (e.g. <http://api.patchube.com/v2/feeds/XXXX.xml><sup>2</sup>). The metering data is returned as an XML document and the measured values and the according time stamp can be extracted.

The extracted information is stored in the PachubeSet class, which models a container for the information that is provided by the patchube data feed. Further data is requested in predefined time steps and added to the list of received PachubeSets. With this information a chart can be generated, e.g. a line chart visualizing the last 10 received metering data sets. The chart is generated by using the free library JFreeChart. The result of the chart rendering process that is performed by the library is a BufferedImage that holds the chart image. This image is returned as the current frame for the getFrame() function of the RealTimeLayer interface that the Player implements. The refresh rate of the frame in this case depends on the frequency in which the sensor provides new values or in which frequency the patchube platform updates the information. The refresh rate can vary quite significantly for different data feeds.

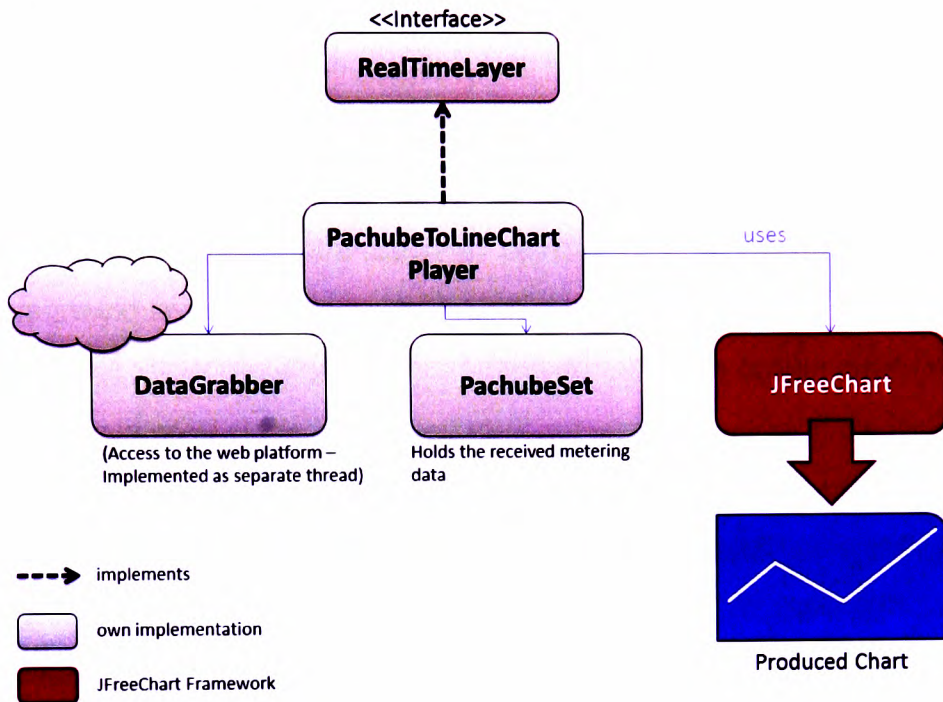
---

<sup>1</sup>[www.patchube.com](http://www.patchube.com) (accessed 13.06.2011)

<sup>2</sup>The XXXX stands for the ID of the specific sensor feed that is published on the platform



## 5. PROOF OF CONCEPT



**Figure 5.12: Pachube-Player implementation** - Real-Time Layer content player for accessing metering data on the Internet

As the platform also enables requesting metering histories, e.g. values from the last 24 hours, it would be possible to integrate much more information and probably a kind of 'metering information window' into the façade textures of 3D buildings.

### 5.7.5 Rendering Real-Time Data embedded in the Façade Texture

The construction shader in this case actually has to fill two null-textures in order to produce foreground and background content for the real-time layer. The changes in order to implement the concept can be integrated into the former 'construction' fragment shader. Uniform variables are used to indicate that a background null-texture is present and the shader needs to decide which content to write to the front- and which to the back texture. The use of the FBO makes this process rather easy. The FBO enables the attachment of more than one render target and rendering into them simultaneously. In this case two texture objects are attached to the FBO as rendering targets, one for

## 5. PROOF OF CONCEPT

---

the foreground and one for the background content. The two texture targets can be accessed inside the fragment shader by using the command `glFragData[x]`, where `x` is the index of the attached render target. Fragment colour values can be assigned to both render targets separately by using this built in function of the fragment processor. Inside the construction shader the algorithm decides if a fragment is situated in a zone of the foreground layers, according to its texture coordinates. If it is located inside a zone the normal colour is applied and the fragment in the back texture is set to black. If the fragment is in none of the foreground layer zones the fragment in the front texture is set to transparent and the colour for the back texture is looked up in the background layers. If it is inside a background-layer zone the appropriate colour is set, otherwise the background-colour is set for the texel in the back texture. This procedure generates a front texture with transparent areas, where the real-time content and the back-texture can be seen. The matching content for these void areas are covered by the background content. The combination of the two would produce the 'normal' façade representation. The only difference is, that the real-time content can be put between them.

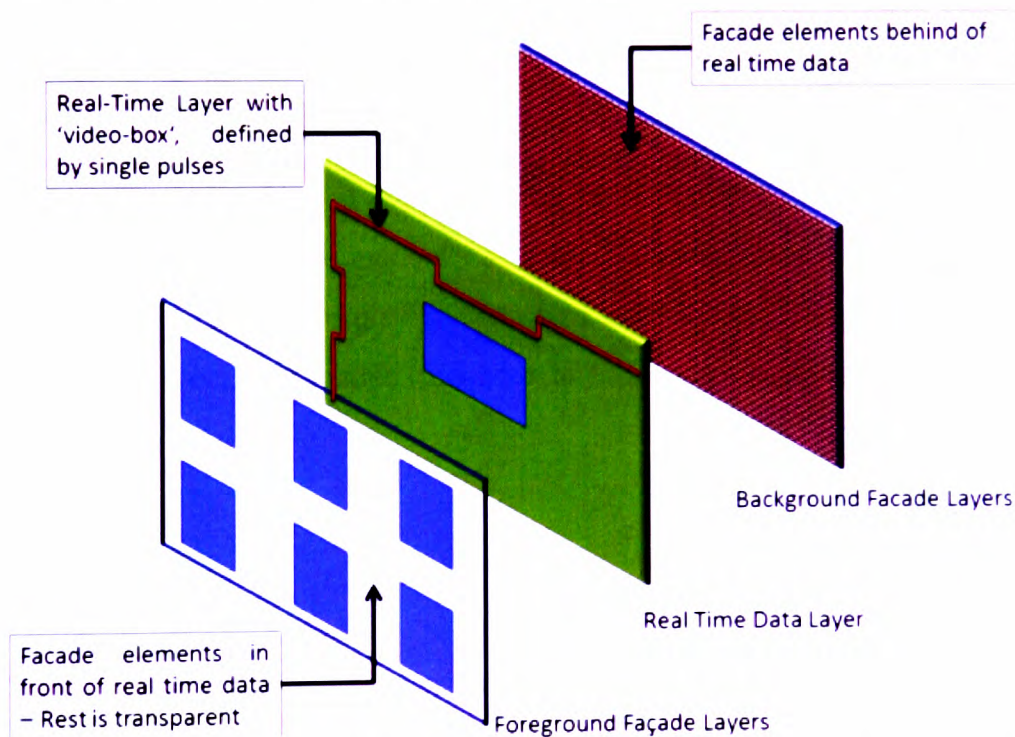
The 'standard' texturing is replaced in case the real-time layer is active. The texturing in the real-time layer mode does not only map one texture onto to the geometry but three textures need to be applied and mixed (multi-texturing). The textures that need to be mixed correspond to the information containers that are defined in case the real-time content is active:

- The front texture - the filled foreground null-texture
- The real-time texture - the real-time layer content
- The back texture - the filled background null-texture

Actually the fragment shader programme for the 'standard' rendering process, which is used for rendering the 3D scene when there is a filled null-texture, is a multi-texturing shader. It solely uses the front-texture when only one image needs to be applied to

## 5. PROOF OF CONCEPT

geometry. This is the case when the geometry is linked to a digital image, which is used as a standard texture. This is also the case when no real-time layer is active and there is only one layer/zone based texture. In the latter case one null-texture is filled and applied to geometry. This single filled null-texture is passed to the front-texture object and applied to geometry. In case the real-time layer is active there are the three aforementioned textures (front, real-time and back). In this case the shader programme will mix them in order to integrate the real-time content with the façade content. Uniform variables are used as flags to tell the shader programme whether a real-time layer is active or not and if the back-texture needs to be used as well.



**Figure 5.13: Real-Time Layer Rendering** - The integration of real-time content into the rendering process

When the three textures are 'active' then the current fragment is filled with the front-texture colour unless it is completely transparent. Texels in the front texture are normally set transparent by the construction shader when the fragment is not part of a zone. In this case it is intended that the real-time content or the background 'shines'

## 5. PROOF OF CONCEPT

---

through the front texture. In case the front texture texel is transparent the shader would check if the fragment is in the real-time layer box. As depicted in figure 5.13 the real-time content does not necessarily cover the whole façade area. The concept of the real-time layer includes a bounding-box rectangle for the real-time content defined by two pulses (see fig. 5.13). In this way the real-time texture can be mapped to a specific part of the façade texture without the need to produce a texture with transparent areas. If the real-time texture would be applied to the whole null-texture as it is provided by the player (no transparencies) the back-texture would always be covered completely, hence it would be rendered useless. In case the fragment is inside the real-time layer bounding box the fragment will receive the corresponding colour of the video frame pixel. When the processed fragment is not in the real-time box then the colour from the background texture is applied.

Using three different textures makes it obvious why the content for the real-time data (e.g. video) can change independently from the façade elements. The null-textures for front- and back texture are only refilled when a change for the layers is requested (e.g. z-order, LoR, etc.). The video (real-time) texture can change independently at the frame-rate of the video.

### 5.8 Summary

In this chapter the implemented prototype was presented that uses the programmable part of the rendering pipeline of modern graphics hardware. It implements the concepts that are presented in the previous chapter and shows the feasibility of the suggested approach for flexible façade texture content. The presented approach for flexible texture generation is based on a two-step process using separate shader programmes for constructing the façade texture and for mapping the generated texture to geometry. As the 'standard' texturing shader is performing the normal task of texturing, the underlying process is relatively simple and well defined. The larger potential for performance

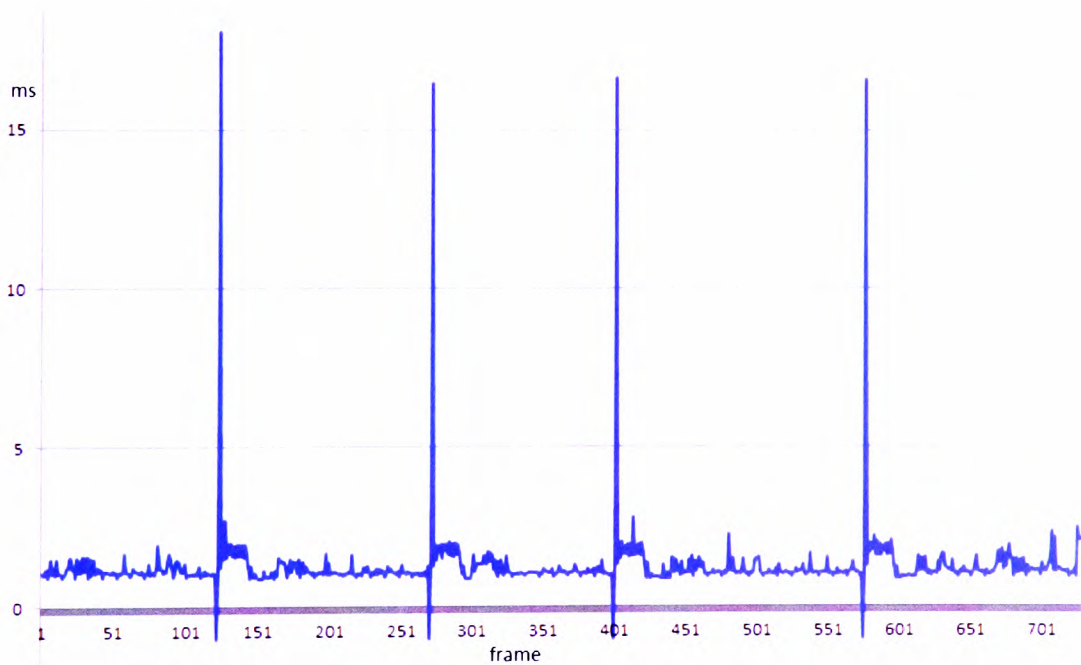
## 5. PROOF OF CONCEPT

---

optimization, by achieving fast content changes, lies in the reconstruction process. The two areas for optimization, which influence each other, are the reconstruction shader itself and the description. The description needs to be designed in a way the shader can optimally benefit from, hence a compact and efficient structure.

### 5.8.1 Performance

The rendering process for the layer/zone-based texture is rather complex and involves a quite significant set of input information. The presented prototype shows the feasibility of the GPU-based approach and handles content changes of the test buildings' façades in quite reasonable time. By measuring the frame time<sup>1</sup> it was possible to find out that loading the atlas- and data texture consumes a lot of time in comparison to the actual reconstruction process.



**Figure 5.14: Prototype Rendering Performance Measurement** - frame time measurement: at each content change a '-1' was added into the measurement set at each re-rendering of the layer/zone-texture

---

<sup>1</sup>The time that is needed to render one frame of the 3D scene

## 5. PROOF OF CONCEPT

---

This problem is solved by a variable management of the two texture objects for the layer/zone-based textures in the 3D scene. Normally the atlas texture, as well as the data texture, is not needed anymore once the façade representation is generated. The two texture objects can be deleted because it is not known if and when the content of the façade is going to change again. The current representation is stored in a texture object that is used by the 'standard' shader for texture mapping. When a content change is triggered the two information textures need to be loaded again for the construction of the new content. As loading of textures consumes a lot of time it is better to hold the atlas- and data texture in graphics memory to be used for further content changes. In that way three textures would be stored for each layer/zone-based texture: the current reconstructed façade texture and the two information textures. A clean-up process could then operate a kind of garbage control that starts if the information textures'<sup>1</sup> data size reaches a certain threshold. Above this threshold the system starts deleting the two information textures for objects that are out of sight or very distant, in order to free memory. Only when objects come closer or return into sight and a content-change occurs, the information textures need to be reloaded.

But the information textures' load process is not the only issue in terms of performance. As the description and the reconstruction process are relatively complex there are two points that can influence performance. Texture look-ups (retrieving a colour value based on given coordinates) are regarded as relatively time consuming and should be minimized, especially on older hardware or on mobile platforms. As the description is encoded into a texture the reconstruction programme needs to perform a number of texture look-ups to read it. Inside the data texture there is information encoded in order to help minimizing texture look-ups (fences, see section 5.5). However, when measuring the time for one frame during the reconstruction process, the difference between using fences or not is relatively small. On other hardware this might be an issue (and also in significantly large city models), but this has not been tested yet. Note that the

---

<sup>1</sup>the data texture and the tile atlas

## 5. PROOF OF CONCEPT

---

reconstruction process is spread over several frames and is not done in a single frame, which might also add to a relatively constant frame time. A further way to speed up texture look-ups might be the use of TextureBuffer-Objects (TBO). These buffers work like 1D textures and are more efficient in terms of look-ups than 2D textures. However, a new encoding strategy needs to be found for the 1D texture buffer. As this feature is relatively new, not all platforms support this concept at the moment and can only be realized for certain hardware environments.

An additional issue on GPUs are conditional if-else branching structures. As the reconstruction process depends on some conditional decisions (e.g. layer switched on/off) the use of if-else-branches is hardly avoidable. On some GPUs the branching can be a bottleneck in some situations, because the hardware is made for highly parallel computations but not for conditional branches, although they are defined by the OpenGL Shading Language (GLSL).

Especially the matters of performance in terms of the proof-of-concept using the GPU involve in depth knowledge of computer graphics principles and specific know-how about GPU hardware. The prototype implementation shows that the concept is feasible but has potential for optimization. However, the optimization of rendering is on the one hand hardware specific and needs to be solved for specific real world projects on particular hardware. On the other hand it is a sub-problem of the presented texturing/visualization concept and there cannot be a final optimized solution provided in this PhD thesis. This issue can definitely be regarded as a research question for related disciplines and is certainly subject for future research.

### 5.8.2 Scalability

In terms of scalability and the possibility to use the presented concept on different platforms and for differently sized 3D city models in diverse scenarios a few thoughts will be discussed at this point. As demonstrated throughout this and the previous chapter

## 5. PROOF OF CONCEPT

---

the concept of layered content supports the flexibility of appearance and texture content and therefore significantly contributes to information visualization and knowledge construction. However, there is also a recognizable benefit in a technical sense when using categorized, layered texture content in terms of scalability and adapting the data size to different use cases and system environments.

For example, if the bandwidth or the client capabilities are restricted in a client-server scenario, only specific content needs to be transferred to the client for visualization. The layers on the server can be categorized according to their importance for the appearance in the given use-case. One example of categorization is the aforementioned LoR concept. When restrictions exist in a given hardware/system scenario (e.g. mobile client), the server can just send a low LoR texture representation to the client, hence a small portion of the overall texture data. This would allow a 'rough' representation of the 3D urban scene. By using an intelligent streaming mechanism additional layers can be added to the façades to increase the LoR step by step and fitting available resources. As layers can also stay in off-mode even though they are loaded, it would be possible to load additional content in the background and the user can then decide if he or she wants to include this information by activating the layer. For very large 3D city models the streaming/loading of specific façade layers can be prioritized according to the distance of objects to the actual viewpoint. One possible solution might be to load LoR1 for all façades in sight, but then load LoR2 and LoR3 for the closest buildings first, before streaming these levels for objects farther in the back. In today's models and systems scalability in terms of texture data size is achieved mainly by different resolutions for the same image texture, streaming low resolutions first before adding higher resolutions for nearby buildings. By using layered content the density of elements can be adapted due to lower or higher number of façade elements. However the actual portion of the content, with lower number of elements, can be observed at full resolution. This is especially true for nearby objects, when the full texture would still be too large to be loaded at once in a reasonable time. When tiles are loaded, concepts like mipmapping



## 5. PROOF OF CONCEPT

---

can still be applied during the rendering process to increase performance by reducing resolution for objects at greater distance. The initial load process and the transmission of content can yet be scaled down by reducing the content (number of elements in the texture) rather than scaling down the resolution of one complete façade texture.

As well as automatic adaption of the layer content in terms of scalability for various platforms, the user can also define the density of content and the combination of different information through appropriate (web-) service interfaces. This aspect is discussed in chapter 4.8.

### 5.8.3 Real-Time Content

Finally, the real-time content is integrated into the façade representation according to the texture content model presented in chapter 4. The additional real-time layer can change its content independently from the façade content. The resulting textures for foreground, real-time and background content produced by the construction shader are rendered on top of each other by the texture-mapping fragment shader. This concept makes it possible to integrate content that changes at a very high frequency and keep the rendering effort to a minimum (façade null-textures only need to be filled when content changes). It would also be worthwhile to investigate if it is possible to integrate video or other real time content to a zone in a layer rather than to the additional real-time layer. On the one hand this would allow adding more than one real-time stream. On the other hand issues about defining which zones are feed from the tile atlas and which zones read their content from other sources would have to be addressed. These investigations, as already mentioned, can be considered as future work.

# Chapter 6

## Pedestrian Navigation using 3D City Models – a Case-Study

In the previous chapters a layer-based model for façade texture content was introduced and a prototype was presented in order to show the technical feasibility of the suggested approach. The use of latest computer graphics capabilities and methods was described in order to achieve a technical solution that provides good performance and great flexibility for façade visualization, which can be changed in real-time during the rendering process. Chapter 2.2 also described (technical) aspects about the environment in which type-6 city models normally exist and how they are integrated into SDIs. The concept for façade content also needs to be able to integrate into this environment (chapter 4.8). It needs to be possible to support workflows and processes that are realized by SDIs in order to introduce the new façade texture concept into the wider context in which 3D city models are used. Otherwise, if the concept would not integrate into the idea of SDIs it would be rather difficult, if not impossible, to establish the new texture content approach for digital 3D urban models.

After presenting the new content model and showing its technical feasibility this chapter is going to discuss the actual impact and effects of the new approach in a use-case scenario in order to provide an example for its qualities in task-driven 3D scenarios.

This case-study is going to show the conceptual benefits of the presented approach, rather than the technical aspects, for a specific scenario in which 3D city models are currently introduced. The discussed use-case is just one example for many possible scenarios a multipurpose 3D city model might need to serve and many other tasks can

## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

---

be solved with the help of the same model. That is why a multipurpose model needs to provide a high degree of flexibility in terms of visual content and visualization styles. The case-study presented in this chapter is going to show how flexible visualization is used to serve a specific scenario.

The use-case, which was chosen for this work is pedestrian navigation. The author of this PhD thesis participated in the 'MoNa3D' project, which investigated the use of 3D city models for mobile pedestrian navigation. As this project provided a scenario where the 3D city model is an integral part of the solution to the given task, this scenario was selected as the case-study for this work.

In this chapter the differences between car navigation and pedestrian navigation will be described in order to define the actual task that needs to be solved. Pedestrians navigate in a different way to car drivers and therefore other requirements exist. The use of 3D models and their benefits in the selected scenario are discussed, especially in regards to landmarks. The chapter also presents details about the MoNa3D project. Part of the technical and conceptual requirements, against which the texture model were tested, are taken from this project. The infrastructure that was defined for the envisioned navigation system as well as the implemented smartphone prototype are also presented. The remaining chapter looks at requirements for pedestrian-navigation and suitable visualization, which were defined in the MoNa3D project and retrieved from relevant literature. It presents possible solutions using the zone/layer-based texturing approach. Looking at these solutions the chapter tries to assess the performance of the texture content model in a real-world scenario and the general benefits for 3D city model visualization.

### 6.1 Concepts of Navigation - Cars compared to Pedestrians

Navigation in general is a very complex task that involves two phases: route finding and the actual process of navigating from A to B. The most well-known tool for way-finding nowadays is the GPS-supported car navigation system. It makes us almost forget that there were paper maps in the past. Car navigation systems support drivers in both of the aforementioned phases: it determines the shortest/fastest route from the starting point to the destination point and directs the driver by appropriately provided commands through the road-network in order to reach the final destination.

Car navigation systems work based on a line graph, which represents the street network. This line graph is used for route finding utilizing algorithms like Dijkstra (Dijkstra, 1959) or A\* (Hart et al., 1968). For route guidance the systems basically work with instructions providing the distance to the next decision point (node in the graph) and which action to perform at this point (e.g. 'turn left'). Additional information like the name of the street that the driver needs to take or a 2D map that provides the spatial context, on which the route is superimposed, helps the driver to take appropriate action and ensures the driver being on the right way. For car-based navigation, this information is basically sufficient and adequate as cars move only on streets, respectively on an edge of the line graph that represents the street network. A car can only move along this edge, forward or backwards, therefore the orientation of the car is given. Instructions like: 'turn left into Baker street', show only little ambiguities because the car is aligned to the road and it is well defined what 'left' means. We will learn later in this chapter that this is rarely true for pedestrian navigation. Pictograms and arrows are also provided to support the driver to choose the correct road when there is more than one possible road in the same direction (see figure 6.1).

Distances are a little more complicated because it is quite hard to estimate a specific distance ('in 200 m . . . turn left') and this is especially true if one is moving and not

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---



**Figure 6.1: Example for decision support in car navigation systems** - Car navigation systems very often provide an additional graphical representation of junctions to support the driver in choosing the correct road (own depiction)

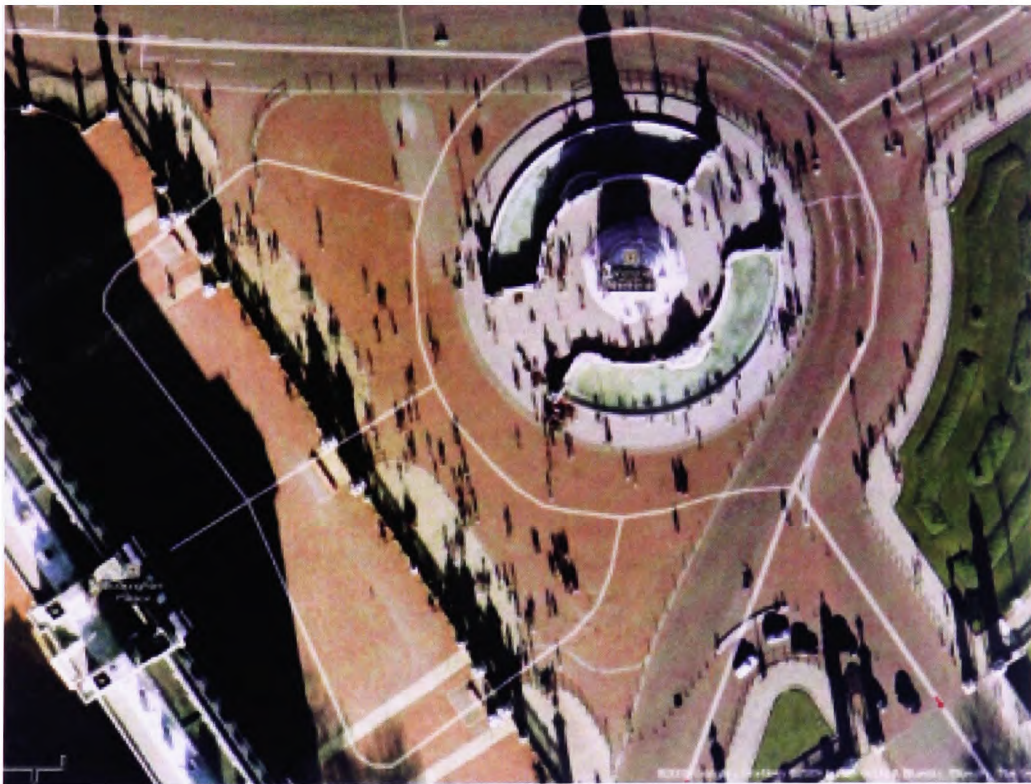
standing still because the relative speed has to be taken into account, which influences the time that is left until the decision point is reached. This estimation is very complicated and requires quite a bit of training. Car navigation systems normally support the driver by repeating spoken instructions at specific points before reaching the decision point (in 250m ... in 100m ... in 50m).

As we can see, car navigation works based on certain assumptions that are specific for driving a vehicle (move on streets, etc.). Using these assumptions, and the 'car-concept' as such, for pedestrians can also guide users to their destination. However, the user experience might be unsatisfactory, because pedestrians tend to navigate differently.

People who walk through the urban environment are not necessarily bound to streets, respectively edges of a graph. Pedestrians have a larger degree of freedom and do seldom align themselves along a street axis naturally. They can also turn their head and look around in contrast to a driver who is mostly looking into the direction in which the car is moving as this is inherent to the process of (safe) driving. Pedestrians normally navigate more or less freely in open spaces like pedestrian zones, squares and junctions (which can be crossed in multiple ways), etc. Restricting pedestrians to a car

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

based navigation concept leads to some unpleasant effects. Users who are walking would need to align themselves to the street axis in order to understand 'turn left'- and 'turn right'- instructions. They might also need to cross open areas like squares in de-tours because current graph based systems approximate open areas by lines connecting one or more 'centre points' with the adjacent streets. Following these pre-defined paths might lead to confusion and inconvenient navigation instructions (figure 6.2).



**Figure 6.2: Lines to Approximate Square** - A square that is approximated by lines, which does not cover all pedestrian options to pass through this area (Bogdahn and Coors, 2009b); Screenshot from Google Maps

Bogdahn and Coors (2009b) suggested that pedestrians navigate in zones of different accessibility. Gaisbauer and Frank (2008) also suggest widening decision points into 'decision scenes' covering the user's vista space. These scenes can be entered and left through portals to other scenes. These concepts, as well as others that can be found in literature, suggest that pedestrians have a different way of navigating urban space. A zone- or scene-based approach better reflects the degree of freedom pedestrians have to navigate. Although a graph based structure can still be the basis for

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

route finding (although concepts for zone based way-finding do exist (Funnel algorithm (Chazelle, 1982), (Kallmann, 2005)), the guidance instructions certainly have to be different to the ones in car navigation systems. In fact, they can also be more complex. Pedestrians travel at a different pace and the cognitive load is much smaller compared to driving a car. More cognitive resources can be used to understand and decode navigation instructions and transfer them to the own situation. Pedestrians actually have the time and the possibility to stop, analyse their spatial surrounding, relate it to their own position and the provided navigation instruction and take the appropriate actions. That means pedestrians have the time to analyse the current 'decision scene' and compensate inaccuracy of system sensors by their own senses. The important aspect is that the spatial context of the 'decision scene' is provided in an intuitive, task-driven manner. In this way the user can match objects in his vista space and the provided spatial context representation and decode the related navigation instruction. Digital urban 3D models are an interesting medium to provide the spatial context as they transport a perspective view of the surrounding in the same dimensionality. However, purely photo-realistic 3D models might be counter-productive in this scenario as well, because when there is a complicated navigation/decision situation in the real world and this situation is visualized photo-realistically by the system in order to provide the spatial context for the decision, the situation is also visually complicated in the 3D model. Therefore a task-driven visualization, which is abstracted and enriched by appropriate information, is important for this use-case. The related visualization aspects will be discussed in a later section of this chapter. At this point another interesting aspect of pedestrian navigation is discussed first, as it can also be related to the use of 3D city models.

Several sources in literature suggest that pedestrians make strong use of landmarks when navigating urban space or when describing routes to others. Street names and distances play a minor role, which is also a big difference to car navigation systems. For pedestrians landmarks are an integral part of the actual navigation instructions. Schroder et al. (2011) conducted an experiment that *'explored how route directions*

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

were formed as the participants traversed the route. In particular, this experiment sought to identify those features used in the descriptions and where and when they were used in relation to the route... exploring factors governing choice, how and why they were referred to within route directions. From this, a classification schema of these features was formed... it was found that the majority of the references were related to features of interest (70%) rather than to streets (30%}'. Similar test done by Elias and Paelke (2008) revealed that it is very likely that most of these 'features of interest' are buildings.

*'Despite the fact that the routes differ significantly in their environment (Route 2 leads through the shopping area in the pedestrian zone, Route 1 leads through a typical residential area and the university campus), in both routes about 50 % of the referenced objects are buildings. The proportions of the other groups vary slightly. It should be kept in mind that these are only preliminary observations, since only two different routes described by twenty people were examined so no statistically significant statement is possible.'* (Elias and Paelke, 2008).

These studies show that the urban surrounding (especially buildings) is both, a part of the actual problem but also an essential part of the solution of the navigation task. Therefore 3D city models are a useful tool for supporting pedestrian navigation as they model urban space and can be adapted to the user's task through scenario dependent visualization. In the pedestrian navigation scenario the emphasis is clearly on landmarks (focusing on building landmarks in this work), hence at this point the work will describe what landmarks actually are and which properties they have in comparison to other buildings.



### 6.2 Landmarks

Landmarks in general are objects that stand out of their surrounding comparing a specific property or a set of properties. These objects show a certain saliency compared to the environment in which they are located. Landmarks can be strong or weak depending on the degree of saliency, hence how much they differ from their neighbouring buildings. Furthermore landmarks are categorized into distant (or 'global') and local landmarks (Lynch, 1960) depending on the spatial extent within the city for which they are valid. For example, Edinburgh Castle can be regarded as a global landmark for the city as it is a very prominent site, which can be seen from many places within the city. Hence, it can be a landmark for many different 'decision scenes' although it is maybe far away. The Eiffel Tower in Paris is also a good example for a global landmark. However, landmarks are not necessarily landmarks because of their height or exposed position. Other attributes can also add to their saliency. Their colour, shape, texture/material and other properties can make them more prominent than their neighbours. When there is a street with red brick buildings, a building painted in blue clearly stands out and can act as a landmark. Another example would be a building with a glass façade in a square where all other buildings have stone or concrete walls.

Besides the visual attributes that are described above, there are also non-visual properties that make buildings or objects salient in comparison to their surroundings. In some cases the function of a building creates saliency. This is sometimes accompanied by visual saliency but in some cases it is complicated to recognize the functional difference of a particular building. In case of a post-office, for example, the saliency is given by the function, because it is likely that the surrounding buildings are not post-offices. As post-offices have a certain corporate identity (signs, colours, etc.) a visual saliency within the decision scene is very likely and these landmarks are rather easy to recognize. Elias and Paelke (2008) found in their study that for downtown area scenarios shops and shop signs are very often used as landmarks as the brands and signs can be easily related to the verbal description/their names (e.g. 'turn left at Macy's').

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---



**Figure 6.3: Kunstmuseum Stuttgart** - The Museum of Arts in Stuttgart is a strong landmark due to its form and its glass façade (Foto: Gonzalez, copyright Kunstmuseum Stuttgart))

Nevertheless, there are other examples where the visual saliency is not supporting the difference in function. For the museum of art in Stuttgart (fig. 6.3) the function is unique in the proximate surrounding. A navigation instruction might integrate it like 'turn right at the museum' and use its function property to explain it to the user. However, if the user does not know that the gallery looks like a glass cube it would be hard to recognize the landmark as a hint for the navigation decision. The information about the form and other visual attributes of the landmark must either be incorporated into the guidance instruction (text, speech, etc.) or as a part of the visual aid that provides the spatial context to the user (pictogram on 2D map, 3D model highlighted, photo image, etc.). Elias and Paelke (2008) investigated different ways how to visualize landmarks appropriately in 2D maps. This research work suggests using a 3D city model to provide

## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

---

the spatial context for the navigation process. Therefore this chapter is going to look at the visualization issues for buildings and landmarks in 3D city models with regards to pedestrian navigation.

Landmarks as an important aspect of pedestrian navigation are one part of the MoNa3D project, which acts as the case-study scenario. In the MoNa3D project 3D city models were investigated in regards to the pedestrian navigation concept that is described in the previous sections of this chapter. The project takes these aspects into account and envisions a pedestrian navigation system that bases its navigation instructions on landmarks that guide the user through the urban environment. This scenario is actually making use of urban characteristics and the urban environment itself to solve a problem. Therefore it is beneficial to use 3D city models to provide the spatial context for the pedestrian navigation scenario and use the presented texture content approach (see chapter 4) to achieve the required and suitable visual appearance. This work therefore also uses requirements defined in the MoNa3D project to conduct this case-study and to test the texture concept. The MoNa3D project is presented in the next section in more detail.

### **6.3 The MoNa3D Project**

The project 'Mobile Navigation with 3D City Models' (MoNa3D), in which the author of this work participated, investigated the aforementioned aspects of mobile pedestrian navigation. Two universities of applied sciences participated in this project as well as several industrial partners from the navigation industry. Pedestrian navigation was identified as a potential new market for this industry sector, not only on dedicated devices (like most car systems), but also on personal digital assistants (PDA) and especially on smartphones. However, the project did especially investigate 3D city models and their potential use for pedestrian navigation, integrating 3D-landmarks into 'semantic route descriptions'.

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

Coors and Zipf (2007) define:

*'The aim of the project . . . is to assess and develop the infrastructure, tools and methods for developing a mobile navigation system that enables the user to navigate in a 3D urban environment on mobile device (smartphone, PDA etc.). It provides navigation support through semantic route descriptions, using 3D landmarks. In order to achieve sustainable project outcomes, 3D city models for navigation support have to be available within a functioning 3D spatial data infrastructure (<http://www.3d-gdi.de>). Further techniques need to be developed for creating storage efficient synthetic textures for 3D city models optimized for mobile devices. Thus the two main goals of the project are:*

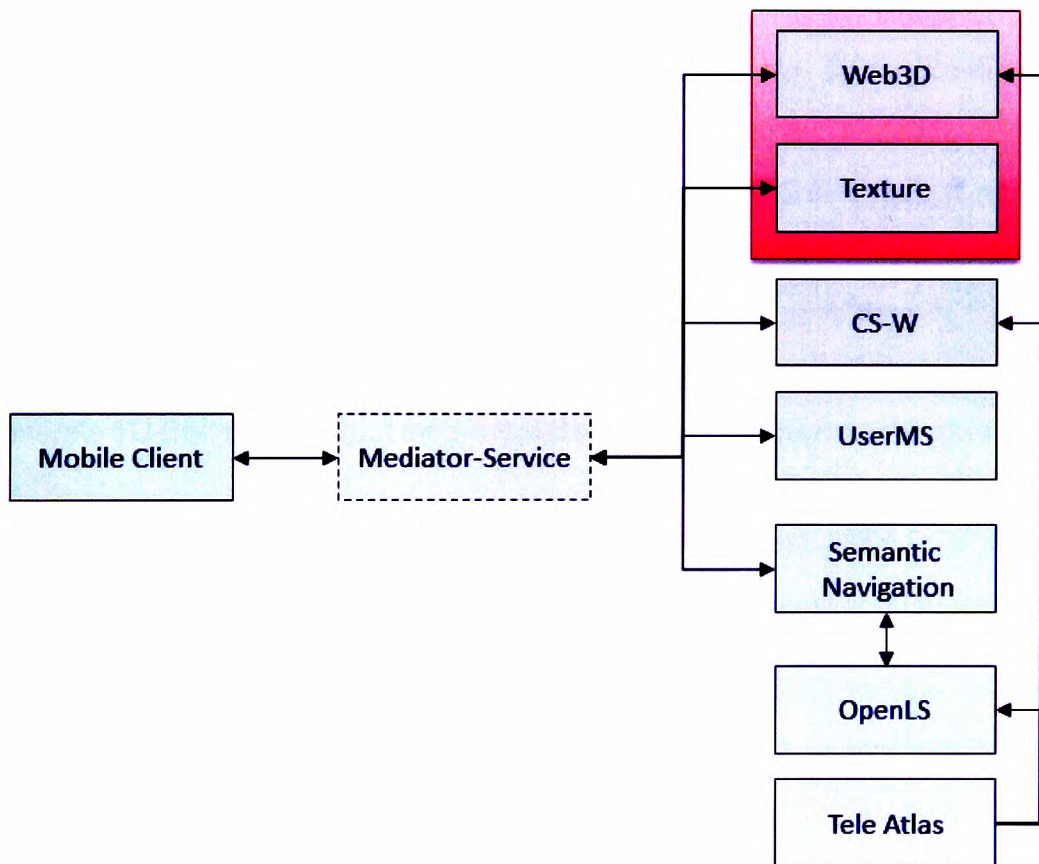
- To provide a cognitive semantic route description by using landmarks in 3D, which allow context dependent personalized navigation support*
- To develop an approach to create suitable non-photorealistic building textures using image processing methods and synthetic textures along with a corresponding compression system for an efficient storage and transfer of 3D building models.'*

In this definition we can find several requirements of the project related to the texturing of the model, which can be linked back to points that were discussed in this thesis as well.

First of all the envisioned navigation system should be based on 3D city models, *'which have to be available within a functioning 3D spatial data infrastructure'* (Coors and Zipf, 2007). This requirement links back to the aspects and investigations in chapter 4.8 and shows that in this particular scenario the 3D model needs to be flexibly accessed and is also part of the SDI process. This demonstrates its character of being a part of the problem solving process rather than a fixed, pre-defined visual background for other information.

### 6.3.1 The MoNa3D Spatial Data Infrastructure

The envisioned server-side infrastructure of the system is based on several OGC conform web-services and additional custom services, which interact in order to provide route instructions and the appropriately styled city model to support users during navigation. The structure of the system architecture is depicted in figure 6.4.



**Figure 6.4: MoNa3D Architecture** - The schema of services and components in the MoNa3D project

The back-end of the system has two major tasks to fulfil: 1) Calculating the route and find local and global landmarks that are an adequate support for the particular route and which can be incorporated into the navigation instructions. 2) Produce an adequate 3D model with appropriately visualized and information-enriched landmarks, hence a suitably styled 3D model supporting the user in the way-finding task. The ac-

## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

---

tual server side process is coordinated by the so called 'Mediator Service', which links the mobile client to the 3D-SDI on server side. The foremost task of the Mediator is to implement the logic of the server side process, querying specific information from one service that is needed as input for another service. The second task of the Mediator is to finally translate the 3D city model and the resulting route instructions into a format that is required by the particular mobile device. This transformation is necessary because in 3D-SDIs often XML formats and VRML/X3D is used, whereas on mobile devices very often formats like MPEG4 and others are in use. To provide the resulting information of the server side process in an adequate form to the mobile clients is the task of the Mediator. Another reason for the Mediator Service is to provide one contact point to the SDI for mobile clients, as the workload for the client itself, accessing the separate SDI services and buffering their results in order to send them to the next service, would be too high. This task is delegated to the Mediator. The actual process to generate 3D-navigation output for a requested route is structured as follows:

- the OpenLS service is queried by the Mediator for the route from A to B and receives a route in the classical sense, a polyline and textual route instructions.
- this route description is sent to the Semantic Navigation service, which transforms the instructions and adds information. It retrieves suitable landmarks (from a repository of local and global landmarks) and extends the route instructions accordingly.
- as the Mediator has the semantic route description at hand (IDs of landmark, instructions and route polyline) it uses the W3DS interface to query the 3D model for the relevant area around the route and uses the IDs of the landmarks to accentuate the 3D model using layer/zone-based textures. As the service is aware of buildings that were identified as landmarks (the IDs are provided by the SemanticRouteService) these buildings can be visualized in a special way to support the user in identifying them in the real world and relate them to the navigation instruc-

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

tions. The W3DS and the 'Texture'-component in the MoNa3D architecture (see fig. 6.4) can be realized in the same 3D management system.

- The 3D model (which now has the suitable appearance) and the route instructions need to be transmitted to the client. Optionally this information is transformed into a suitable client format by the Mediator and/or compressed for efficient transmission.

As one can see the 3D city model is embedded into a quite complex infrastructure in this scenario and needs to be able to provide information in a suitable visual appearance and for the required spatial extent. These aspects and requirements reflect issues discussed in chapter 4.8, especially when it comes to appropriate interfaces. In this scenario landmarks and the surrounding buildings need to be visualized in a different way, according to the IDs provided by the Semantic Navigation Service. This requires a certain level of flexibility when setting the appearance of the objects, which can be achieved by zone/layer-based textures. However, it also requires having appropriate interfaces for the W3DS service to style particular building objects in a specific way, not to mention that this is only possible when objects are identifiable as separate entities. Hence, this is only feasible with type-6 or type-3 models and certainly not with type-1 (see chapter 2.2.1).

### 6.3.2 Storage Efficient Synthetic Textures

Another requirement defined by Coors and Zipf (2007) for the project is the creation of storage efficient synthetic textures. These textures should be optimized for mobile devices and efficient in terms of transfer of 3D city models as well.

This requirement can be met by the presented texture content model, because zones can be filled by texture tiles. These tiles are just small portions of the overall façade texture representing the visual content for a specific façade element. This texture portion

## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

---

can be reused for all zones representing this particular element. Therefore the overall amount of texture data is smaller than a normal image texture of the façade. This is beneficial in terms of storage, as tiles might be also shared among many façades, if necessary. In chapter 4.8 several scenarios were discussed, with different points for the texture content reconstruction to happen. In certain scenarios the tile-based approach is also helpful in terms of data transmission. When the reconstruction process is actually conducted on the client only the texture tiles need to be transmitted, not complete textures. This is much more efficient.

In terms of client memory consumption the client side reconstruction creates complete textures before rendering and memory consumption is comparable to normal texturing, unless only textures of buildings in the view of the user are reconstructed and the rest of the textures remain in tiled form. The same is true for the reconstruction of the texture content during the rendering process when using shaders. The two level rendering approach presented in chapter 5.7.1 creates normal texture objects in the graphics memory, which consumes the same amount of memory as normal textures. This can be handled by only reconstructing the textures of buildings that are in the current view of the user. However, there are promising approaches presented by Haegler et al. (2010) and Fabritius et al. (2008), which show that it is possible to generate textures in real-time for each rendered frame. In this way the texture objects for the façades are not permanently present, because the content of the texture is generated for each frame. This consumes less graphic memory. Investigating this issue and developing the prototype presented in chapter 5 further into this direction would be very interesting and a good point for future work.



### **6.3.3 The Smartphone Client Prototype**

During the MoNa3D project a mobile navigation prototype on the basis of a smartphone was developed. At the time the project started no smartphones were on the market providing a programmable rendering pipeline, thus the project team decided to develop a scenario 2 solution (chapter 4.8.3), where 'raw' texture content information (layers/zones/tiles) is sent to the client. The client constructs a complete façade texture (in the application) and uses them for 'normal' texture mapping, in absence of the possibility to use shaders for reconstruction. The shader-based prototype was developed in parallel on another hardware platform in order to assess the feasibility and the flexibility that can be achieved realizing the reconstruction in the programmable rendering pipeline.

However, the realized smartphone prototype still uses local 3D models and local zone/layer textures as the suitable service interfaces for the W3DS in order to expose the new texture capabilities are not yet available. Locally the prototype could show that the layer/zone approach is feasible on a smartphone in terms of rendering and performance and it also showed that existing 3D data sets (provided by an industry partner) that are used by current navigation systems can be used for the zone/layer-texturing approach. Although texture tiles are defined slightly different in those data sets and used in order to reconstruct realistic representations of buildings. The smartphone prototype also integrated navigation hints (explained in chapter 6.4.2) into the visualization of the aforementioned data set for an exemplary pre-defined route to illustrate the landmark-based navigation concept.

### **6.3.4 Summary**

All in all the MoNa3D project showed that 3D-SDI integration of the new texture model is crucial in this use-case in terms of flexibility and in terms of appropriately designed interfaces, so that other systems can query the type of model they require. The project

## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

---

scenario also demanded storage and transmission efficient texture information in terms of data size, which can be achieved by the tile-based approach of the presented content model (chapter 4). The implemented smartphone prototype showed the potential of the scenario 2 solution (chapter 4.8.3) in a real world use-case and also provided initial results for navigation hint integration (see chapter 6.4.2) on a mobile device.

### **6.4 Visualization of Buildings and Landmarks for Pedestrian Navigation**

After the overview of the MoNa3D project this chapter is going to discuss pedestrian navigation specific requirements for 3D city model and landmark visualization in more detail. The conditions that are under investigation in the following sections were specified in the MoNa3D project but can also be found in relevant literature. Basically, looking at the presented pedestrian navigation concept that is also described in relevant work of other researchers (e.g. the use of landmarks, shop signs, etc.) one can deduce requirements that can be looked at in terms of zone/layer-based texture content. In later sections several of these requirements are examined and it is assessed how the new texture approach can address specific issues. At the end of the chapter the results are analysed and an attempt is made to relate the findings to the wider field of 3D model visualization.

#### **6.4.1 Level of Realism**

The complexity of certain 'decision scenes' especially in downtown areas can be very high. In specific situations many buildings, shops, street signs, pedestrian lights, etc. can be inside the vista space and make it rather difficult to recognize a specific landmark or a specific exit of the decision scene. If there are also many people around and

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

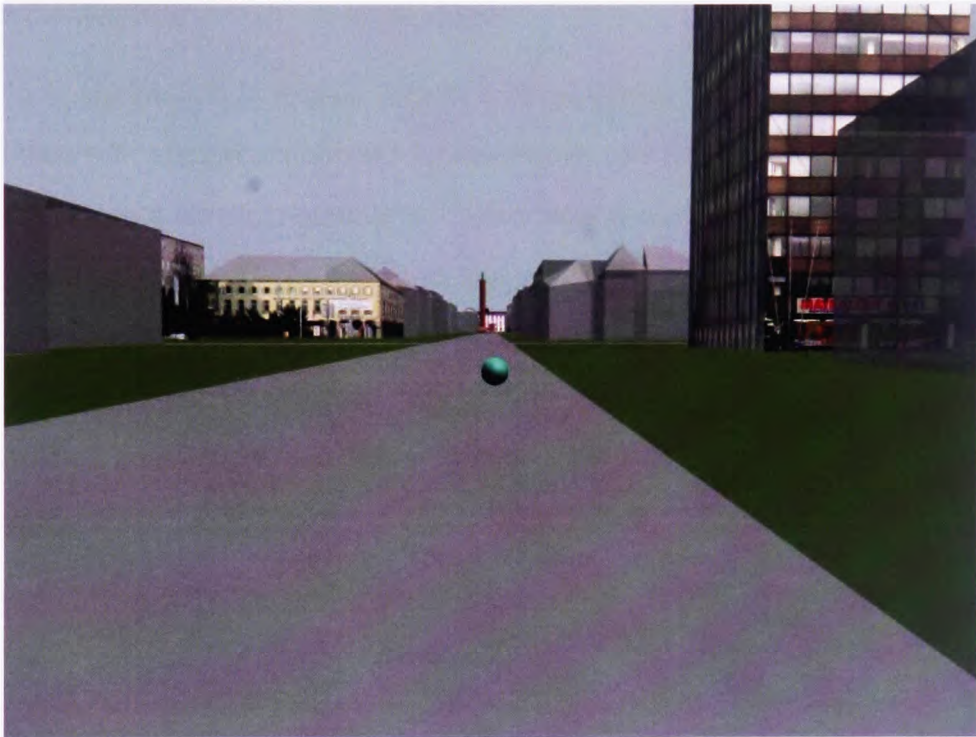
a lot of traffic on the roads this might also influence the ability to identify a certain landmark and to understand/decode the navigation instruction. One could think of a place like Piccadilly Circus in London compared to a small junction in a suburban area where the complexity is a lot smaller and navigation decisions would be easier to make. Assuming a situation like the one present at Piccadilly Circus, where many possible exits and lots of shops, signs and other objects are present (e.g. the fountain in the middle blocking parts of vista space and occluding possible exits), a 3D model can help to understand the spatial context in which the user is situated. However, a photo-realistic model cannot support the user appropriately because the visual complexity would be at the same level as in the real world scene, without pedestrians and traffic, of course. Superimposing a polyline representing the route, like in the car navigation approach, might help but it can still be considered to be a complex and complicated situation, because the spatial context in which the route is embedded is still as complex as the real world scene. Furthermore, it was already discussed that guidance along polylines results in various problems for pedestrians.

The ability to adapt the level-of-detail or more precisely, the level of realism, for the 3D spatial context is crucial in this situation. Reducing the visual complexity of the 3D content in the representation of surrounding buildings can help to identify landmarks by accentuating them. This effect can be achieved by using the LoR parameter of the texture layer (chapter 4.6.2) in order to adjust the detail for façade textures for specific buildings or building groups. Visualizing landmarks with higher detail and surrounding buildings with lower detail emphasizes the landmark but still provides the surrounding to which it can be related. The change of detail (decreasing detail) in the 3D model can also emphasize certain characteristics of landmarks and other buildings, e.g. their form or the material of the walls, etc. By reducing the visual complexity the position of landmarks can also be emphasized more, because users concentrate more on the spatial relation to neighbouring buildings and they might concentrate more on this spatial aspect in the real world scene as well.

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

Kray et al. (2003) used different detail for their landmark visualization in a 3D city model environment by visualizing landmarks with textures and other buildings as semi-transparent blocks in order to highlight the objects that are relevant for navigation (figure 6.5).



**Figure 6.5: Emphasizing landmarks in 3D Scene** - Navigational landmarks are visualized in detail with textured models to attract the user's focus while buildings with less dominance are shown in grey-scale and semi-transparent rendering style (Kray et al., 2003)

Using the layer-based approach defined in this work for façade textures, it is possible to adjust the appearance of the urban surrounding in a more fine-grained way (compare figure 4.8). It is possible to adjust the number of elements and to add relevant content. The landmark can also include a particular shop sign, for example. This sign can be added to the façade texture as a separate zone in a separate layer. Hereby the façade texture includes only one shop sign, the one that is mentioned in the verbal/textual instruction, although the real world façade might include many more signs or advertising bills. The zone/layer-based approach allows including only relevant information into the digital 3D representation and specifically omitting other content to support the user in

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

his task. This example shows that in the pedestrian navigation use-case one research objective of this work plays a significant role: the adjustability of texture content. As the photo-realistic model is too complex to support the user in the given spatial context, the appearance of the model needs to be adjusted (decrease visual complexity), which reflects the aforementioned research objective.

In terms of abstraction (a related aspect to level-of-realism) it is also possible to use texture tiles with abstracted content for landmarks and context buildings. In case the landmark's texture consists of material-, window and door layers, it would be possible to use abstracted tiles for the windows and doors, hence further reducing the object to its general properties and therefore reducing the scenes visual complexity. Döllner and Buchholz (2005) investigated non-photorealistic visualization of 3D city models and identified abstraction as a useful feature for building objects in 'map'- scenarios (figure 6.6).



**Figure 6.6: Sketch Rendering** - Sketch-rendered buildings through Edge-Enhancement approach (Döllner and Buchholz (2005))

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

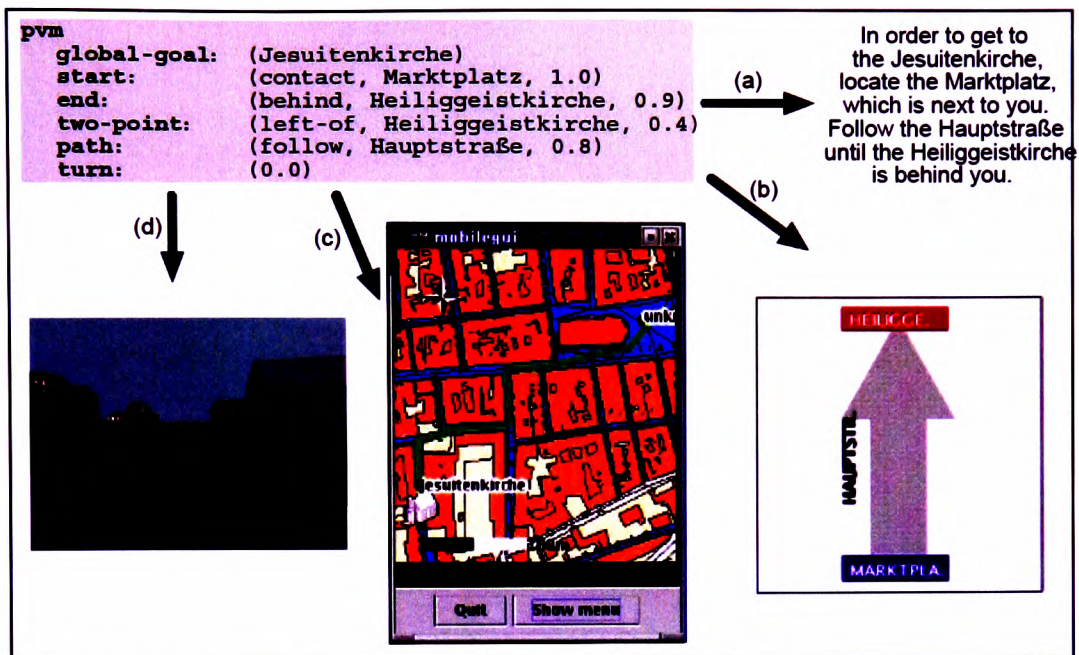
This effect can be achieved with the layer/zone-based texture as well. As the zones can be filled with any texture tile content they do not necessarily need to use realistic 'window'-tiles (how to fill zones; see chapter 4.6.4.1). Zones can also be filled with sketch-like representations of the related façade element and therefore generate a different representation than the photo-realistic one. This is one property, besides the layer capabilities, that adds to the flexibility of the new texture content concept. The sketch rendered façade can also be combined with relevant information (e.g. shop signs, etc.) modelled in additional layers.

### 6.4.2 Navigation Hints

Navigation instructions or route descriptions of pedestrians for other persons are very often based on landmarks rather than street names (Schroder et al., 2011). Guidance instructions are often linked to prominent buildings or objects. As Kray et al. (2003) pointed out there are several elements of a navigation instruction that form the *'pre-verbal message'*. This message can be turned into a verbal, textual or graphical representation. In terms of a 3D city model and pedestrian navigation it would also include landmarks (called *'anchor points'* in Kray et al. (2003)) or their IDs.

This means that the *'pre-verbal message'* can be turned into a spoken/written message, a 2D map with pictograms or images of landmarks, or it needs to be turned into an appropriate 3D visualization. Assuming a message like 'Pass between the two university towers', the 3D model representation should translate this message into a suitable visual representation. Recently developed car navigation systems also include 3D models of single buildings into the context visualization, however, the model just acts as a kind of base map. Instructions still have the form 'Turn right into A street'. The 3D model just delivers the three-dimensional spatial context for this instruction, but is not part of it.

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY



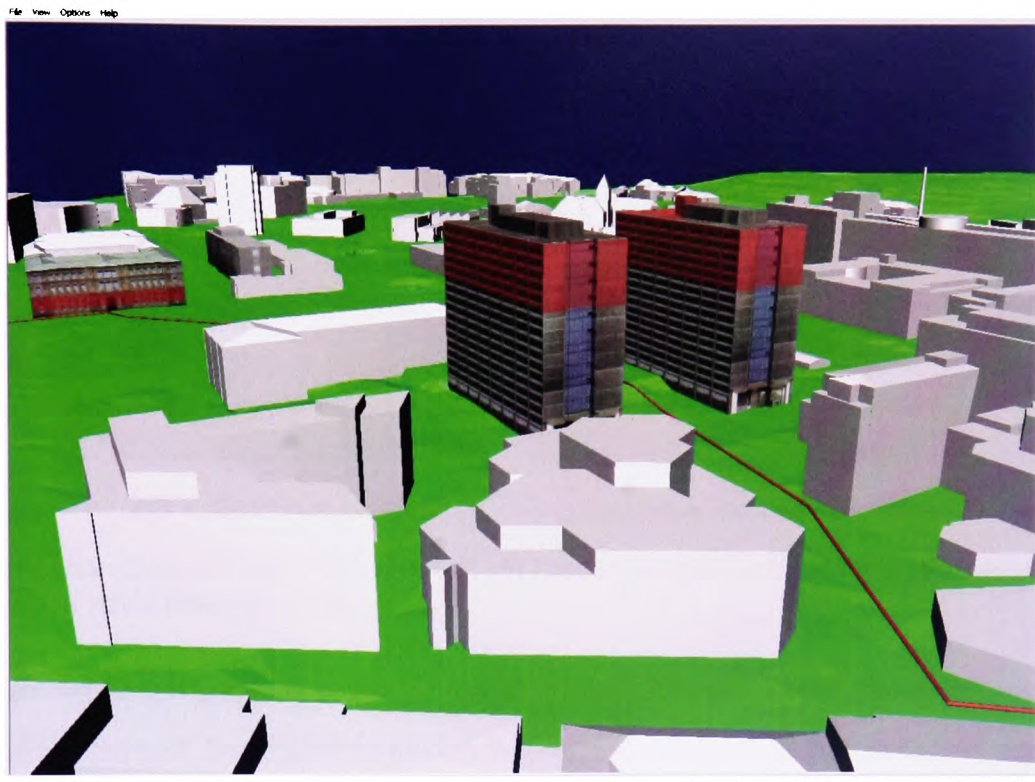
**Figure 6.7: The Pre-verbal Message** - A pre-verbal message and four ways to present it to the user. (Kray et al. (2003))

Looking at the pedestrian instruction beforehand, a movement instruction ('Pass between') is linked directly to an object in the urban environment ('The university towers'). Using the flexible content for textures this relation can also be represented in the 3D visualization (fig. 6.8).

By adding additional layers with thematic information or with additional graphical elements into the façade textures it is possible to highlight landmark buildings as well as integrating direction instructions directly into the relevant building's appearance. This was also achieved on the mobile navigation smartphone prototype for an exemplary route and showed the feasibility on mobile clients.

In figure 6.9a) a link between the landmark and the guidance instruction is only expressed through the textual/written instruction. A graphical representation is not included into the landmark. The material property ('yellow house') needs to be used to identify the building in the real world and to relate the guidance instruction to it. In figure 6.9b) the red arrow is added into the layer-based façade content dynamically after route calculation.

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY



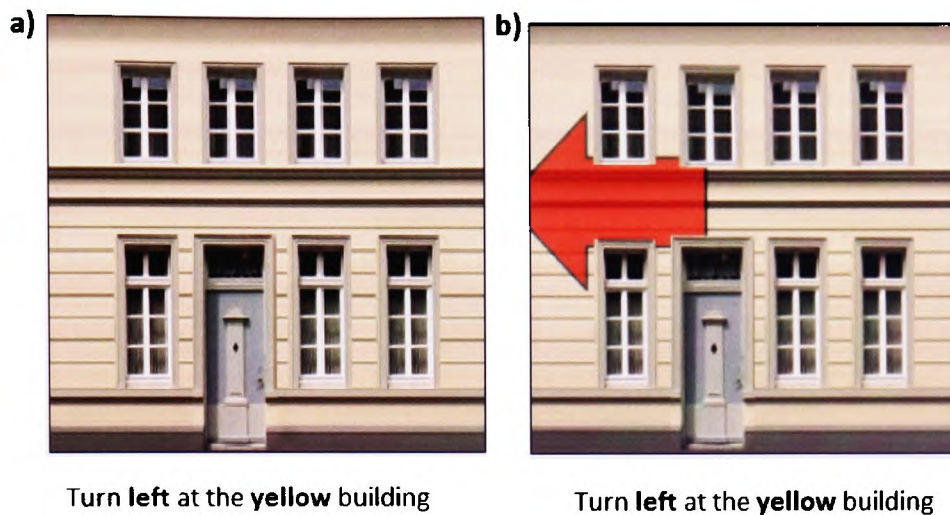
**Figure 6.8: Visual Hints for Landmarks** - The University tower buildings are highlighted as they act as landmarks for the particular route

In this case the landmark based instruction is provided by text and is also included into the 3D visualization. But the navigation instruction is not only integrated 'somewhere' in the 3D scene (the arrow could also be placed at the centre axis of the road, etc.), it is integrated into the related landmark that is part of the verbal/textual instruction. With the integrated navigation hint the landmark is also highlighted, because other less relevant buildings do not include hints, and therefore the landmark is easier to recognize. Furthermore, the link between the navigation instruction and the landmark, on which it is based, can be better represented in this way.

Besides the aspect that the connection between the instruction and the landmark can be visually maintained by integrating the appropriate hint directly into the landmark texture, there is also another benefit that should be taken into account. In the actual pedestrian navigation process the guidance instruction should be based on landmarks. The user would therefore look at the screen with the 3D representation and read the



## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY



**Figure 6.9: Navigation Hints in Façade** - A building façade a) without navigation hint and b) with a visual navigation hint

written instruction and look at the 3D model to find out how the landmark looks like and where it is located. To determine the landmarks position the user needs the information about the landmark itself (its shape, material, etc.) and its direct neighbours. The neighbouring buildings, the landmarks spatial context, are used to verify one's match of the digital model and the real world scene. Visualizing solely the landmark would not be as efficient. In addition, the landmark's close spatial surrounding needs to be provided to the user as he will base further locomotion on the visualization of the surrounding. For example, if the instruction is: 'turn right at the yellow building', the area on the right side of building should be visible in order to support the user in his navigation decisions. In figure 6.10 it can be seen that there are visualization approaches that occlude parts of the spatial context, which is counter-productive especially on small displays (e.g. of mobile devices).

Integrating hints and additional information directly into the landmark's façade texture, or more generally into its visual representation, by an additional layer this effect can be avoided. This aspect is very important for the pedestrian navigation scenario as the spatial context plays a significant role in this case.

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY



**Figure 6.10: Bubbles for linking nav-instruction to landmark** - Bubbles are used to provide a link between landmark and instruction but they occlude parts of the spatial context

Another benefit of the presented texturing concept regarding navigation hints is the ability to change content in real-time (in the suitable scenario (see chapter 4.8.4) when the texture description is present on the client.

This is also beneficial for the navigation scenario and the related landmark visualization. When a route needs to be recalculated for some reason (e.g. change of destination point), parts of the 3D model might need to be loaded in addition to the model that is already present on the client. However, the part of the model that is already present needs to change its appearance, because the route changed and other buildings act as landmarks for the modified route. In case of flexible façade textures not the whole model needs to be reloaded, or new image textures need to be requested. In case of layer-based texture content it is just sufficient to switch relevant layers off/on (e.g. decrease detail for non-landmark buildings) and to change navigation hints by removing/adding 'hint' layers to the appropriate buildings. By designing a smart update process only relevant parts of the model and the textures need to be added/replaced, which increases response times of the client-server system and increases performance. By developing the concept further and by implementing smarter

## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

---

update algorithms it might be possible to load single layers from the server and merge them with the zones/layers in a texture content description that is already present on the client. Hereby only relevant parts need to be queried from the server side.

### **6.4.3 Shop Signs**

Elias and Paelke (2008) identified shops and brand names or logos as important landmarks in downtown areas, as they are well known and the logos are easily recognized and related to the brand. Logos and pictograms are also a solution for landmark visualization for digital 2D maps presented in Elias and Paelke (2008). This kind of information, which appears to be very useful for pedestrian navigation, can also be included into the 3D visualization of the spatial context. Using the presented texturing approach also adds more flexibility in terms of adding and placing logos and signs.

The logo or a sign of a shop in current 3D city models are just a part of the texture, the real world photo-image, that is applied to the wall geometry. The sign or logo is represented by a number of unrelated pixels having the appropriate colour values to represent the sign. However, the sign or logo as such is no separate entity and it cannot be separated from the rest of the wall texture. Using the layer-based texture concept the shop sign can be a separate zone in a separate layer, which can be used independently from the rest of the content. As already mentioned in the section on Level-of-Realism for landmarks, it is possible in this way to reduce the detail of the façade to a lower level (still enough detail to recognize the real world building) and still use the logo or shop sign as a visual hint for the landmark. Another advantage of the zone/layer approach is that the brand's logo can be put anywhere on the façade by changing the zones coordinates or place it on multiple façades. In this way the 'shop'-landmark can be recognized by the user from multiple directions, even though the sign in the real world is not visible from the user's position. Using a 3D model with realistic photo-textures this effect could not be achieved.

## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

---

A further advantage of the zone/layer-approach in terms of signs and logos of shops is not only beneficial for the pedestrian navigation scenario: The texture tile that represents the logo of the shop can be updated independently from the rest of the façade. The image information for the tile can be changed by replacing the old logo with the new one, without the need to change the rest of the façade content (e.g. taking a new façade picture). The other layers and tiles can stay the same as they represent elements that have not changed. In case of normal façade textures the image would have to be edited manually or be replaced completely, which would make it necessary to take a new picture of the building's façade. For the tile-based approach it might not even be necessary to take a new photo of the building. For very well-known brands the logo could come from other sources, like the Internet, or the company's PR department. Another way would be to extract logos from pictures from a photo-website like 'flickr', etc. (complying with copyrights). As the texture tile information can come from very different sources and does not necessarily require sending someone out to make a picture of the relevant building, the update process for content that changes relatively frequently is made easier and less complicated.

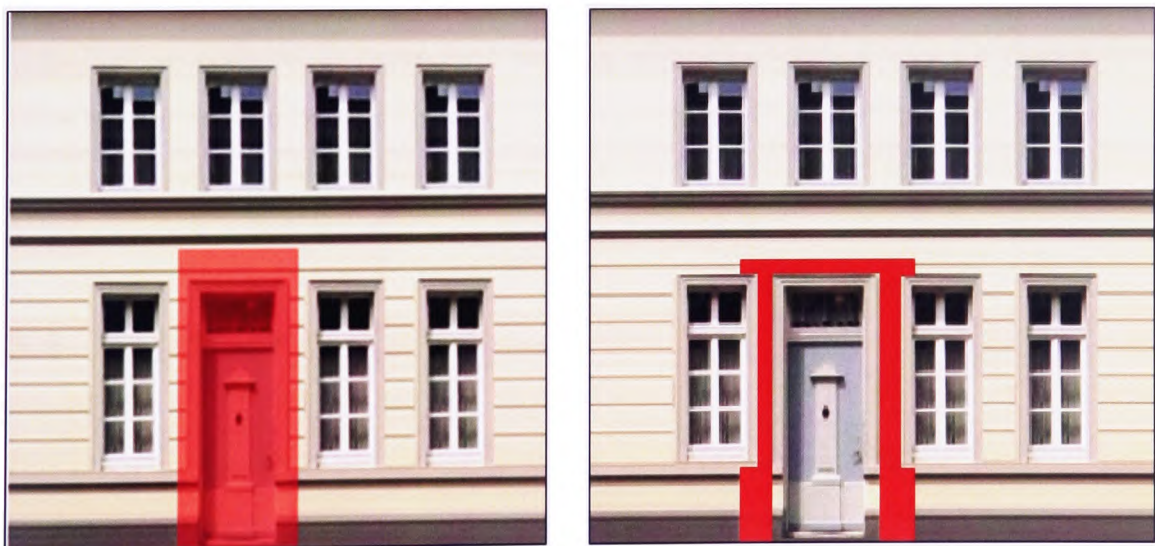
### **6.4.4 Link to Indoor Navigation**

A quite interesting aspect of pedestrian navigation that is a current topic in this field is the link between indoor and outdoor navigation (Gartner (2004), Rehrl et al. (2005)). For example, for large and complex buildings it is relevant to know, which entrance needs to be used or on which floor and in which part of the building a certain company or shop is located. Using the correct entrance can be relevant for the indoor navigation because there might be no direct connection inside the building to reach the destination when using the wrong entrance. In this case the indoor navigation would need to send the user back onto the street to go to the correct entrance. Even in a scenario where no indoor navigation in the destination building is provided it would be useful to have some information about the destination building (in this case-study buildings are in

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

focus as destination objects) rather than the simple 'you have reached your destination' output. For pedestrians the aforementioned aspects of choosing the correct entrance of a building or to know on which floor a specific business is located are quite important. Assuming that navigation data sets for pedestrian navigation systems will store this kind of information in the future to link indoor and outdoor navigation or to provide additional information for destination points, appropriate visualization methods for LoD2 building models need to be found.

Highlighting the correct entrance of a building or the floor of a building creates the same effect as the navigation hints that are integrated into the façade texture: they represent the navigation instruction visually in direct relation to the building. Highlighting the appropriate elements in the façade can only be achieved when these elements are modelled as explicit objects in geometry, so that they are identifiable as separate entities. This is not the case for 3D city models when they are modelled in LoD2, for example. In this LoD buildings do not model windows, doors, or building floors geometrically and in a standard façade texture these objects cannot be identified as separate objects. Hence, it is not possible to identify and highlight them.



**Figure 6.11: Highlights for façade elements** - The doors are highlighted in different ways. Left: Classical overlay with transparent box; Right: Box in separate layer with higher z-order behind windows and door

## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

---

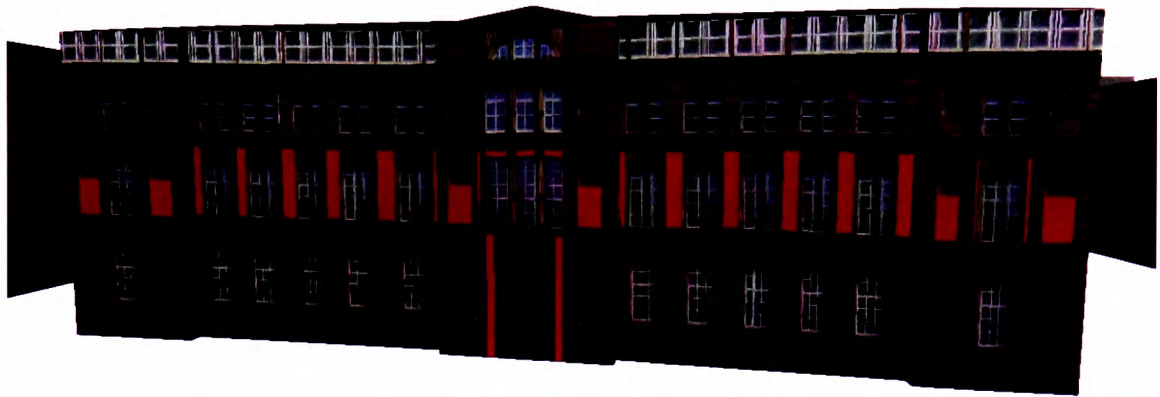
In the zone/layer-concept the window and door entities, besides other objects, are modelled separately and managed in different layers. Using appropriate acquisition methods in the pre-processing it is also possible to generate separate window rows for 1st-, 2nd-, 3rd-, etc. floor. This makes it possible to distinguish building floors. Information about certain shops or businesses, etc. can be linked by an ID to the according 'door'-zone of the façade description. That would allow linking the final destination (inside the building, e.g. a business) to the appropriate façade element (an entrance). A data set that stores on which floor a company or store is located, would allow relating this information to the appropriate window layer. Having this information at hand it would be possible to integrate additional coloured zones into the façade texture content by adding additional 'highlighting layers'. Highlighting can be done in different ways knowing the position of the actual element that needs to be emphasized. Using a traditional texture the 'box' to highlight the e.g. door would need to be overlaid onto the standard texture and a certain transparency needs to be applied so the element can shine through (see fig. 6.11, left). In a layer-based texture the 'box' can be added in the suitable z-level, so the additional visual element is behind the windows and doors (see fig. 6.11, right). This allows giving additional information the appropriate position in the texture content. In figure 6.12 the use of these additional layers is depicted to show the entrance and the first floor of building 1 of HFT Stuttgart (University of Applied Sciences) where the student office is located.

### **6.5 Summary**

This chapter described a case-study for the zone/layer-based texture approach presented in chapter 4 for the field of mobile pedestrian navigation. Certain requirements of the MoNa3D project, which investigated the use of 3D city models in this field, as well as from literature, were investigated according to possible visualization solutions using the new texture concept. The case-study also presented the integration into a project specific 3D-SDI.

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---



**Figure 6.12: Navigation hint for correct entrance and floor of a building** - Appropriate colour-coded elements in order to highlight entrances and building floors provide additional information about the destination building, e.g. to find a specific office

The MoNa3D project defined two specific requirements in terms of textures that were examined in this chapter. The first one was the development of storage and transmission efficient synthetic textures for use on mobile devices. The second was the setup of a flexible 3D-SDI, on which the pedestrian navigation system should be based and in which the textures should be integrated. Both of these requirements were met by the presented layer/zone-based textures. Storage and transmission efficiency is achieved by the texture-tile approach reducing the texture data size significantly. The flexible integration of the new texture concept and its use in several SDI client-server scenarios were already discussed in chapter 4.8.

Several other scenario specific demands defined by the project and in literature were discussed in terms of flexible texturing. These demands were also met by the texture concept and can provide an adequate visualization in task-driven scenarios.

## 6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY

---

The chapter underlines three aspects of flexibility in terms of the new texture content model:

- **Flexible visualization in a scenario specific way** - In the pedestrian navigation scenario it is important to accentuate landmarks. This can be done by decreasing detail of surrounding buildings and optionally highlighting the landmark itself. It is also beneficial to visualize landmark-based navigation instructions. Adapting detail and integrating additional information into textures (flexibility of content) is one of the research objectives and also a requirement in the case-study, which was achieved by the presented texturing approach.
- **Flexibility in terms of SDI integration** In the MoNa3D project (the case-study scenario) a 3D-SDI was defined in order to be the server side back-end of the envisioned pedestrian navigation system. The texture concept had to be integrated into this SDI. As described before, there were no smartphones on the market at the beginning of the project, which provided a programmable rendering pipeline. At that point the texture concept showed its flexibility as the texture information can be provided in different form (see chapter 4.8) and the reconstruction can happen on different stages of the client-server process. This shows the flexibility of the approach in terms of SDI environments and their processes under certain prerequisites, e.g. different client capabilities.
- **Flexibility in terms of data size** As texture tiles are significantly smaller than the complete façade texture they are easier to store and it is much more efficient to transmit them among systems. This was also a requirement of the MoNa3D project. Especially, when providing several representations, e.g. photo-realistic and sketch-rendered, only two tiles with different representation are needed, not two complete façade textures.

But also regarding texture content and data size, there is flexibility because of the layer-based content for textures. When an application does not need a specific



## **6. PEDESTRIAN NAVIGATION USING 3D CITY MODELS – A CASE-STUDY**

portion of the texture information, e.g. no shop signs or no additional thematic texture layers, only the relevant data (texture layers) need to be added to the service output. There is no need to load all available texture layers for all scenarios. Some might require only a subset of them and therefore data size can be kept to a minimum in terms of data transmission.

These three aspects are not only beneficial for the pedestrian navigation scenario and can certainly be helpful in other urban, task-driven scenarios, although one needs to look at the specific prerequisites in each particular case to make the best use of layer/zone-based textures.

# Chapter 7

## Conclusions & Future Work

This chapter concludes the thesis and sums up the results of the presented research. The chapter revisits the objectives that were formulated in chapter 1 and discusses outcomes in relation to these points. After looking at the single objectives a further section tries to draw conclusions for the wider research field and to answer the research question defined in chapter 1. In the second part of the chapter starting points for future work are suggested and open issues about the texture model as well as the implemented prototype are discussed. Future work on (3D-) SDI integration is also suggested as well as work on data acquisition and pedestrian navigation.

### 7.1 Conclusions

#### 7.1.1 Objective 1: Flexible Description

**Find a way to describe the content of (façade-) textures that allows the flexible representation and combination of different content as well as the integration of additional (non-photorealistic) information.**

This objective is met by using the zone/layer-based texture content approach presented in chapter 4. This concept is based on an empty texture, the null-texture (see chapter 4), that is already mapped onto geometry (by providing the suitable texture coordinates) and needs to be filled with the appropriate content. The appropriateness is defined by the scenario in which the 3D model is used.

## 7. CONCLUSIONS & FUTURE WORK

---

This work defines a model for texture content that is layer-based. Hence, different kinds of information can be managed and structured in separate layers. When filling the null-texture specific layers can be included or excluded and therefore different content for the null-texture can be produced. This concept provides a high flexibility and allows mixing different information inside the texture. Switching between different representations is also possible in this model.

The information inside a layer, however, needs to be defined as well. As a layer needs to hold separate façade entities (e.g. windows of a specific type), so they can be added to the null-texture independently of the other façade elements, the layer needs an internal concept to describe separate façade elements.

The independence among façade elements and the ability to put them into separate layers is achieved by the introduced zone approach. A zone is regarded as a box, in which a specific area of the façade is covered and arbitrary content for this box can be defined. Hence, boxes with the same type of information can be grouped into a layer. In order to describe these boxes this work adopts the pulse concept in Parish and Müller (2001). This pulse concept is extended (especially in the aspect of layers). In the work of Parish and Müller (2001) the pulse function approach is used to create valid, realistically looking façades for procedurally generated city models. In this thesis the reconstruction of realistic façades is not in focus, but aspects of information visualization, which made it necessary to extend the concept and add more semantics to layers (e.g. LoR).

Generally the zone/layer-based approach actually decouples façade elements inside textures and groups them according to their type. The decoupling allows rearranging content in new ways inside the texture. This is currently not possible or at least very hard to achieve with the standard texturing approach for 3D city models. The new texture content model tries to achieve a greater flexibility in term of information that can be transported or visualized in (façade-) textures. Especially for type-6 models, but for other types as well, which include various additional information about urban objects

## 7. CONCLUSIONS & FUTURE WORK

---

on city scale. This new form of visualization capability is expected to be beneficial in many scenarios.

### 7.1.2 Objective 2: Adjustability and additional Information

**Define concepts in order to adjust the ratio of realism as well as the ratio between realistic elements and the integrated additional information**

Using layers as the basis of the presented texture concept allows mixing and merging information in a flexible way when filling the null-texture. This also allows controlling the ratio of realism and to add additional information into the façade textures.

The user can certainly choose to add all the realistic content layers into the texture. In this way he would retrieve a result that is very close to photo-realistic texturing. Another representation that can be achieved by using the LoR concept (see section 2.3.4) is to omit certain realistic elements of the façade and adjust the realism of the building to the given scenario (e.g. Landmarks in pedestrian navigation in chapter 6). However, the layer/zone-based approach is not limited to modelling the realistic elements of a façade, also abstracted representation is possible. As zones, respective to their content, are not limited to photo-realistic tiles of the real-world equivalent, it is also possible to use standard textures from a texture database or a sketch like representation of elements. The latter would enable visualizations similar to the ones presented in (Döllner and Buchholz (2005), p.13).

Moving to an even more abstracted level, zones do not even have to represent façade elements, they can also model attribute information. When there are attributes available for an object, which refer to a specific part of the building a zone in the façade can be defined (e.g. to cover the first floor) and be colour-coded to represent the attribute value. This 'thematic layer' in which the attribute zone resides can also have a suitable z-level, hence windows can still be in front of the additional information, whereas the background material is behind it. This allows visualizing attributes by colour for a spe-

## 7. CONCLUSIONS & FUTURE WORK

---

cific part of the building, even though this part is not modelled as a separate geometry, because the information is included in the texture.

In terms of adjusting the ratio between realism and abstraction it is possible to reduce the number of realistic elements in the façade (e.g. no ledges, signs, etc.) when attribute representation is added. In this way it is possible for the system, into which the 3D model is integrated, to accentuate the relevant information and make it more prominent in contrast to the realistic façade elements. This capability supports ideas presented in Ferwerda (2003) on functional realism and aspects in the discussion provided by Schirra and Scholz (1998) on 'Abstraction vs. Realism' (chapter 2.3.2).

A further solution for integration of additional information that is presented in this work is the 'Real-Time Layer' (see section 4.6.3). This layer can integrate information into the façade content, which changes very frequently and independently from the rest of the content. An example for this kind of information is a video stream that is integrated into the façade. A more detailed discussion on the real-time layer is provided in the next section.

### 7.1.3 Objective 3: Real-time Content Changes

**Implement a suitable prototype in order to test the flexibility of (façade-) texture content and its appearance in regards to real-time contextual events.**

Changing the texture content in real-time during visualization is also a research objective in this work. This idea relates to the aspects of flexibility and adaptability of the model appearance, which this work wants to achieve. Therefore, the null-texture content does not only have to be filled at the time when the model is loaded (although this would be a valid scenario) and remains constant during the visualization. For specific scenarios like analysis, task-driven visualization, it is necessary that texture content can change during visualization. These changes can be triggered by users as well as the system itself.

## 7. CONCLUSIONS & FUTURE WORK

---

In the presented prototype in chapter 5 the texture reconstruction process is implemented directly by the programmable parts of the rendering pipeline on the graphics hardware. This allows changing the texture content very quickly and to react to scenario-specific events or user interaction. Therefore textures can be part of the 'user interface'. When the user requests certain information about buildings, it can be delivered through the façade texture. The implemented prototype shows that the texture can lose its relatively static character and, in a sense, become an interactive information display and part of 3D map-space.

The prototype in this work shows the capabilities of zone/layer-based textures only for one test building at the moment. This is due to open issues in data acquisition (see section 7.2.2). Nevertheless, the general approach to rebuild the façade textures during the rendering process directly on the graphics chip for entire city models is feasible. This is underpinned by work of Haegler et al. (2010) and Krecklau and Kobbelt (2011). Testing the zone/layer-based approach for larger parts of urban areas and for entire cities is part of future work as soon as data acquisition methods are available and connected with the presented ideas in order to model façades in the presented way for larger areas.

A further element that is implemented in the presented prototype, and refers to real-time content changes, is the Real-time Layer (see section 4.6.3). In this layer it is possible to integrate information that is frequently changing, whereas the rest of the façade content remains more or less unchanged. One example would be to integrate a video into the façade: here the video frames are changing very fast but at the same time other layers and content seldom change. Therefore the real-time layer decouples the real-time and the zone-based information, which makes a reconstruction of the façade for each video frame unnecessary. This aspect increases the efficiency of the presented approach and achieves better (user-) interaction. Triggering the null-texture reconstruction process for real-time information like a video, where content changes at about 30-60 fps, is not feasible at the moment.

## 7. CONCLUSIONS & FUTURE WORK

---

Real-time content changes are an important aspect to be considered for analysis and information visualization. As textures cover the whole object, especially important and prominent surfaces like façades, they are a very suitable channel for all kind of information. This is also presented by Lorenz and Döllner (2010) who investigated the capabilities of the programmable rendering pipeline in order to visualize 'feature surface properties' (see section 2.3.1 and 4.9.2). They also emphasize the possibility of real-time interaction in order to change the visualized information for analysis purposes. However, their prototype allows direct control of shader code to merge information and alter the visualization. This thesis argues that a layer-based structure helps to organize the content. Layers clarify possible combinations of content and aspects like the z-order become more understandable, especially for non-expert users. Another aspect is that Lorenz and Döllner (2010) do not describe how their analysis information can be merged with realistic elements, an aspect this work achieves through the layer concept. The relevance of window elements in combination with residential quality data in order to relate this information to certain areas of the building was discussed in section 4.9.2. Therefore an integration of their 'surface properties'-approach into the real-time layer would be interesting to investigate in future work.

### 7.1.4 Objective 4: 3D-SDI Integration

**Investigate how the new texture model can be integrated into 3D-SDIs and how services can address the absence of certain client capabilities.**

As 3D city models investigated in this work are mainly part of SDIs, where they can be linked with other geo-data and services, it is crucial for the new texture concept to be able to integrate into this environment (chapters 2.2.2 - 2.2.4).

The integration of the presented texture concept can be achieved at several stages of the client server process and therefore services should be able to provide different outputs for different scenarios. In chapter 4.8 several scenarios with different

## 7. CONCLUSIONS & FUTURE WORK

---

client and server capabilities are discussed and the chapter presents solutions how the zone/layer-based texture concept can be integrated into SDI environments.

In terms of client/server scenarios the presented approach appears to be sensible, because the reconstruction of the texture can be performed either on server or on client side, whichever component of the system has better capabilities to perform the task and where it fits best into the overall process.

For example, when the reconstruction can be performed on client side there is an additional benefit for the client/server system. In this case the texture content that needs to be transmitted from server to client only consists of tiles and the description. Hence, a smaller amount of data needs to be sent from server to client compared to a scenario where complete façade textures are exchanged. The smaller data size is especially beneficial for use-cases with mobile clients. Here normally bandwidth is limited, as well as available client memory. The tiles that need to be held in memory on the client need much less memory than complete textures. The number of reconstructed textures in graphics memory can be adapted to the capabilities of the graphics chip and other system resources.

Generally the concept can be beneficial in terms of SDIs as it is flexible both in regards to technical system capabilities as well as conceptual terms. Looking at the scenarios in a technical way the concept can deliver texture data in different forms with regards to client capabilities and resources. From a conceptual perspective 3D city models can be requested in different representations and according to scenario needs. As SDIs are designed to serve many different scenarios and to implement different workflows and processes, the ability to query for specific texture content in this environment is extremely beneficial. The zone/layer-based texture content appears to be a promising extension for 3D-SDIs and future test-bed scenarios would be useful to validate the performance of the new content model in SDI workflows.



## 7. CONCLUSIONS & FUTURE WORK

---

### 7.1.5 Objective 5: Evaluation of the presented Approach

**Evaluate the capabilities of the new texture content model in a specific use-case.**

The case-study on pedestrian navigation showed that 3D city models can be used as an integral part of the problem-solving process in terms of pedestrian navigation and that photo-realistic appearance is not the best type of visualization in this case.

The layer/zone-based texture allows adapting the degree of realism and in general the appearance of the model according to the pedestrian navigation scenario. Integration of additional information is easier, because elements of the façade can be controlled independently. Because façade elements are separate entities they can be put into specific z-levels and turned on/off independent from the other elements of the content. Controlling the z-level in the pedestrian navigation scenario helps to integrate additional information and put it into the foreground or background of the façade representation.

Façade elements can also be omitted completely if it is beneficial for the scenario, e.g. shop signs and logos that are not part of the navigation instruction and would distract the user from the relevant logo or aspect of a landmark. In general the possibility of grouping and packing elements into layers makes it easier to add additional information, like navigation hints, into the façade representation. This can be done by just adding further layers to the façade description. In this way the representation can be extended by an arbitrary number of additional layers, if necessary.

The layer/zone-based approach also provides more flexibility when the content of the texture needs to change. For example, when the route is recalculated the navigation hints in the textures might need to change for specific buildings. In this case only the layer with the additional hints needs to change and might require a reload of tiles. However, the rest of the texture content can remain unchanged and does not need to be reloaded. This capability provides more flexibility and efficiency in comparison to a normal photo-texture, where different representations would need to be pre-processed and the pre-processed textures would have to be loaded. In this case complete textures

## 7. CONCLUSIONS & FUTURE WORK

---

would have to be loaded and separate façade textures for each navigation hint of the particular façade might need to be stored.

The flexibility of the layered texture concept can also be observed in terms of the level of realism (chapter 2) with regards to the case-study. In the pedestrian navigation scenario the capability of adjusting the detail, also in terms of textures, is very important. This capability helps to accentuate the model, to highlight landmarks in their spatial context and to support the user in recognizing the landmark. This is important because pedestrians navigate a lot more with the help of landmarks than car drivers. Highlighting specific buildings also works by adjusting the LoR (besides other measures). By decreasing the detail for context buildings and increasing the detail for the landmark, it stands out of the surrounding even more and can be recognized more easily.

### 7.1.6 Summary

Summarizing this work one can say that a flexible way was found to represent adaptable texture content. The need for this new way of defining texture content is described in chapter 2 where the possible extension of map-space to building surfaces is discussed. As models can include a lot of additional information in form of attributes and semantics (type-6) it becomes necessary in certain use-cases to visualize this additional information. In '3D-maps' there needs to be some space where this information about buildings and/or building parts can be placed. Inserting 3D building objects into the map as icons can only help to a certain extent. Therefore, changing the idea of texture content in order to represent map-space instead of containing static photo-images can help to visualize city models in more ways than the photo-realistic one. This is why the new idea of texture content presented in this work refers to concepts of (digital) maps. Using layers as a concept to organize content and providing mechanisms to adjust realism/abstraction can be found in similar ways in the world of cartography and geovisualization. And as there is no single ultimate map illustration for all pur-

## **7. CONCLUSIONS & FUTURE WORK**

---

poses, there should also be different combinations of texture content in 3D city models in different scenarios.

In general this work widens the view on how textures can be used inside 3D city models. Textures should not only be regarded as static pixel-matrices, which can hold photo-image information to achieve a realistic digital representation of urban space. Textures can be more than that and recent developments in computer graphics hardware and algorithms make a different use possible. Information about urban objects should be visualized appropriately and lead to a thematic appearance of models. This requires learning from computer graphics and to use findings made in this field in combination with concepts from cartography and geovisualization to achieve suitable and scenario-dependent visualization of urban models. Textures in this regards can be used in a much more flexible way and can turn from static containers for image data to flexible information channels.

### **7.2 Future Work**

#### **7.2.1 Layer/Zone Model**

It would be interesting to investigate the presented concept further in terms of integrating additional forms of information as input for zones and extend the concept in this way. At the moment more or less texture input is generated for zones or single colour values are applied. It would be interesting to design a concept where arbitrary operations can be used to fill zones. Another interesting part regarding the concept is to define further operations how different layers can be merged inside the null texture. A simple case is currently implemented (Painter's Algorithm) but more complex operations can be envisioned.

The implemented prototype could be used in future work to investigate the rendering concept in shaders. The look-up of information for specific texels can be improved

## **7. CONCLUSIONS & FUTURE WORK**

---

and search strategies can be optimized. Special knowledge of the rendering pipeline concepts can lead to further performance gains and lead to a more satisfying user experience. These investigations need to be seen in the context of integrating large amounts of additional data and the aforementioned integration of new forms of zone input. This might lead to additional workload that needs to be handled by the rendering process to ensure the required performance for suitable visualization and interaction results.

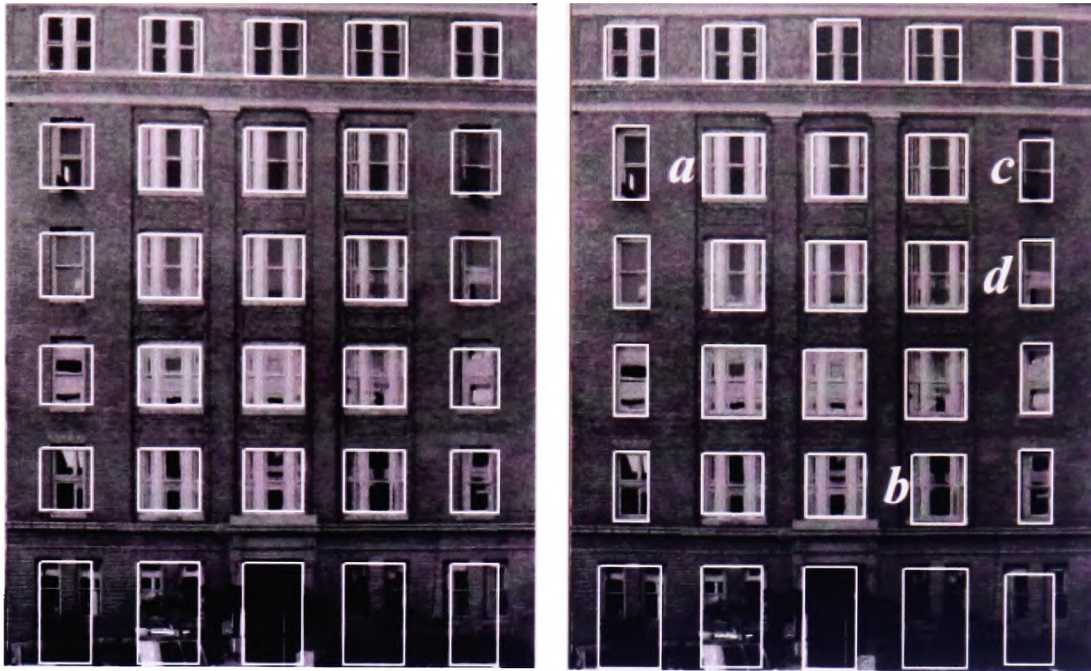
### **7.2.2 Data Acquisition**

At the moment data acquisition for tiles (tile textures) and the description (pulse positions) is done manually by digitizing the positions of the façade elements and cutting the relevant portions of the real-world image for the tiles' content. In the future an automated process for generating zone/layer based input data would be desirable.

Concepts that can already be found in literature and recent research appear to be promising candidates to be linked to the texture concept presented in this work. Algorithms and approaches for (semi-) automatic feature extraction from façade images can be found in Müller et al. (2007), Lee and Nevatia (2004) and others.

These concepts present ways to extract façade elements like windows, doors, etc. The provided information normally consists of the position of the element and the related portion of the façade image, which visually represents the particular element. Part of future work would be to transfer the information that is produced by the feature extraction process into the layer/tile representation. A piece of software, which can perform this transfer automatically, would be desirable in order to process input data for large city areas and high number of buildings.

## 7. CONCLUSIONS & FUTURE WORK



**Figure 7.1: Window Extraction Example** - Immediate results of extracting windows from Lee and Nevatia (2004)

### 7.2.3 Real-Time Layer

Future work in regards to real-time information integration would also be to investigate if there are alternatives to the real-time layer solution. The question would be if a separate layer for real-time content is necessary, which is restricted to one single input stream, or if it is possible to conceptually integrate real-time data as 'input' for specific tiles in the façade. If tiles' input/content would be allowed to change in real-time a solution needs to be found in order to change the portion of the overall texture that is represented by the tile in real-time. The benefit of integrating the real-time content as tile input would be that the content model as a whole would be more homogeneous. The real time data would be part of the zone/layer concept and would not be a separate information channel (real-time layer).

## **7. CONCLUSIONS & FUTURE WORK**

---

### **7.2.4 3D Spatial Data Infrastructures**

Several possible ways to integrate the new texture content model into client-server scenarios were discussed in this work (chapter 4.8). For future work in this regard it would be necessary to identify the extensions that need to be made to relevant data formats and interface standards in this field. Adapting these interfaces and data formats is necessary to expose new capabilities of the presented texture content model to other systems and client applications. Extensions to existing standards could be introduced into the standardization processes of relevant institutions, e.g. of the OGC.

### **7.2.5 Pedestrian Navigation**

In future work it would be interesting to see the presented approach for texture content to be implemented inside a pedestrian navigation prototype. This would allow conducting user tests in future work in order to investigate if pedestrians actually perform better in way-finding when the visualization options suggested in chapter 6 are used. These user tests would also need a cross-disciplinary approach with researchers from Human-Computer-Interaction, human perception/psychology, pedestrian navigation, etc. in order to find suitable ways of visualization using the presented texturing approach. As this cross-disciplinary investigation was out of scope of this work it would be interesting to see results for this specific case in the future.

# References

- Becker, S., Haala, N. and Fritsch, D. (2008), Combined knowledge propagation for facade reconstruction, in 'ISPRS Congress Beijing 2008, Proceedings of Commission V', p. 423 ff. 19
- Bildstein, F. (2005), 3d city models for simulation & training requirements on next generation 3d city models, in Gröger and Kolbe, eds, 'Proceedings of ISPRS WG III/4, Next Generation City Models, Bonn'. 15
- Blinn, J. F. and Newell, M. E. (1976), 'Texture and reflection in computer generated images', *Commun. ACM* **19**, 542–547.  
**URL:** <http://doi.acm.org/10.1145/360349.360353> 74, 76
- Bogdahn, J. (2006), *A Web3D Service for Public Participation in Urban Planning*, Faculty for Mathematics, Informatics and Surveying at Hochschule für Technik Stuttgart University of Applied Sciences, Thesis (Dipl.). 120, 125, 126
- Bogdahn, J. and Coors, V. (2009a), Procedural facade textures for 3d city models, in 'Proceedings of 27th Urban Data Management Symposium, Ljubljana, Slovenja', pp. 3–15. 120
- Bogdahn, J. and Coors, V. (2009b), Using 3d urban models for pedestrian navigation support, in 'Proceedings of the ISPRS working group III/4, IV/8, IV/5, 'GeoWeb 2009 Academic Track Cityscapes', Vancouver, BC, Canada, 27-31 July'. 158
- Bogdahn, J., Coors, V. and Sachdeva, V. (2007), A 3d tool for public participation in urban planning, in 'Urban and Regional Data Management UDMS Annual 2007', pp. 231–136. 120
- Brenner, C. and Haala, N. (1998), Fast production of virtual reality city models, in D. Fritsch, M. Sester and M. Englich, eds, 'Proceedings of the ISPRS Commission IV Symposium on GIS Between Visions and Applications', Vol. 32/4 of *International Archives of Photogrammetry and Remote Sensing*, ISPRS, pp. 77–84. 16
- Brenner, C., Haala, N. and D.Fritsch (2001), Towards fully automated 3d city model generation, in E. P. Baltsavias, A. Gruen and L. V. Gool, eds, 'Automatic Extraction of Man-Made Objects from Aerial and Satellite Images III', Proceedings of an International Workshop, Ascona, June 2001 v. 3, Centro Stefano Franscini, Monte Verita, Ascona, A. A. Balkema, Swets & Zeitlinger Publishers, p. 47 ff. 16
- Buchholz, H. (2006), REAL-TIME VISUALIZATION OF 3D CITY MODELS, PhD thesis, Mathematisch-Naturwissenschaftlichen Fakultät, Universität Potsdam. 31, 79
- Burger, W. and Burge, M. J. (2005), *Digitale Bildverarbeitung: Eine Einführung mit Java und ImageJ*, Berlin ; Heidelberg ; New York : Springer. 71

## REFERENCES

---

- Chazelle, B. (1982), A theorem on polygon cutting with applications, *in* 'Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on', pp. 339–349. 159
- Checkland, P. and Holwell, S. (2007), Action research, *in* N. Kock, ed., 'Information Systems Action Research', Vol. 13 of *Integrated Series in Information Systems*, Springer US, pp. 3–17. 54
- Coelho, A., Bessa, M., Sousa, A. A. and Ferreira, F. N. (2007), 'Expeditious modelling of virtual urban environments with geospatial I-systems', *Computer Graphics Forum* **26**(4), 769–782.  
**URL:** <http://dx.doi.org/10.1111/j.1467-8659.2007.01032.x> 17
- Coors, V. and Zipf, A. (2007), Mona 3d – mobile navigation using 3d city models, *in* 'Proceedings of the 4th International Symposium on LBS & Telecartography, Hongkong'. 12, 163, 164, 167
- Crnkovic, G. (2010), 'Constructive research and info-computational knowledge generation', *Model-Based Reasoning in Science and Technology* pp. 359–380.  
**URL:** <http://www.springerlink.com/index/4665535318660J02.pdf> 53
- Czerwinski, A., Kolbe, T. H., Plümer, L. and Stöcker-Meier, E. (2006), Spatial data infrastructure techniques for flexible noise mapping strategies, *in* K. Tochtermann and A. Scharl, eds, 'Proceedings of the 20th International Conference on Environmental Informatics Managing Environmental Knowledge. Graz 2006', pp. 99–106. 13
- Davis, A. (1992), 'Operational prototyping: a new development approach', *Software, IEEE* **9**(5), 70–78. 60, 61
- de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O. (2000), *Computational Geometry, 2nd Edition*, Springer. 92
- Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs', *Numerische Mathematik* **1**, 269–271. 10.1007/BF01386390.  
**URL:** <http://dx.doi.org/10.1007/BF01386390> 156
- Döllner, J. and Buchholz, H. (2005), Non-photorealism in 3d geovirtual environments., *in* 'Proceedings of AutoCarto. Las Vegas, NV, USA', ACSM, pp. 1–14. 173, 188
- Dykes, J., Andrienko, G., Andrienko, N., Paelke, V. and Schiewe, J. (2010), 'Editorial geovisualization and the digital city', *Computers, Environment and Urban Systems* **34**(6), 443 – 451. GeoVisualization and the Digital City - Special issue of the International Cartographic Association Commission on GeoVisualization.  
**URL:** <http://www.sciencedirect.com/science/article/pii/S0198971510000839> 39
- Dykes, J., MacEachren, A. and M.-J., K. (2005), *Exploring Geovisualization*, Elsevir, chapter 1, pp. 3–19. 26, 39
- Elias, B. and Paelke, V. (2008), *User-centered Design of Landmark Visualization*, Springer, chapter 3, pp. 33–56. 160, 161, 162, 179



## REFERENCES

---

- Fabritius, G., Krassnigg, J., Krecklau, L., Manthei, C., Hornung, A., Habbecke, M. and Kobbelt, L. (2008), City virtualization, *in* M. Schumann and T. Kuhlen, eds, 'Virtuelle und Erweiterte Realität: 5. Workshop der GI-Fachgruppe VR/AR'. 15, 16, 168
- Ferwerda, J. A. (2003), Three varieties of realism in computer graphics, *in* T. N. Rogers, Bernice E.; Pappas, ed., 'Human Vision and Electronic Imaging VIII. Proceedings of the SPIE', Vol. 5007, pp. 290–297. 37, 49, 189
- Finkenzeller, D. (2008), Modellierung komplexer Gebäudefassaden in der Computergraphik, PhD thesis, Fakultät für Informatik (Fak. f. Informatik), Institut für Betriebs- und Dialogsysteme (IBDS), Universitätsverlag Karlsruhe. 18
- Finkenzeller, D. and Schmitt, A. (2006), Rapid modeling of complex building facades, *in* D. Fellner and C. Hansen, eds, 'Short paper proceedings of Eurographics 2006', pp. 95–98. 18
- Frueh, C., Sammon, R. and Zakhor, A. (2004), 'Automated texture mapping of 3d city models with oblique aerial imagery', *3D Data Processing Visualization and Transmission, International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'04)* **0**, 396–403.  
**URL:** <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1335266> 17, 77
- Gaisbauer, C. and Frank, A. U. (2008), Wayfinding model for pedestrian navigation, *in* 'Proceedings of the 11th AGILE International Conference on Geographic Information Science', University of Girona, Spain. 158
- Gartner, G. (2004), Location-based mobile pedestrian navigation services the role of multimedia cartography, *in* 'International Joint Workshop on Ubiquitous, Pervasive and Internet Mapping, Tokyo, Japan', ICA. 180
- Gillham, B. (2000), *Case Study Research Methods*, London: Continuum. 63
- Haegler, S., Wonka, P., Arisona, S. M., Van Gool, L. and Müller, P. (2010), 'Grammar-based encoding of facades', *Computer Graphics Forum* **29**(4), 1479–1487.  
**URL:** <http://dx.doi.org/10.1111/j.1467-8659.2010.01745.x> 168, 190
- Hamruni, A. M. (2010), The use of oblique and vertical images for 3D urban modelling, PhD thesis, Faculty of Engineering, Departement of Civil Engineering, The University of Nottingham. 16
- Hart, P., Nilsson, N. and Raphael, B. (1968), 'A formal basis for the heuristic determination of minimum cost paths', *Systems Science and Cybernetics, IEEE Transactions on* **4**(2), 100–107. 156
- Heckbert, P. (1986), 'Survey of texture mapping', *IEEE Computer Graphics and Applications* **6**, 56–67. x, 74, 75
- Hu, J., You, S. and Neumann, U. (2003), 'Approaches to large-scale urban modeling', *Computer Graphics and Applications, IEEE* **23**(6), 62–69. 16

## REFERENCES

---

- Igarashi, T. and Cosgrove, D. (2001), Adaptive unwrapping for interactive texture painting, in 'Proceedings of the 2001 symposium on Interactive 3D graphics', I3D '01, ACM, New York, NY, USA, pp. 209–216.  
**URL:** <http://doi.acm.org/10.1145/364338.364404> 79
- J. Haist, V. C. (2005), The w3ds-interfaces of cityserver3d, in 'Proceedings of the 1st International Workshop on Next Generation 3D City Models'. 24, 26
- Kada, M. (2007), Zur maßstabsabhängigen Erzeugung von 3D-Stadtmodellen, PhD thesis, Institut für Photogrammetrie der Universität Stuttgart. 47
- Kada, M., Klinec, D. and Haala, N. (2005), Facade texturing for rendering 3d city models, in 'ASPRS Conference 2005', pp. 78–85. 17, 77
- Kallmann, M. (2005), Path planning in triangulations, in 'In Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, Scotland', pp. 49–54. 159
- Kasanen, E., Lukka, K. and Siitonen, A. (1993), 'The constructive approach in management accounting research', *Journal of Management Accounting Research* 5(June 1991), 243–264.  
**URL:** <http://www.mendeley.com/research/constructive-approach-management-accounting-research/> 53, 55
- Kersting, O. (2002), Interaktive, dynamische 3D Karten zur Kommunikation raumbezogener Informationen, PhD thesis, Mathematische-Naturwissenschaftliche Fakultät der Universität Potsdam. 139
- Knapp, S., Bogdahn, J. and Coors, V. (2007), Improve public participation in planning processes by using web-based 3d-models for communication platforms, in 'Proceedings of REAL CORP 2007'. 14, 22
- Kokas, N. (2008), AN INVESTIGATION INTO SEMI-AUTOMATED 3D CITY MODELLING, PhD thesis, University of Nottingham. 13, 14, 16
- Kolbe, T. H. (2008), *Geospatial Information Technology for Emergency Response (ISPRS Book Series)*, Taylor & Francis Group London, London, chapter CityGML 3D city models and their potential for emergency response, pp. 257–274. 15
- Kray, C., Elting, C., Laakso, K. and Coors, V. (2003), Presenting route instructions on mobile devices, in 'Proceedings of the 8th international conference on Intelligent user interfaces', IUI '03, ACM, New York, NY, USA, pp. 117–124.  
**URL:** <http://doi.acm.org/10.1145/604045.604066> 171, 172, 174, 175
- Krecklauer, L. and Kobbelt, L. (2011), Realtime compositing of procedural facade textures on the gpu, in 'Proceedings of the 4th ISPRS International Workshop 3D-ARCH 2011: "3D Virtual Reconstruction and Visualization of Complex Architectures"'. 190
- Lafarge, F., Descombes, X., Zerubia, J. and Pierrot-Deseilligny, M. (2007), 3d city modeling based on hidden markov model, in 'Image Processing, 2007. ICIP 2007. IEEE International Conference on', Vol. 2, pp. II–521–II–524. 16

## REFERENCES

---

- Laycock, R. and Day, A. (2006), 'Image registration in a coarse three-dimensional virtual environment', *Computer Graphics Forum* **Volume 25**(1), pp. 69–82. 16
- Lee, S. C. and Nevatia, R. (2004), Extraction and integration of window in a 3d building model from ground view images, in 'Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on', Vol. 2, pp. II–113 – II–120 Vol.2. 19, 196, 197
- Legakis, J., Dorsey, J. and Gortler, S. (2001), Feature-based cellular texturing for architectural models, in 'Proceedings of the 28th annual conference on Computer graphics and interactive techniques', SIGGRAPH '01, ACM, New York, NY, USA, pp. 309–316.  
**URL:** <http://doi.acm.org/10.1145/383259.383293> 18
- Lipp, M., Wonka, P. and Wimmer, M. (2008), Interactive visual editing of grammars for procedural architecture, in 'ACM SIGGRAPH 2008 papers', SIGGRAPH '08, ACM, New York, NY, USA, pp. 102:1–102:10.  
**URL:** <http://doi.acm.org/10.1145/1399504.1360701> 18
- Lorenz, H. and Döllner, J. (2006), Towards automating the generation of facade textures of virtual city models. ISPRS Commission II, WG II/5 Workshop, Vienna. 16, 17
- Lorenz, H. and Döllner, J. (2010), '3d feature surface properties and their application in geovisualization', *Computers, Environment and Urban Systems* **34**(6), 476 – 483. GeoVisualization and the Digital City - Special issue of the International Cartographic Association Commission on GeoVisualization.  
**URL:** <http://www.sciencedirect.com/science/article/pii/S0198971510000360> 32, 33, 113, 114, 191
- Loya, A., Adabala, N., Das, A. and Mishra, P. (2008), A practical approach to image-guided building façade abstraction, in 'Computer Graphics International Conference (CGI) 2008'. 86
- Lynch, K. (1960), *The Image of the City*, The MIT Press. 161
- Maaß, S. (2009), Techniken zur automatisierten Annotation interaktiver geovirtueller 3D-Umgebungen, PhD thesis, Mathematisch-Naturwissenschaftlichen Fakultät, Universität Potsdam. 31
- MacEachren, A. M. and Kraak, M.-J. (1997), 'Exploratory cartographic visualization: advancing the agenda', *Computers & Geosciences Special issue on exploratory cartographic visualization* **23**, 335–343.  
**URL:** <http://dl.acm.org/citation.cfm?id=260786.260788> 39, 114
- MacMillan, R., Jones, R. and McNabb, D. H. (2004), 'Defining a hierarchy of spatial entities for environmental analysis and modeling using digital elevation models (dems)', *Computers, Environment and Urban Systems* **28**(3), 175 – 200.  
**URL:** <http://www.sciencedirect.com/science/article/pii/S019897150300019X> 41

## REFERENCES

---

- Müller, P., Wonka, P., Haegler, S., Ulmer, A. and Van Gool, L. (2006), 'Procedural modeling of buildings', *ACM Trans. Graph.* **25**, 614–623.  
**URL:** <http://doi.acm.org/10.1145/1141911.1141931> 18
- Müller, P., Zeng, G., Wonka, P. and Van Gool, L. (2007), Image-based procedural modeling of facades, in 'ACM SIGGRAPH 2007 papers', SIGGRAPH '07, ACM, New York, NY, USA.  
**URL:** <http://doi.acm.org/10.1145/1275808.1276484> 18, 19, 34, 196
- Murray, N. and Hughes, G. (2008), *Writing up your university assignments and research projects*, Open University Press. 58
- OGC (2008a), 'City geography markup language (citygml) encoding standard'. 20, 45, 46, 84
- OGC (2008b), 'Kml standard'. 43
- OGC (2010a), 'Draft for candidate opengis web 3d service interface standard'. 104
- OGC (2010b), 'Opengis web feature service 2.0 interface standard'. 104
- Ortin, D. and Remondino, F. (2005), Occlusion-free image generation for realistic texture mapping, in 'Proceedings of the ISPRS WG V/4 "Virtual Reconstruction and Visualization of Complex Architectures"', Venice, Italy'. 16
- Parish, Y. I. H. and Müller, P. (2001), Procedural modeling of cities, in 'Proceedings of the 28th annual conference on Computer graphics and interactive techniques', SIGGRAPH '01, ACM, New York, NY, USA, pp. 301–308.  
**URL:** <http://doi.acm.org/10.1145/383259.383292> 18, 89, 90, 112, 113, 187
- Poynton, C. (2003), *Digital video and HDTV: algorithms and interfaces*, Morgan Kaufmann. 70, 72, 73
- Qiu, F., Zhao, Y., Fan, Z., Wei, X., Lorenz, H., Wang, J., Yoakum-Stover, S., Kaufman, A. and Mueller, K. (2004), Dispersion simulation and visualization for urban security, in 'Proceedings of the conference on Visualization '04', VIS '04, IEEE Computer Society, Washington, DC, USA, pp. 553–560.  
**URL:** <http://dx.doi.org/10.1109/VISUAL.2004.24> 15
- Rehrl, K., Göll, N., Leitinger, S. and Bruntsch, S. (2005), Combined indoor/outdoor smartphone navigation for public transport travellers, in G. Gartner, ed., 'Proceedings of the 3rd Symposium on LBS & TeleCartography 2005', number 74 in 'Schriftenreihe der Studienrichtung Vermessungswesen und Geoinformation an der TU Wien', pp. 235–239. 180
- Ricard, J., Royan, J. and Aubault, O. (2008), Visualization of real cities based on procedural modeling, in 'Proceedings of Workshop on Virtual Cityscapes: Key Research Issues in Modeling Large-Scale Immersive Urban Environments, 2008 IEEE Virtual Reality Conference'. 19, 110

## REFERENCES

---

- Ripperda, N. and Brenner, C. (2007), Data driven rule proposal for grammar based facade reconstruction, in U. Stilla, H. Mayer, F. Rottensteiner, C. Heipke and S. Hinz, eds, 'Proceedings of PIA07 - Photogrammetric Image Analysis', Vol. Volume 36, Part 3 / W49A of *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*, pp. 1–6. 19
- Rost, R. J., Kessenich, J. M., Lichtenbelt, B., Malan, H. and Weiblen, M. (2006), *OpenGL Shading Language, Second Edition*, Addison Wesley. 118, 119, 120, 121, 122, 123
- Royan, J., Gioia, P., Cavagna, R. and Bouville, C. (2007), 'Network-based visualization of 3d landscapes and city models', *Computer Graphics and Applications, IEEE* **27**(6), 70 –79. 87
- Schirra, J. and Scholz, M. (1998), *Computational Visualization Graphics, Abstraction and Interactivity*, Springer, chapter Abstraction versus Realism: Not the Real Question, pp. 379–402. 35, 36, 48, 49, 189
- Schmidt, U. (2009), *Professionelle Videotechnik: Grundlagen, Filmtechnik, Fernsehtechnik, Geräte- und Studioteknik in SD, HD, DI, 3D*, Springer-Verlag Berlin Heidelberg.  
**URL:** <http://dx.doi.org/10.1007/978-3-642-02507-5> 73
- Schmitz, A. and Kobbelt, L. (2006), Real-time visualization of wave propagation, in O. Spaniol, ed., 'Proceedings of the First International Workshop on Mobile Services and Personalized Environments', Vol. P-102 of *Lecture Notes in Informatics (LNI) Proceedings Series of the Gesellschaft für Informatik (GI)*, pp. 71–80. 15
- Schroder, C. J., Mackaness, W. A. and Gittings, B. M. (2011), 'Giving the 'right' route directions: The requirements for pedestrian navigation systems', *Transactions in GIS* **15**(3), 419–438.  
**URL:** <http://dx.doi.org/10.1111/j.1467-9671.2011.01266.x> 159, 174
- Schulte, C. and Coors, V. (2008), Development of a citygml ade for dynamic 3d ood information, in 'In Proceedings Joint ISCRAM-CHINA and GI4DM Conference on Information Systems for Crisis Management, Harbin, China, 2008' 14
- Schwalbe, E., Maas, H.-G. and Seidel, F. (2005), 3d building model generation from airborne laser scanner data using 2d gis data and orthogonal point cloud projections, in 'Proceedings of ISPRS WG III/3, III/4, V/3 Workshop "Laser scanning 2005"', ISPRS. 16
- Shiode, N. (2001), '3d urban models: Recent developments in the digital modelling of urban environments in three-dimensions', *GeoJournal* **52**, 263–269. 10.1023/A:1014276309416.  
**URL:** <http://dx.doi.org/10.1023/A:1014276309416> 11, 12
- Stadler, A. and Kolbe, T. H. (2007), Spatio-semantic coherence in the integration of 3d city models, in 'Proceedings of the 5th International ISPRS Symposium on Spatial Data Quality ISSDQ, Enschede, The Netherlands'. 7, 19, 20, 21, 22, 23, 50

## REFERENCES

---

- Stappers, P. J., Gaver, W. and Overbeeke, K. (2003), *Psychological Issues in the Design and Use of Virtual and Adaptive Environments*, Lawrence Erlbaum Associates, Inc., chapter Beyond The Limits Of Real-Time Realism: Moving From Stimulation Correspondence To Information Correspondence, pp. 91–111. 35
- Strzalka, A., Bogdahn, J. and Eicker, U. (2010), 3d city modeling for urban scale heating energy demand forecasting, in 'IAQVEC 2010, 7th International Conference on Indoor Air Quality, Ventilation and Energy Conservation in Buildings, Syracuse, New York, USA'. 13
- Wang, H. (2007), DATA SERVICE FRAMEWORK FOR URBAN INFORMATION INTEGRATION, PhD thesis, School of the Built Environment, University of Salford, Salford, UK. 56
- Wang, X., Totaro, S., Taillandier, F., Hanson, A. R. and Teller, S. (2002), Recovering facade texture and microstructure from real-world images, in 'Proc. 2nd International Workshop on Texture Analysis and Synthesis, pp. 145-149, Copenhagen, Denmark, June 2002'. 19
- Wang, Y., Schultz, S. and Giuffrida, F. (2008), Pictometry's proprietary airborne digital imaging system and its application in 3d city modelling, in 'ISPRS Congress Beijing 2008, Proceedings of Commission I', Vol. 37 Part B1 of *International Archives of Photogrammetry and Remote Sensing*, ISPRS, pp. 065–1069. 16
- Web3D Consortium (2003), 'Iso/iec 14772-1:1997 and iso/iec 14772-2:2004 virtual reality modeling language (vrml)', Website. 21, 26
- Weber, B., Müller, P., Wonka, P. and Gross, M. (2009), 'Interactive geometric simulation of 4d cities', *Computer Graphics Forum* **28**(2), 481–492.  
**URL:** <http://dx.doi.org/10.1111/j.1467-8659.2009.01387.x> 15
- Wloka, M. (2005), *ShaderX3*, Charles River Media, chapter Improved Batching via Texture Atlases, pp. pp. 155–167. 79, 80
- Wonka, P., Wimmer, M., Sillion, F. and Ribarsky, W. (2003), Instant architecture, in 'ACM SIGGRAPH 2003 Papers', SIGGRAPH '03, ACM, New York, NY, USA, pp. 669–677.  
**URL:** <http://doi.acm.org/10.1145/1201775.882324> 18
- www.londonair.org.uk (2011), '3-d map of air pollution in london', website.  
**URL:** <http://www.londonair.org.uk/london/asp/virtualmaps.asp> 14
- Wytzisk, A. and Sliwinski, A. (2004), Quo vadis sdi?, in 'Proceedings of the 7th AGILE Conference on GI Science. Heraklion', pp. 43–49. 29