THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par: *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le 14 Juin 2018 **par:**

Noura EL HAJE

A Heterogeneous Data-Based Proposal for Procedural 3D Cities Visualization and Generalization

---

JURY

Gilles GESQUIERE	Rapporteur
Stéphane MERILLOU	Rapporteur
Nancy RODRIGUEZ	Examinateur
Cédric SANZA	Examinateur
Véronique GAILDRAT	Co-Directeur
Jean-Pierre JESSEL	Directeur

École doctorale et spécialité:

EDMITT: Domaine: Mathématiques et Informatique

Unité de Recherche:

Institut de Recherche en Informatique de Toulouse

Directeur de Thèse:

Jean-Pierre JESSEL

Rapporteurs:

Gilles GESQUIERE et Stéphane MERILLOU

To the soul of my father
The strength of my mother
The ambition of my brother
and the sensitivity of my sister

Acknowledgment

First of all, I would like to thank my thesis supervisor, Prof. Jean-Pierre Jessel, for both his technical and moral support and encouragement throughout the research and thesis writing. I also express my gratitude to Prof. Veronique Gaildrat at IRIT and Dr. Cedric Sanza in Vortex team, for their useful comments which put me in the right track when it was needed.
Many thanks to my colleagues and friends at IRIT whose positive and motivating talks helped me go to the end of my thesis: Chantal, Charlotte, Clémentine, Lenka, Francois and Martin. My friends from Lebanon: Mouin, Azza, Hoda, Amani, and Ranine thank you for the visits, the dinner evenings, and the phone calls to check on me during my most stressful times.

Last but not least, I owe this achievement to my beloved family: my mother Souha, my brother Murad, my sister Maya and my aunt Rouba who mean the world to me. They were and will always be the source of strength in my life. My belated aunt Maha and my second mother, thank you for the care, the delicious meals and the listening you've always given to me and my siblings. May your soul rest in peace. Dad, if it wasn't for you I wouldn't be writing this thesis. Thank you for everything you have done for me, You were and always be my idol and the best man in my life. Let me seize this opportunity to dedicate this work for you and tell you how much I love you and miss you.

Concerning the development part, I am grateful to the Cesium google group community who were willing to answer my questions about the platform and guide me through the development process. Patrick Cozzi: the main developer of Cesium and an active member in graphics and open source communities[1]. Sean Lilly: a 3D software developer who worked on improving Cesium's Graphics Engine and creating the 3D Tiles (a concept we are going to explain further in this thesis). Thank you for Virtual-CitySystems team, who gave me the permission to download a sample dataset of Berlin

[1] https://cesiumjs.org/team/PatrickCozzi/

from their portal[2] in order to run some comparison tests explained in chapter6. The data is updated and maintained by the support team of the business location center.

[2]http://www.businesslocationcenter.de/berlin3d-downloadportal/index.en.html

Abstract

Procedural modeling covers a number of techniques in computer graphics allowing to automatically create 3D[3] models and textures from a set of rules. This automatic generation has been largely adopted by researchers in order to solve the manual generation burdens such as time and cost charges. Procedural modeling has been an active research topic for at least thirty years and therefore many works have been established on tools, concepts and techniques that allow the generation of 3D virtual worlds in general and urban contents in particular.

Treating and managing real urban data have been a challenge, the main reason being the heaviness of the 3D files that the developer is usually working on and his obligation to go through a preprocessing step for files preparation and conversion before usage in an application for city construction or simulation. Many software have been developed and adopted for constructing, editing and visualizing 3D geographic data-based cities. They are destined for visualization, simulation or games contents creation and sometimes they have the capability to deal not only with geometric, but semantics for performance and consistency improvement. However, the visualization and optimization of this kind of data on the web is still restrictive. In this thesis, we propose a solution for real cities visualization and optimization based on standards, tools and APIs[4]. The procedural work in the 3D city modeled is highlighted by the procedural texturing and an optimized multi-resolution rendering.
We will also investigate multiple representation methods for the distinguished buildings throughout the animation rendering. In this context, a saliency method is applied to compute the important points in the scene. The overall contribution of this thesis is to highlight the necessity of optimizing the visualization of large urban data, and knowing how to keep the focus on the objects of high interest by means of saliency and texturing during the visualization process.

This is also to show how some methods and techniques could be unified in one platform to solve the challenging problems of visualizing large 3D urban data.

[3]Three-Dimensional
[4]Application Programming Interface

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction and Related Work

Chapter 1

Introduction

Contents

This thesis project is born from a collaboration project between the research team VORTEX[1] at IRIT[2] from one hand and education professionals, companies and public entities in the other hand.
The collaboration project SCOLA is basically an e-learning platform based on serious games usage at schools. It helps users acquire and spot predefined skills. This platform offers to the teachers a new flexible tool that creates pedagogy-related scenarios and customized students' records.
Several contributions were assigned to IRIT. One of which is to suggest a solution for automatic creation of 3D environments, to be integrated in the game scenario. This solution is meant to prevent 3D computer graphic artists from manually modeling large detailed 3D environments, which can be very costly and time consuming.

1.1 Background and Motivation

Various applications and prototypes have been developed for enabling the user to generalize and visualize his own virtual world mostly from a set of rules. Therefore, there

[1]Visual Objects: from Reality to EXpression
[2]Institut de Recherche en Informatique de Toulouse

is no unique representation schema to the virtual world due to the heterogeneity and diversity of 3D contents conception, especially city models. This constraint has led us to largely rely in our project on real 3D urban data instead of predefined custom data made by the game designer.

The advancement in Computer Graphics, high computational capacities and web technologies have largely revolutionized the data reconstruction and visualization techniques. These techniques are applied in various domains, starting with video games, simulations, and ending with films that use procedurally generated spaces and characters' animations. Though modern computer games do not have the same memory and hardware restrictions that earlier games had, the use of procedural generation is frequently employed to create randomized games, maps, levels, characters, or other facets that are unique on each game play. The tendency nowadays is shifted to the use of GIS to create urban worlds, especially after their success in being implemented around the world to help support many domain applications.

GIS is more particularly dedicated for applications such as simulation, disaster management and urban planning, without much usage in games, except for the game "Minecraft" which latest version offers mapping using real world cities[3]. Existig urban data usage is becoming more and more tempting for use in mapping applications for two main reasons: first it allows to understand the spatial contents of urban objects in a more logical way and second it provides a common platform to integrate city level information from different resources and make them accessible to users.

A virtual 3D city model is a digital representation of urban space that describes geometrical, topological, semantical, and appearance properties of its components. In general, a 3DCM serves as an integration platform for multiple facets of an urban information space, as pointed out by Batty [Bat07]: *In short, the new models are not simply the digital geometry of traditional models, but large scale data bases which can be viewed as 3D. As such, they already represent a way of merging more abstract symbolic or thematic data, even symbolic models, into this mode of representation.* The importance of 3DCMs are demonstrated by application examples from various fields. The most related application field in this work is the *Visualization*, which uses the 3DCM to associate it with thematic data to give spatial context, such as in [WA09]. Furthermore, components of the 3DCM can be used as visual variables, for example, by adapting color or height of buildings according to the data to be mapped. Visualizing

[3]Geodata in Minecraft: http://www.geoboxers.com/inspiration-geodata-in-minecraft/

in the domain of GIS is called GeoVisualization, with the aim to communicate information, explore data, buid and test hypotheses [MK01]. It draws from different fields, such as cartography, computer graphics, and geographic information science, to effectively present geospatial data. In addition to technical issues of rendering, aesthetic and design aspects have to be taken into consideration.

The innovative, yet complicated part of this thesis is to combine the concepts of procedural modeling in one hand and GIS data in the other hand in one 3D map. The map is inspired of real city districts which will be faithfully reproduced. The procedural part is mainly focused on some distinguished buildings texturing mechanism, that respects the optimization idea throughout the scene. The efficiency of the texturing method used is shown by a case study established in the last chapter. A discussion about the generalization necessity for displaying large data in the web is established, and a generalization method adapted to the project context is applied, and is highlighted by a quantitative evaluation, with a possibility of improvement.

1.1.1 Virtual 3D City Models

Virtual 3D city models have been mainly used in the past for the visualization or graphical exploration of cityscapes. Nowadays, the increasing number of applications like urban planning, facility management, and personal navigation require additional information about the city objects given in a standardized representation. The term *C*ity Model is frequently used to refer to generic 3D models for computer graphics applications. Such models allow, for instance, for flying or walking virtually through a city [DB98]. In general, however, a generic 3D model is frequently not sufficient to provide all relevant information of a city. Therefore, 3D city models comprise nowadays not only spatial and graphical aspects, but also ontological structures including thematic classes, attributes, and their interrelationships.

Objects are decomposed into parts to a logical criteria instead of graphical considerations. For example, a building will be decomposed into different building parts, if they have variant roof types and their own entrance, like a house and its garage. Furthermore, virtual city models provide a basis for spatial querying of thematic data [DDA04], and this data is changeable depending on the application domain in which the model is used, and the preferable requirements on the information provided by this city model.

Throughout this thesis, we will be mentioning LOD1, LOD2, LOD3, and LOD4 as defined by the CityGML standard, represented as follows:

- LOD1: Buildings are represented as block buildings, i.e., simple 3D shapes that can be obtained by extruding horizontal footprint polygons along the vertical

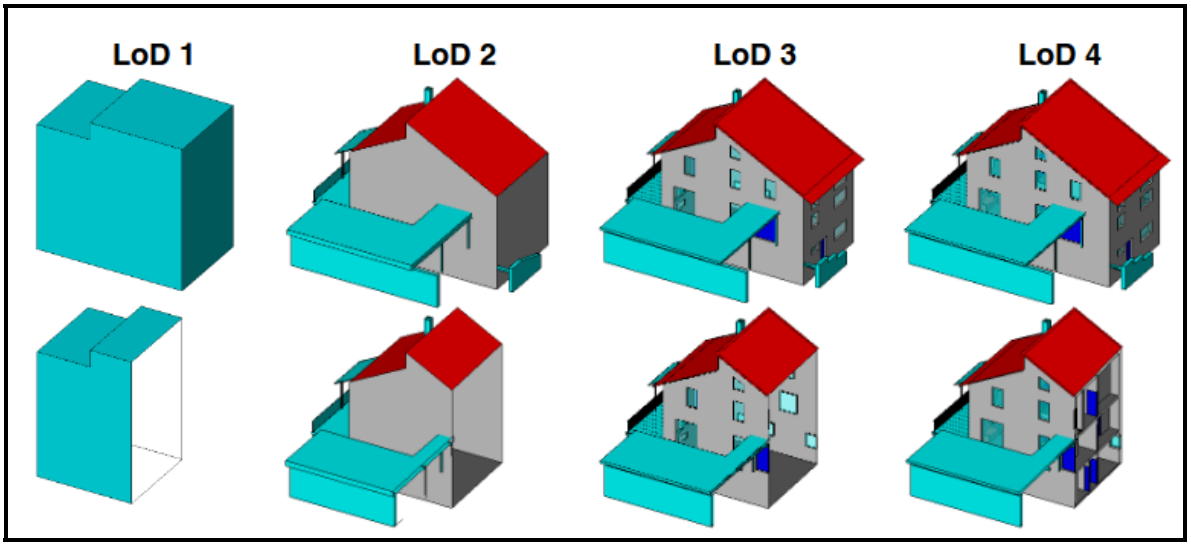


Figure 1.1: The Five Levels of Details in CityGML: [Hae14]

axis. Facades and roofs may be textured in all LODs.

- LOD2: Buildings may have an explicitly specified roof geometry, described by a set of polygons in 3D. In addition, the outer walls and the roof of a building are represented as separate thematic objects.
- LOD3: Buildings are represented as detailed architectural models. Openings such as windows and doors are accessible as separate thematic objects. All parts of a building may be textured.
- LOD4: In addition to LOD3, interior building structures such as interior walls, rooms, and furniture are provided as separate thematic objects.

A single building may contain representations for multiple LODs simultaneously. The geometry of buildings and sub-objects of buildings is specified using a general geometric model in CityGML, which is also used for other city model components. In the geometric model, a 3D object is described by a hierarchical aggregations of polygons that represent the object's boundary surfaces. The model supports the specification of texture images and texture coordinates for all surfaces.

City Databases

In the specific area of Geographic Information Systems, various databases tools have been developed to store spatial information [RSV01]. Oracle spatial provides spatial indexing and search for geographic information[4]. However this tool and similar tools

[4]Oracle Corporation 2012:https://www.oracle.com/database/spatial/index.html

are tend to focus on large geographic 2D data rather than the processing of individual polygon model edits. In addition, open standards have emerged that aim at representing geographic data such as OpenGIS[5], and these have a strong relationship to web 3D tools.

When dealing with large cities, appropriate qualification of 3D data is necessary. The qualification can be done by an automated process or by manual interpretation. According to Kolbe [Kol09], the semantic modeling only makes sense if the semantic information can be used by different customers with multiple applications. This would require to find a common information model over the different users and applications. One of these models is CityGML, which has an aim to reach a common definition and understanding of the basic entities, attributes, and relations within a 3D city model. With the core model provided by CityGML, information exchange between different disciplines can be aligned with the objects of the city model. Similar ideas are discussed in the Towntology Project [TKRL07] and the nD Modeling Project [HWT+05], which discussed the issue of urban data sets integration necessity at different scales by developing an urban scale city database. The integration and interoperability strategies should also be addressed in order to enhance the city proper management.

1.2 Research Objectives

We derive in this section research questions that address the need for optimization techniques for large 3D city models. This need represents a key challenge for an optimized visualization. An important question of this thesis is: Given a converted 3D city model ready to be streamed on the web, what aspects should the optimization take and at which levels of details of the scene?. This question involves some requirements:

Consistent Reduced Information: Optimized representation of the city needs to have reduced information, while maintaining sufficient both the general shapes of the buildings, and a focus on the important buildings throughout the animation.

Semi-Automatic Reduction: The optimization technique, especially the generalization should have some degree of automation, especially regarding the buildings which are more or less similar.

Saliency Control: Saliency metrics should be used to keep the viewpoint with respect to the distinguished buildings. That is why a method for distinguishing these buildings should be applied.

The second research question occurs in the context of texturing rendering: How can

[5]GeoTools:http://docs.geotools.org/stable/userguide/library/opengis/

we render the different *Texture Atlases* (to be explained) with the different resolutions? This question involves the following requirement:

Multi-Resolution Texturing: It is crucial to make the textures adapt the camera viewpoint. The textures display should take into consideration the distance of the buildings from the camera, where the textures of the far buildings should have a lower resolution than the ones of the close buildings. **The necessity of Triangulation:** To optimize the texturing process, the textures should be applied on simplified models of the buildings. It is much easier to treat the meshes as groups of triangles instead if polygons.

1.3 Contributions and Thesis Outline

As we explained before, the research was done along two main lines: Generalization and texturing. The contributions of this thesis are highlighted below:

In Buildings Generalization To improve visualizations of 3D city models and overcome the visual complexity in terms of number of objects and geometric complexity, with a lack of focus on the important landmarks. We present a generalization solution applied on the 3D buildings in order to reduce the heaviness of the polygons in the scene. Properties of the method and resulting representations are as follows:

- The derived representations yield a high degree of generalized objects. Currently, approaches focus on individual buildings simplification, aiming at the reduction of computational complexity and memory consumption, while maintaining visual appearance.
- The technique could be applied to several buildings together which have common base or footprints, or similar outside structure.
- Distinguished objects are not very much refined, to preserve their importance and facilitate orientation. Saliency metrics are applied to detect these objects and keep tracks of them throughout the animation rendering.

In Buildings Texturing Textures are important elements to sustain the characteristics of the city. A normal 3D texturing for all the objects with a very large amount of textures is not an options. Therefore, we present a method to texture the building using texture atlas. It consists of packing a certain amount of textures in one large texture. The textures resolution are adapted to the camera viewpoint. Properties of the technique are as follows:

- The concept of using multi-resolutions textures enhances the focus on the distinguished buildings, even beyond their original visibility.
- The packing algorithm is chosen among several frequently used ones, for a more optimized outcome. It is clearly shown in the distinguished buildings.
- The importance of rendering the scene with different resolutions of textures is also validated.

1.3.1 Thesis Outline

The document is organized as follows:
1: Introduction, this section presents an overview about the research work. It describes the basic ideas of the topic, the problem statement, and the research objectives. It details the facts that led to this research and the possible problems that could occur in the implementation, and an attempt to solve these problems.

2: Literature Review, it deals with the main focus topics of the research: 3D virtual cities visualization, generalization, and focus points in rendering. The study of existing methods and techniques will help us compare the different solution and focus on the most relevant ones to our work.

3: This chapter focuses on the raw data control, what are the different types of files used, and how they are converted and visualized. It also deals with our proposal in simplifying the buildings with a medium complexity. The distinguished buildings are only optimized with one iteration in order to keep the necessary details.

4: This chapter is divided into two parts: the first part deal with the texturing technique used: the texturing the environment using texture atlas and multi-resolution texture adaptation. The implementation of the multi-resolution solution for rendering is also discussed.

5: This chapter is a discussion and analysis of the works in the previous chapters. It allows to evaluate the different results quantitatively and application on use cases. A test of texturing in the animation context is established, the efficiency of the texturing and LOD management is evaluated, along with the constraints encountered.

6: Conclusion and Perspectives, this final chapter gives an insight on the research objectives by a general conclusion and opens a window to further study and improve-

ment of our work, by explaining the technical limits, and how to be solved.

Chapter 2

Related Work

This chapter is divided into two parts. The first part provides an overview of the generalization techniques, with a comparison between different techniques, which form a basis of the generalization approach proposed in our work. The second part explains the direct and standards based visualization, with a focus on the visualization used to enhance the distinguished buildings. We will also discuss the interest behind both the generalization and visualization techniques, and on the other hand, their limitations. The proposals are aimed at optimizing the final visualization for the great amount of data on the web.

2.1 Generalization of Buildings

Different approaches were used to handle the generalization. Some were directly focused on managing some of the standards representing the city, such as CityGML [BAR13]. Other approaches dealt with the base of the building in order to simplify it while in 2D [HMM12].
A common constraint when dealing with large urban data is the high details of polygon meshes represented in the buildings. The idea of employing a generalization method in this work is actually to create multiple representation structures of the city model in different LODs. A generalization technique adapted will have the advantage of improving the visualization experience to the user, as it will be explained in Chapter 4.

A number of techniques for the generalization of 3D building models exist addressing specific requirements and challenges of building simplification [MF07]. Usually, 3D building models already have relatively simple geometry. Application of standard surface simplification methods from computer graphics often results in lost general

appearance and characteristics of the models, e.g., orthogonality and parallelism. In addition semantic attributes need to be taken into account during simplification. The following works represent approaches to handle these characteristics. The approaches explained below are divided into *Single Buildings Generalization* and *3D City Models Generalization*.

2.1.1 Generalization of Single Buildings

Single building model generalization techniques process a building model independent from its environment. In some cases, inconsistencies in visualization and topology need to be managed in a post-processing step, such as in the work of Peter et al. [PHF07]. The building generalization technique presented in Kada 2002 [Kad02] simplifies boundary representation model or BRep, using a set of rules. First, a number of constraints between faces are set up to ensure that the buildings characteristics are maintained, such as parallelism, orthogonality, and symmetry. Then, surface features are detected and simplified. Finally, the simplified model is optimized towards the original model using Least Square Adjustment.

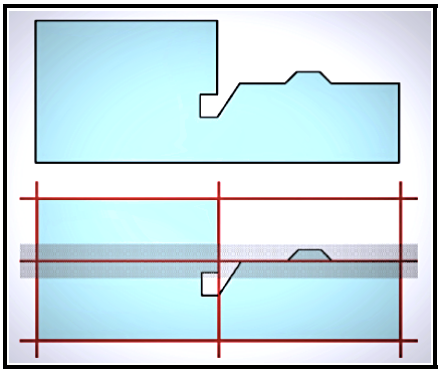
Using Half Spaces concept, Kada [Kad06] also proposed the reconstruction of a building model for each wall surface of the original model. The algorithm works on a plane and a related buffer: by starting with the face having the largest area, it merges faces within a given maximum distance to the current face buffer, adapting the plane parameters and leading to a smaller number of planes. The final planes are used to create a cell decomposition of the building. The cells which not cover a given percentage of the original building area are discarded. The remaining cells are used to extrude 2.5D geometry. To complete the simplification, roof faces are reconstructed separately for each cell using half spaces and finally merged to the resulting generalized buildings.

Rau et al. Present an approach working on building models comprised of prismatic shapes with sloped roof structures. First, the roofs are flattened and adjacent polyhedrons are merged if their height difference is smaller than a given feature resolution, yielding 2.5D shapes[RCT+06]. Then, the footprint of each shape is simplified by collapsing small edges (walls), and the vertices which belong to the longest edges are not allowed to move in order to retain the principal structure. Feature resolution is the minimum distance by which to control the simplification process. It is used at runtime to make the selection of the appropriate representation for visualization.

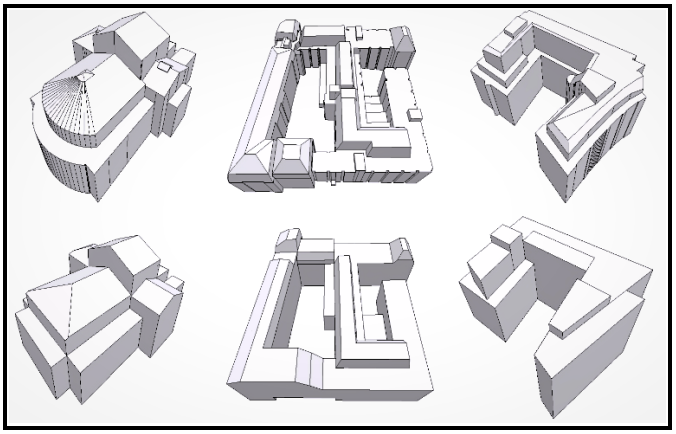
Thiemann in 2002 [Thi02] focus on decomposition of a building model into its principal shapes, following the approach of Ribelles et al. [RHG+01] in the previous

year. In the spirit of hierarchical segmentation. The model is decomposed into smaller parts using planes built from the model's faces. A quality parameter then controls the selection of intersection planes and the order of their application. The resulting shapes together with a boolean operation are stored in a constructive solid geometry tree (CSG). The minimum shape size can serve as a parameter to control the selection of displayed building components for a simplified building model.

A quite similar hierarchical tree-based approach is described by Ripperda et Brenner [RB06] using a formal grammar to describe the structure of the building, especially facades. According to their model, a facade can be decomposed into upper and lower parts, and symmetric parts. They use this description for the interpretation of a facade given in terms of laser scanner measurements or images. The grammar can, however, also be used to synthetically generate a facade. As it is described in terms of an hierarchical tree, it can also be understood as a series of representations in different levels of details. The advantage of using a grammar for the segmentation purpose is that it allows a direct definition of the appropriate generalization rules: e.g. an array of 4 by 3 windows can be generalized to a simplified array of 3 by 2 windows.



(a) 2D ground plan divided by six half-space primitives.



(b) 3D building objects in their original (top) and generalized shape (bottom) using half-space

Figure 2.1: Kada half-space generalization approach

Thiemann and Sester [TS06] worked on adaptive 3D templates, where they categorize building models into a limited number of classes with characteristic shapes. The initial building model is then replaced by the most similar 3D template that best fit the real object. The determination and selection of the templates can be pursued in two ways: on one hand, an appropriate template can be selected based on the attributes of the object. This is similar to 2D map case, where for example churches are assigned a

certain building symbol in a given scale. On the other hand, if such a semantic assignment is not available or, if a lesser degree of generalization is searched, the templates can be generated based on a simplified form of the original object, a basically main geometrically dominant shape.

Older 3D methods have been established. For example, Mayer in 1998 analyzed morphological scale-space and curvature space operators in the context of 3D buildings. Scale-space operators are defined as moving faces in or against direction of their normal, either uniformly by the same distance (morphological operators) or depending on the local curvature (curvature space-operators). The distance by which to shift the faces can be used as a parameter controlling the simplification. Further research in that direction by Forberg and Mayer [FM02] addresses orthogonalization as a requirement for effective employment of morphological operators and the identification of concave/convex structures.

Today, algorithms dedicated for 3D shapes are designed. Forberg [For07] adapts the morphology and curvature space operators of the scale space approach to work on 3D building models. His algorithm is based on an intelligent shifting of parallel building planes. Practically, it works by moving parallel facets towards each other until a 3D feature under a certain extent is eliminated or a gap is closed. Though this method works well for parallel structures, other methods have to be applied for non-orthogonal structures such as roofs. A set of rules was made to decide which part of the roof should be rotated, forced to either become vertical or horizontal. For the generalization of the roofs, the size of an individual roof face is the control parameter to decide if it has to be rotated. The moved faces result in merging building parts, removal of protrusions, or adjustment. The maximum distance by which to shift the faces can be used as a control parameter.

Cell Methods A structurally similar approach of that of THiemann has been proposed by Kada [Kad07]. He also decomposes the whole building into cells that are generated by the individual planes. After that, he aggregates adjacent cells, if filled with building parts. This process would normally result in the same representation. However, the difference between the two works is that Kada used buffered cutting planes to determine the cells. The size of the buffer is a direct measure of the degree of generalization. This method is first applied to the vertical faces and then to the oblique faces of the roof. It conclusively provides very convincing generalization of buildings, even complex ones. His simplification method to the roof structure was extended to use predefined roof types in [Kad09].

Another method including cells is the cell-based generalization technique, proposed

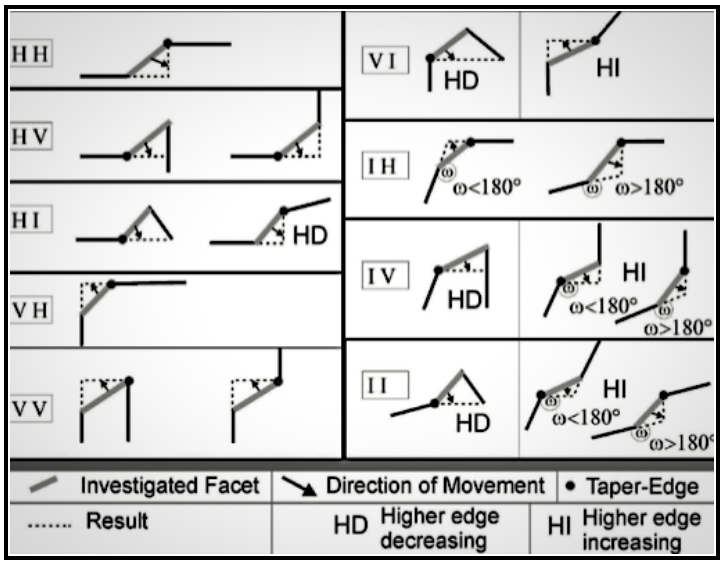


Figure 2.2: The rotated edge (grey circle) and the direction (marked by an arrow) depend on the relation of the inclined facet of a roof to its neighboring facets. Courtesy: Forberg et Mayer 2007.

by Glander and Döllner [GD09]. The cell-based generalization is a technique to create abstract representations of 3D city models. This abstract representation is focused on giving a quick overview about the general structure of the city. Similar to topographical base maps that serve multiple purposes, the resulting abstract 3D city models is intended to facilitate multiple purposes such as car navigation scenarios, tourist information, and efficient visualization of arbitrary thematic data such as air temperature and land use. That's why the base 3D city models is reduced in details to show only major structures. The work of Lynch [Lyn60] describes five major elements forming city images: paths, edges, districts, and nodes. Therefore, a comprehensive representation of a city has to incorporate and even emphasize these elements for easy comprehension, while allowing the user to connect real world structures with the displayed representation. Block cells represent individual buildings abstractly. The cell blocks are further shaped by computational geometry operations and enhanced by landmarks buildings, which are maintained in the visualization.

Adaptation In order to fit the simplified template to the original building, an adaptation process has to be performed. One of the adaptation processes suggested by the authors is to minimize the distances between surface points by discretization of the individual surfaces by points, find the closest distances to the second surface and minimize these distances. In order to have a good representation of the original face by the point sample, the number of points have to be of an adequate size. The higher number

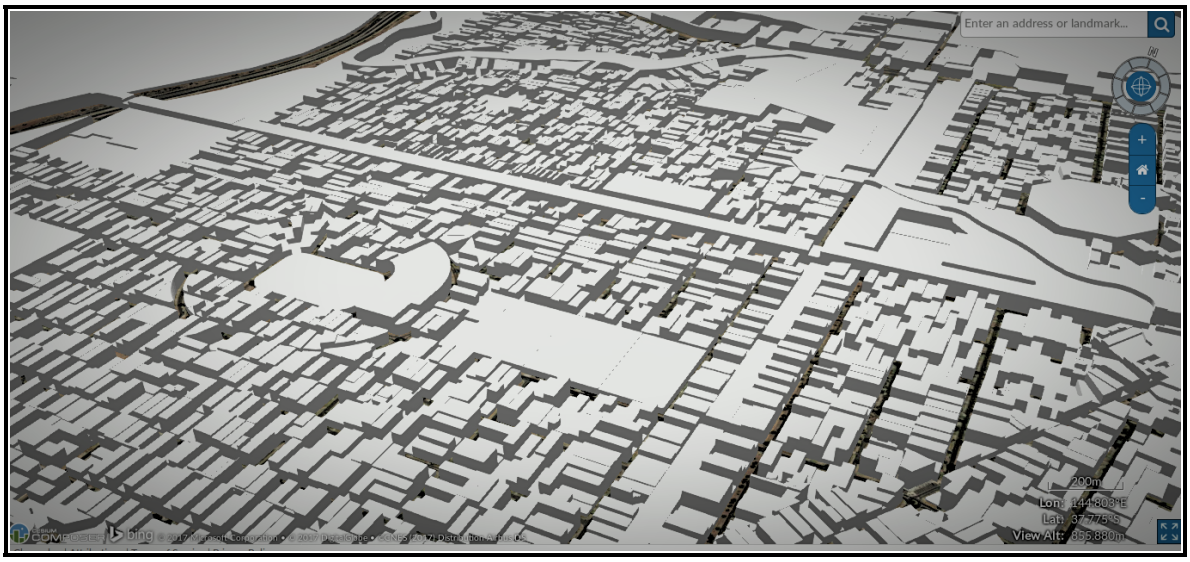
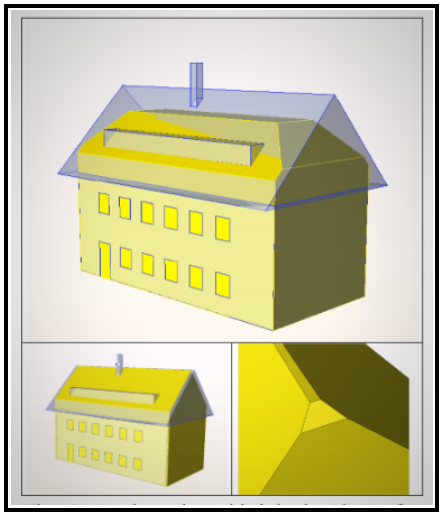
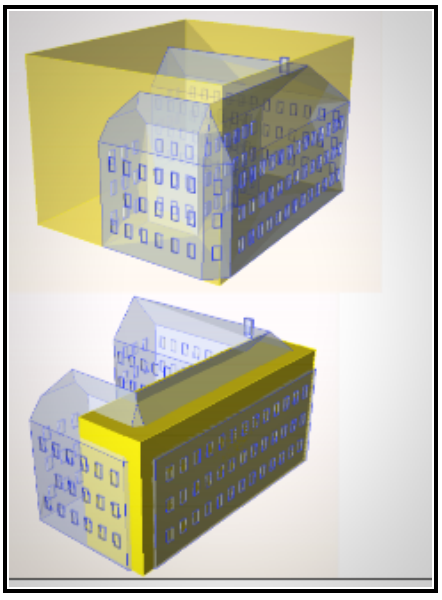


Figure 2.3: Abstract Visualization in a City Based on Cell Decomposition

of sample points is, the better the accuracy will be. Coors applied in 2001 [Coo01] an adapted surface simplification algorithm to simplify single buildings. Introducing dominance values in important parts of the building, e.g. a bell tower of a church, the simplification algorithm is adapted to conserve these parts while simplifying geometric complexity of the remaining model.



(a) Before and After Adaptation



(b) Template Adaptation of U-shaped Building

Figure 2.4: Generic template adaptation of two different buildings shapes

Hierarchical Trees Segmentation Hierarchical Trees Segmentation have been used as part of the simplification process on individual buildings. A segmentation of the building's boundary surface was established by Thiemann and Sester [TS05]. The idea is to segment the building's boundary surface for generating a hierarchical tree. The tree's elements are then interpreted and selectively removed or organized to implement elemental generalization operators for simplification. A conceptual framework for 3D building generalization is suggested by Guercke and Brenner [GB09] that is also based on a hierarchical feature model of a city and its components. For each feature different implementations of generalization operators can be selected, while a central coordination instance is responsible for solving conflicts. The semantical structure expressed by the feature hierarchy forms the basis for a generalization workflow presented by Guercke et al. [GBS09].

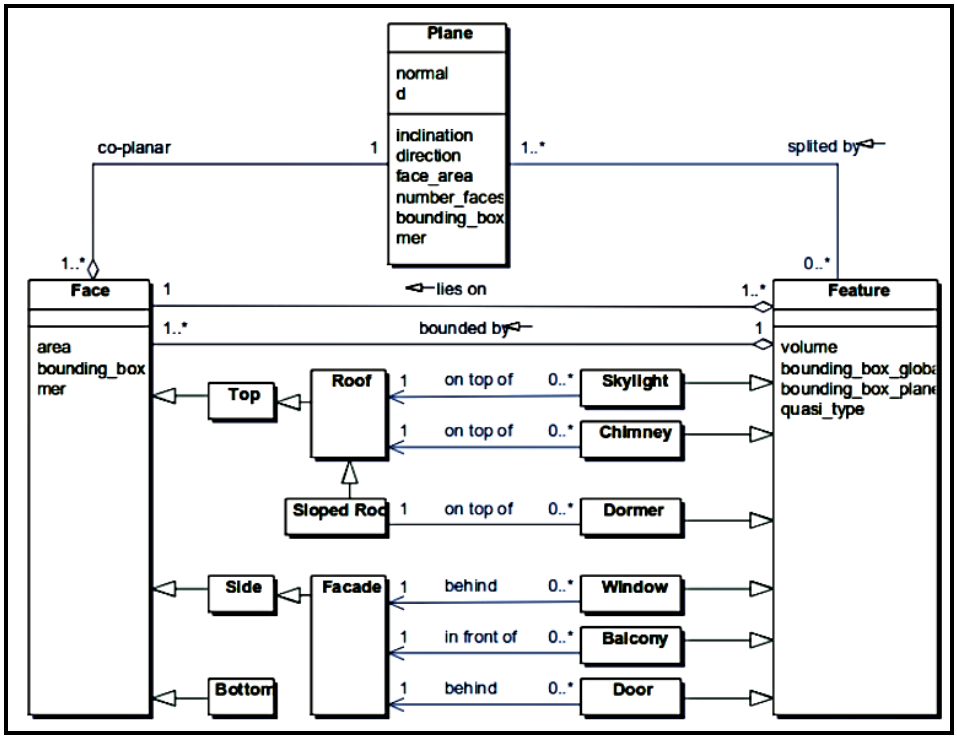


Figure 2.5: Hierarchical Tree of a Building. Courtesy: Thiemann and Sester, 2005.

Iterative Simplification of 2D Ground Plans In kada 2000, an analogous approach for the generalization of 3D building models is presented. The polygonal building models are iteratively simplified by combining a number of edge collapse operations into a single step. Building symmetries, that need to be detected prior to simplification, are maintained in the process. Footprint has been serving as the connection between 2D and 3D. Many block models of buildings were extruded from cadastral maps using

their footprints and heights. But more detailed models could not be acquired in this way. Therefore, a question that rises is: how can we translate 3D building generalization issues into 2D scope for generalizing more detailed 3D building models?

Several algorithms have been developed that remove line segments under a predefined length by extending and crossing the neighbouring segments and by introducing constraints about their angles and minimum distances [Reg01], [VK01a], [Har99],and [Wei96].

The table presents our personal evaluation of some generalization algorithms for 3D building models: 2.1 shows the capabilities of some generalization algorithms for 3D building models such as characteristics functions describing objects under assessment: feature extraction characteristic, continuous representation on different scales characteristics, and individual/group buildings characteristic.

Characteristics/ Criteria	**Generalization Using**		**Scale**		**Individual/Group**	
	Feature Extraction	**Semantic Information**	**Fixed**	**Continuous**	**Individual Buildings**	**Group of Buildings**
Sester	Yes	No	Yes	Yes	Yes	No
Mao, Ban, Harrie	No	Yes	No	Yes	No	Yes
Kada	Yes	No	Yes	No	Yes	No
Foreberg	Yes	No	No	Yes	Yes	Yes
Fan, Mang	Yes	Yes	LOD3-LOD2	No	Yes	Yes

Table 2.1: Evaluation of Most Used Generalization Algorithms

2.1.2 Generalization of Multiple 3D Building Models

Sester [Ses02] presented a 3D visualization of simplified, aggregated, enhanced, and displayed buildings by extruding the generalized building footprints to solids. As a use case, adaptive generalization is applied to building models depending on their distance to important building models. Anders aggregates linear groups of building models, such as along a road. The proposed algorithm first projects the building geometry onto its principal axes, yielding three sets of footprints. Then for each set, the footprints are aggregated and simplified using 2D generalization operations [And05].

Finally, the simplified footprints are extruded and the resulting shapes are intersected, yielding the final aggregated building model. Similarly, Tsai et al. [TLC12] aggregate compact building groups by projecting and rasterizing their models along the principal axes. After performing morphological operations on the images, the top raster image is segmented according to the side images. A simplified 3D shape is finally reconstructed by identifying vertices and edges, creating polygons from the raster

images. An approach especially addressing the grouping of LOD1 building models as a prerequisite for aggregating them is presented by Guercke et al [GZBZ12]. Properties of the individual buildings, e.g., relative and absolute height, are taken into account to decide which neighboring buildings should be aggregated. The decision task is formulated as a mixed-integer programming problem that minimizes conditions, e.g. the difference between the summed original and aggregated building models' volume. In contrast to heuristic, greedy algorithms, the resulting building groups are optimal with respect to the given conditions.

2.1.3 Generalization Based on CityGML Standard

The approach of Fan et al. is directed at generalizing CityGML LOD3 building models. In their approach, all the polygons that belong to one wall are projected to the farthest of its polygons' planes, polygons that are not parallel or coplanar are discarded. Thus, the facade structure with window and door openings is maintained. Similarly the roof polygons are clustered by their orientation and projected to the farthest plane of each cluster's polygons.

One of the optimized ways of representing a model is by a discrete and continuous levels of details representation of *polygonal* and *triangular meshes*, in the aim of reducing storage space, shortening geometric computations, and improving rendering performance. The triangle-based boundary representation is the most common representation of a 3D object in the context of real-time computer graphics. It is a composition of one or more triangle meshes, formed by multiple triangles in 3D.

During the simplification process, the triangle meshes should be managed in a way that preserves the geometric accuracy as good as possible. The generalization algorithms used so far usually show quite satisfying results, despite the geometric error metric. Some are applied on highly complex objects, made from hundreds of thousands to millions of primitives.

Problems could occur where some datasets are too complex to fit in main memory and must therefore be processed out of core [Lin00]. Several objects that make up the building block exhibit certain characteristics that need to be preserved during simplification, such as right angles.

Also, the geometric error metric as used in surface simplification does not account for such characteristics, usually because the simplification operators were developed

for smooth surfaces rather than angular 3D shapes and therefore their application to a 3D building model often results in a skewed or tilted model.

A similar kind of problems occur for the automatic simplification of 2D building ground plans. The line (e.g. the algorithm of Douglas and Peucker in 1973 and revisited in 2011) [DP11] does not yield very good results for such objects. Since then, specialized generalization algorithms have been proposed. Sester [Ses00] proposed a work on 2D ground plans by the application of a set of simple rules followed by a least squares adjustment. Focusing on creating simplified LOD models, Guercke

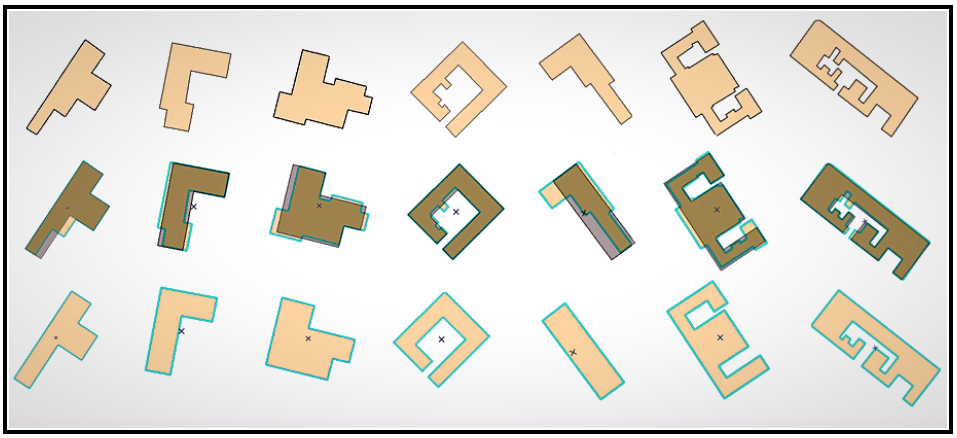


Figure 2.6: The Least Square Adjustment Applied to 2D Buildings Shapes. Courtesy: Bayer 2009.Automated building simplification using a recursive approach

et al. as well [GBS09] suggested a typification technique using the *Least Squares Adjustment*, a Bézier surface approximating the original surface is computed. Then the tile distribution (e.g. the number of rows and columns) is detected. For the selection of the optimal LOD model, perception measures are suggested by the authors that adapt the tile density in the LOD models to the depicted size.

2.1.4 2D/3D Buildings Generalization Algorithms: Study and Comparison

In order to choose the best generalization method for hiding unnecessary details during visualization, we carried out a study of 3D city models generalization algorithms. The study focuses on the works rather dealing with simplifying the geometry of buildings by focusing on their 2D footprints, and specific roof shapes. Our idea of generalization starts with the 2D, meaning that the buildings footprints are first simplified and aggregated according to the method adopted by Sester and Brenner in their work [SB05]

and extended by Fan et al. in their book [FLM09]. They focused in their algorithm on two parts: the first part was about the simplification of ground plans and the removal of intrusion and extrusion. The second part deals with roof structures adjustment with respect to the ground plans. Moreover, they considered height, width, and positions of facades and facade accessories at different LODs.

In block level, a number of methods have been developed for aggregation of 2D cartographic objects and 3D buildings. Bundy et al. [BJF95] introduced in 1995 Direct-merge operator and Snap-merge operator, which are two aggregation operators to move two objects near to each other. The direct-merge operator maintains the alignment of the objects by moving the objects together directly. The relationship between the facing edges is represented by the triangles that have an edge in one object and a point in the other. Figure2.7 below shows an example of direct-merge.

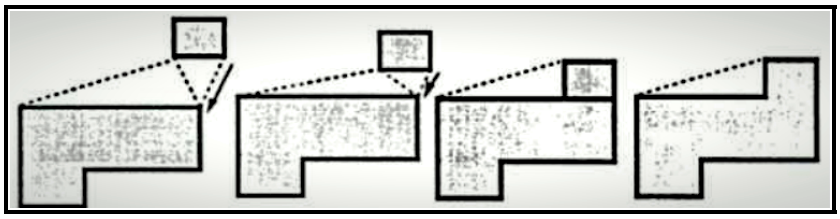


Figure 2.7: Direct-merge operation. Courtesy: Bundy et al. 1995

Meanwhile, the snap merge align the objects' nearest parallel edges. It can be achieved by aligning the merge vector with the shortest outer connecting edge, as shown in figure.

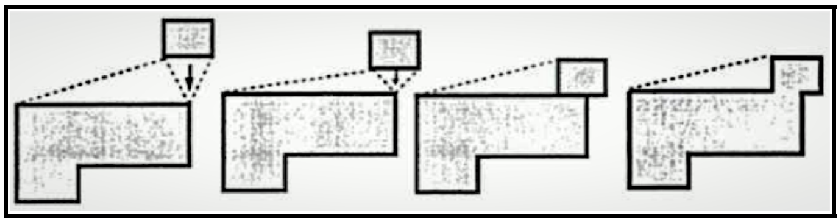


Figure 2.8: Snap-merge operation. Courtesy: Bundy et al. 1995

Another method developed by Anders [And05] is quite suitable for aggregation of buildings located on a straight line with same orientation. This method preserves the location of original ground planes, and creates the new ones by prolonging the existing one. He also introduced an algorithm to detect the grid structure of 3D building groups. The buildings' projections along height, width, and length should therefore be in a straight line or superposition. These types of constraints are considered a big

restriction for us, since most of the buildings in our environment are far away from being aligned. And this risks to make our generalization approach applicable on a very small scale.

For the single building model simplification, Mayer [May05] and Meng and Forberg [MF07] created a scale-space technique partly based on the morphological operators opening and closing to simplify 3D building model. Zhao et al. 2012 [ZZD+12] explored the generalization method for detailed building models in both appearance and internal structure with authentic architectural components. Fan and Meng proposed in 2012 a method with three steps to simplify 3D buildings, however applicable only in one format, which is the CityGML [FM12]. An ancient method used for generalizing 2D geo spatial data is to extend and cross neighboring segments to remove line segments of ground plan on a given both thresholds of the angles and distances. This idea was used by Sester and Brenner in 2004 [SB05], and extended by Fan et al. in 2009 [FLM09]. It will be adopted in our proposal with certain modifications. Some applied rules to buildings are defined in the authors' algorithm:

- **Rule 1**: A Building side smaller than a threshold should be removed. Important objects lying on the side, however, should not be removed.
- **Rule 2** : If more than one smaller parts consecutively exists, which are smaller than a threshold and make a pattern, then all of them needs to be treated collectively not individually.
- **Rule 3**: If the removal or modification affects the neighboring feature, or if there's an overlapping between two smaller features, than the two features should be treated together.

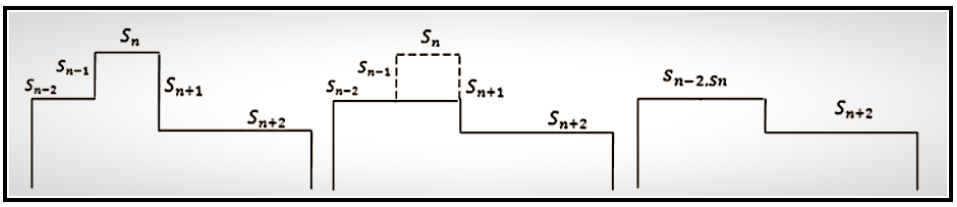


Figure 2.9: Removal of offset by taking neighbouring edges into account. Courtesy: Mao et al. 2010.

The figure represents a removal of offset for simplification of footprints according to Rule1 of Sester and Brenner. The building side S_n is smaller than the threshold and there's no important part of the building that lies on it. Other sides connecting with S_n are S_{n+1} ,and S_{n-1}, which are larger and parallel to each other, while S_{n-2} share

an edge with S_{n-1}. Therefore, the larger side $S_{n-2}$is extended and vertical side S_{n+1} is reduced, resulting with the new side $S_{n-2,}$ S_n.

Existing generalization algorithms are restricted by the data format of input building models and different models (explicit or parametric). The approaches presented above are only applicable to a single building model.

2.2 3D City Models Visualization

After giving an insight that led us to propose a building generalization method for the buildings in our city (to be explained in Chapter4), we are going to focus

3D city visualization combines both the disciplines of visualization and real-time rendering. It is considered as a special case for data visualization. The definition of rendering was given by Schumann and Mũller as being part of the visualization process [SM13]. In computer graphics, rendering means the creation of synthetic images fast enough so that the viewer can interact with a virtual environment. Rendering in games and simulation is calculated and displayed in real time, at rates approximately 20 to 120 frames per second.

In real time rendering, the goal is to show as much information as possible as the eye can process in a fraction of a second (or in one frame). Rendering for interactive media, such as games and simulations, is calculated and displayed in real time, at rates of approximately 20 to 120 frames per second. In real-time rendering, the goal is to show as much information as possible as the eye can process in a fraction of a second (a.k.a. in one frame. In the case of 30 frame-per-second animation a frame encompasses one 30th of a second). The primary goal is to achieve an as high as possible degree of photorealism at an acceptable minimum rendering speed (usually 24 frames per second, as that is the minimum the human eye needs to see to successfully create the illusion of movement).

In the case where a city model contains geometry specifications and appearance information, the terms "Visualization" and "Rendering" are frequently used synonymously. However, a distinction should be established between the two. The reason is that sometimes the actual appearance of a city is not always the primary aspect of interest, like in the case of urban development, where the geometry of the city model serves basically as a medium to visualize spatially referenced thematic building information which is not necessarily visible. Another reason is that direct rendering of a model might not necessarily provide actual insight.

Therefore, it can be used to amplify the resulting image perceptually by using non photo-realistic techniques. In the other hand, the effectiveness of interactive visualization is not determined by the rendering process only. It depends strongly on the effectiveness of the navigation techniques available, which control the way in which a user steers the virtual camera within the virtual space. To all of these reasons, city model visualization includes but is not restricted to 3D rendering. For an accurate visualization of big data sets, much information is often reduced. This lead to consequences like losing some of the data because many tools use single precision files, as well as losing the geo-referencing since most of the software have their own coordinate systems. This could also lead to unusable textures because of the control over the Level of Detail.

In recent years, many frameworks have been proposed for the visualization of cities purpose and listed by Bo in 2011 [Mao11a]. Another work by Rodriquez et al [RFC13] describes the implementation of a Web3DGIS based on the conversion of CityGML into X3D models and its visualization through the web. Another implementation of a visualization framework is presented in the work of Kramer and Gutbell [KG15]. On the other hand, Thick Client based on Nasa World Wind Virtual Globe has been extended to support the visualization of 3D city models [PDS+15]. This open source has limitations when compared against Google Earth. This last allows easy creation and placement of accessible, low resolution 3D models and is the most popular and best supported virtual globe tool.

2.2.1 Direct Visualization

In order to visualize a city in real-time, Beck [Bec03] directly emloyed OpenGL[1] to visualize 3D city models. In order to achieve the high speed, the framework employed the computer graphics related technologies such as pre-stripping the triangles, frame controls, texture management etc. The data is contained in a databse in self-defined 3D format and integrated with the textures. LODs are used to display several geometric representations of the same object at different times. Fewer details are represented when the object is far away and more detailed when it is closer to the observer.

Dollner et al. [DBB06] presented in their book a direct illustrative visualization technique to provide expressive representations of large-scale 3D city models. Their rendering algorithm consists of four phases: Phase 1 generates a texture encoding shadowed regions in image space, Phase 2 renders the scene with enhanced image-

[1]Open Graphics Library

space edges, shaded and textured facades, Phase 3 renders stylized edges, and Phase 4 renders remaining components of the 3D city model.

Although it may increase the visualization speed, the direct visualization framework has to deal with basic computer graphics issues such as frame control, scene rendering etc, which could increase development difficulties and workload. In addition, different versions of the visualization programs need to be developed for different platforms. Furthermore, the details related with the basic operation system and graphics interfaces may highly influence the portability of the visualization framework. For all of these reasons, our focus will be on the web visualization of 3D cities based on 3D standards.

2.2.2 Visualization using 3D standards

Many related-GIS standards have been used to reconstruct and visualize 3D cities on the web such as:

KML Standard Keyhole Mark-up Language (KML) is an XML based file format to represent geo-location on web applications such as Google Earth and Google Maps. Collaborative Design Activity (COLLADA) is also an open standard XML scheme to represent 3D models. With COLLADA integration, KML is being successfully used for 3D visualization on the web browsers. KML/COLLADA integration makes it possible to visualize CityGML data with the help of the software 3DCityDB.[2].

3DcityDB is an open source software, which allows to import CityGML dataset to a 3D geo database such as PostGIS 2.0 and Oracle Spatial. The major features of this software are the supports all kind of level of details and appearances, the support of complex terrain and building models. With the import of CityGML to a geo-database, it allows direct analysis of a city model. It also allows the export of CityGML from the geo-database. To visualize virtual cities from existing GIS data, many attempts have been made. For example, a virtual model of London was generated from 2D plans and supported in a collaborative virtual world [SF99].

Many 3D WebGIS applications are built around 3DcityDB. One of them aimed at retrieving and visualizing data according to some semantic characteristics [PD16].The 3DCityDB schema was originally installed, which supports the multi-scale and rich semantic structure of CityGML [KNH13]. CityGML data was chosen for representing the buildings in LOD2 and LOD4. In addition, storing data into DB was achieved by the use of 3DcityDB importer/exporter.

[2]3D City Database:http://www.3dcitydb.org

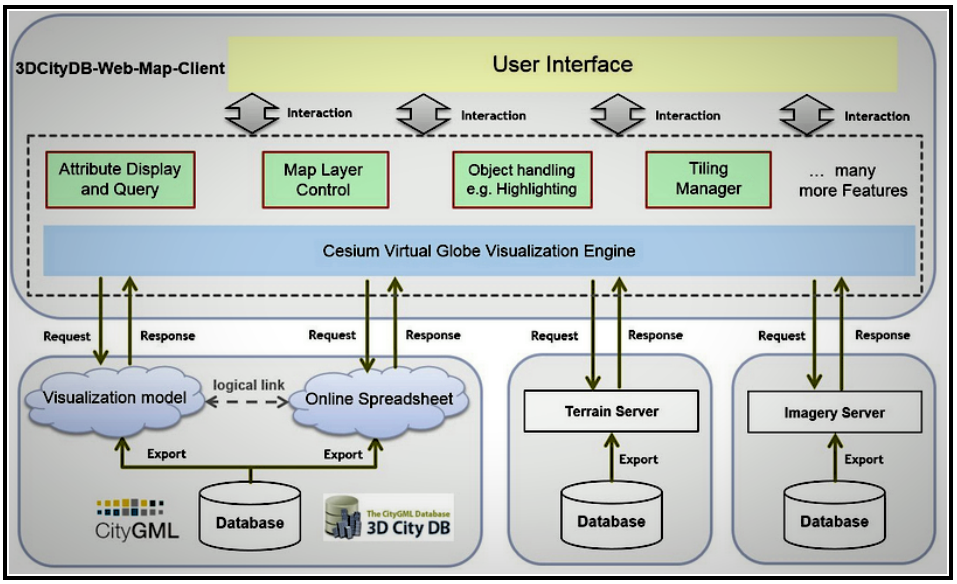


Figure 2.10: 3DcityDB General Architecture.

A more recent work by Byelozyorov et al. [BJPS13] demonstrated an open architecture to view models from OpenStreetmap data. Gaillard [GVB+15] also demonstrated generation of a web visualization of city models starting with CityGML as input. Also, a discontinued tool for city curation in Google Earth has been replaced by an automatic visualization of building models from aerial imagery. Although no clear reason exists from the decision of replacement, results of the automatic visualization look impressive when textured, but the undelying mesh, while sufficient for simple map visualization, is quite noisy and inaccurate. With the visualization proposal for our platform, we aim at reducing the time required to visualize buildings while trying to maintain a high quality level.

Cesium It is an open source Javascript library for world-class 3D globes and maps. It offers a time-dynamic content with a quite fast performance, precision, and ease of use. A number of open 3D geospatial formats have grown out of Cesium. Open formats used adapted to Cesium create interoperability for a full ecosystem of tools.
CZML, the Cesium Language, is a JSON schema for describing time-dynamic 3D scenes. Another format, the *quantized-mesh* enables efficient 3D terrain streaming and rendering. The Cesium team has worked closely with Khronos to create *3D Tiles*, a specification for streaming massive heterogeneous 3D geo spatial datasets.

2.2.3 Visualization of the Distinguished Buildings

Researches focused on the *Distinguished Buildings* touch several areas including visualization style and generalization context. In addition, adaptive maps and map-like representations include landmarks. Regarding visualization styles for landmark buildings, Elias and Paelke [EP08] analyze different graphical representations and introduce a design matrix to help choose the appropriate one for different categories of buildings, including visually outstanding buildings. For the case of perspective images, Lee et al. [LKPM01] suggest depicting distinguished buildings by placing photographs in the scene, which have been taken from a similar perspective. In the context of generalization, enhancing the important features is a job that belongs to cartographic generalization. The resulting conflicts due to a potential overlap with the surrounding features of the important buildings need to be resolved. Ruas used in 1998 [Rua98] the *Local Incremental Displacement*, whereas Sester [Ses05] focused on a more general approach, the*Global Optimization*. 3D geo virtual environments, as a specific class of map-like visualizations, need the distinguished buildings of the city to ensure that users can orient themselves and navigate. Detailed guidelines that we will be taken into consideration in Chapter4 for designing and placing the buildings in the virtual environment are suggested by Vinson [Vin99]:

- Landmarks should be visible at all times, especially at all navigable scales.
- Landmarks should be distinguishable from heir environment.
- Concrete representations of landmarks should be preferred over abstract ones.

In classical 2D maps, different visualization styles for landmarks are used, ranging from textural and iconic to realistic styles.

Depending on the current scale, landmarks can be depicted larger than their neighborhood [HGM94], highlighted by different colors or drawing styles, and exposed by clearing their immediate surroundings. Clearing surroundings generate some undesirable effects, for instance occlusion occurs which could hide important objects. In addition, having an interactive visualization of a 3D city model repeatedly changes the image, requiring constant attention of the user.

2.2.4 Smooth Visualization

To provide a smooth visualization, some concepts such as *Morphing* are applied. It is suggested as a generalization operator specific for interactive visualization that performs a smooth transition between different representations. Morphing is suggested

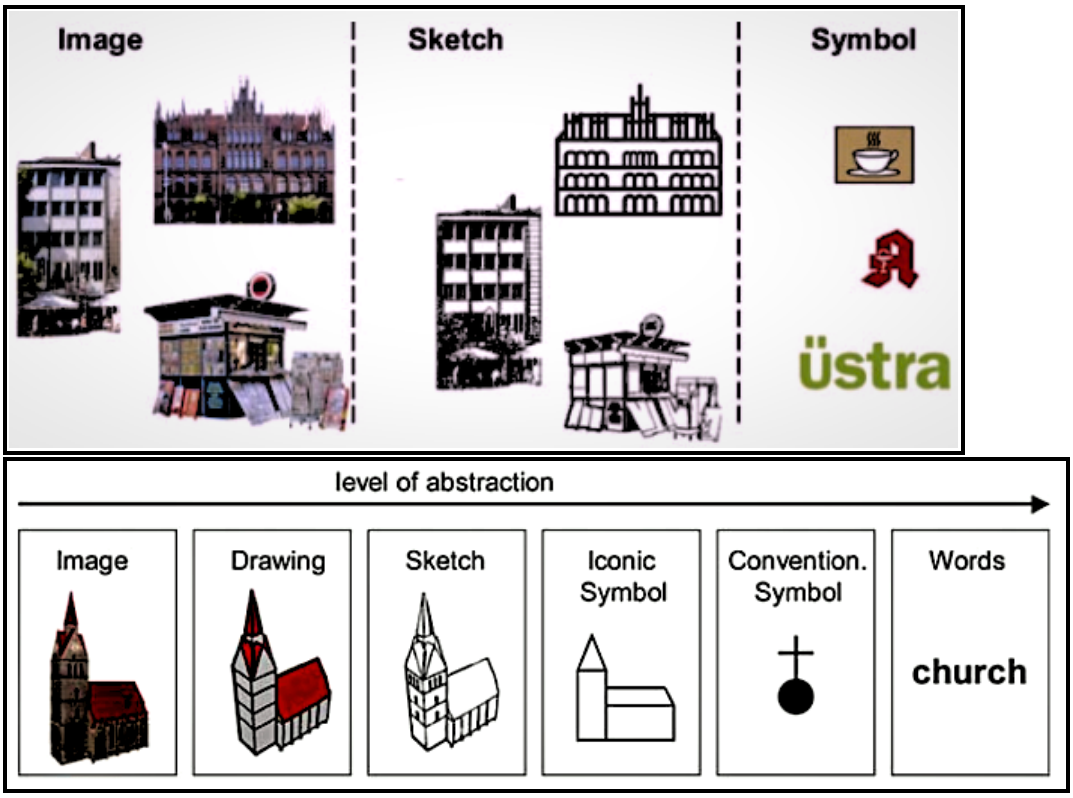


Figure 2.11: Visualization Styles for map Generalization. Courtesy: Paelke 2008

by many authors: Van Kreveld, Cecconi and Galanda. [VK01b] [CG02]. Semmo et al. [STKD12a] presented an interactive visualization system for generalized virtual city models base on image blending and multi resolution models and discuss geometric morphing as a means to transform city models from one level of abstraction to another.

As is it not reasonable and rather costly to produce models at different LODs, they should be generated with highest possible details and then automatically simplified to the required LOD as needed. Although the necessary techniques have been proposed in the context of height field rendering [Hop98], [LKR+96], [RHSS98], and mesh simplification [Hop96], [GH97], cartographic generalization methods for 3D building models are not yet capable of generating what is referred to as *Continuous* LODs, a collection of techniques that allows for the smooth transition between different LODs in both directions.

In Chapter4 , we will cover a concept of implementing a geometric landmark enhancement. The technique aims at dynamically highlighting landmark objects by considering the current view and applying deformation as a working principle. It resolves the problem of landmarks being occluded or too small. By scaling landmarks depending on the distance to the virtual camera, their visibility is enhanced. This enhancement supports the user in taking advantage of the landmarks and can be justified with their importance. In addition, surrounding objects are displaced as a consequence

of the scaling. The presented real-time rendering technique is generally suitable for applications which rely on visualization. Despite the implementation of methods and algorithms that we are going to develop in the following chapters, a manual pre and post intervention in our work seems unavoidable.

2.3 Discussion

In an interactive city scene, visualization and generalization are correlated: new visualization techniques may introduce other generalization considerations addressing specific challenges in 3D city models. The process of visualization forces the kind of data processing to make, and at which stage it should be done. Depending on the visualization technique of the city models adapted, a convenient generalization technique is applied. In the process of visualization, Ware describes in his book [War12], the different stages of the visualization pipeline. The four stages consist of:

- The collection and storage of data itself
- The preprocessing designed to transform the data into something we can understand
- The display hardware and the graphics algorithms that produce an image on the screen
- The human perceptual and cognitive system (the perceiver)

In the first stage, the original raw data is obtained by measurements or simulations is preprocessed, enriched, and selected in the filtering stage. This results in a derived, processed data. In the mapping stage, the data is prepared for visualization by mapping it to visual variables like color, size, and shape. The resulting mapped data is used to create images on the screen. The interactive visualization could be controlled by the user at several levels in different ways: through direct interaction (e.g. changing the view parameters), through mapping adaptation by selecting different colors, and by controlling the filtering stage, e.g. by loading a new data set or by changing preprocessing parameters. According to Glander et al. [GTD11], generalization can be applied within the visualization pipeline at all the stages mentioned above.

The presented decomposition incorporates five operators: *Selection, Transformation, Join, Generalization, and Rendering.* The concrete application of an operator on a given data is guided by the set of given visualization requirements that act as additional input to the operator. Each operator can be associated with a stage of the

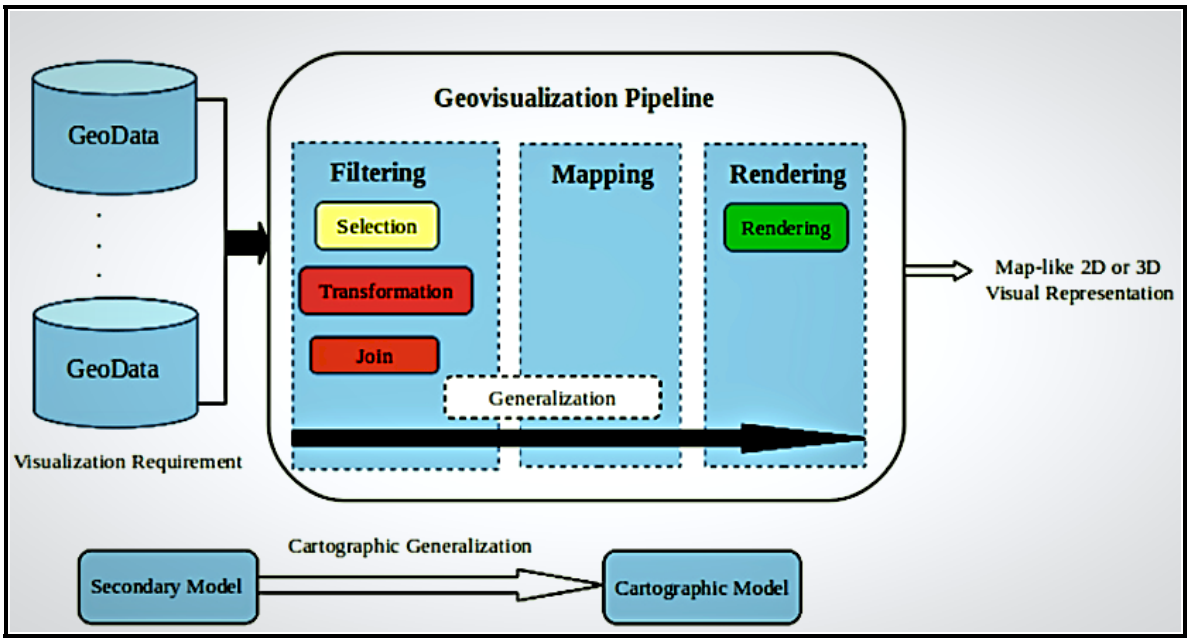


Figure 2.12: Map-Like visual representations from geodata visualization requirements

visualization pipeline, i.e. filtering, mapping, and rendering. The generalization operator is an exception. Some parts of its functionality are related to filtering(e.g. data abstraction), while others are related to mapping(e.g. assignment of visual variables). In general, it is not possible to split the functionality of the operator into a filtering and a mapping stage that are executed in sequence without iteration. This is because of complex interrelations that exist between the parts. The input geodata model is the secondary model, the intermediate output of the mapping stage corresponds to the cartographic model and the generalization operator corresponds to the cartographic generalization mapping.

Generalization operations such as *Selection, Aggregation, and Simplification* could be applied to the original geodata of a virtual landscape model, leading to a generalized one. Other operators such as exaggeration, enlargement, and displacement are only functional in case a cartographic landscape model is targeted. At the *Rendering* stage, generalization operations can be applied to adapt the graphical representation and enforce cartographical principles as the user changes the view parameters, e.g. camera position and projection. In practice, concrete generalization techniques uses a combination of these generalization operators. In particular, if they have to provide adaptive, dynamically generalized models for interactive 3D visualization systems, generalization operators for all the stages mentioned above are required.

In Generalization The presented state of the art techniques and the comparison established between some of them has made us synthesize clearly what technique is the most adopted to our needs and to the context of our work. Our proposal in generalization of some buildings which will be explained in 3 aims at optimizing the visualization on the web. The advantage of applying a general concept is to avoid the distinctive and individual generalization of each of the objects in the city, as it takes into consideration the similar building shapes and only one operation will be applied to each group. In addition, simplification on the level of the building alone is a long and not specifically important for the least detailed buildings. Also as we have seen, the presented methods above also maintain the general structure of the buildings well. Therefore, we will be reducing the geometric complexity without sacrificing the visual quality.

In Visualization The visualization method adopted in Chapter4 as well, highlights the importance of assuring a constant view of the landmarks of the city in each change of frame as the user moves into the scene. This keeps the hierarchy in the scene and also helps in the texturing process, as we will see in Chapter5. It also complements the previous visualization methods, which also discuss deformation of either the landmarks, or the surrounding buildings. Nevertheless, it has its inconvenient that is elaborated in 5. We will also evaluate the result of the buildings texturing, LOD management, and mapping by a salience map creation in 5

Part II

Design and Implementation

Chapter 3

Data Conversion and Simplification

Contents

In this chapter, we are going to explain the data management and the conversion process adapted in order to obtain the suitable city file to be visualized. We are also going to explain our 3D generalization method with different LODs. Since the LOD0 cannot reflect the full properties of 3D city models and LOD4 is not of interest for us in this thesis, we will focus on the generalization models in LOD1, LOD2 and LOD3.

3.1 Datasets Used

Before the city visualization, a reconstruction phase has to be done in order to merge and synchronize all datasets and formats as a one framework compatible with cesium import data for visualization purposes. For the reconstruction task, different formats of the city components are chosen. Having each city layer represented by a raw data gives a high flexibility to style and adapt the map in many different ways. The real

advantage of having multiple datasets sources is the availability of manifold points of interest (POIs). POIs are point locations which may be useful or interesting. Thus, the 3D City model can be enriched with numerous items of information, making the 3D map generated much more informative for the user [FOJ04].

3.2 Data Management and Conversion

The objective in this phase is to prepare the files retrieved from different sources and representing the city layers in one single platform. A manual work is mandatory for synchronizing the datasets. For the data fusion, our work is focused on the city of Moscow, which was chosen due to the large availability and ease of access of related geographic data, and the quite rich architectural models of the buildings. The input and output data of 3D city layers of Moscow are represented in the following formats:

- *Source Data: OpenStreetMap(.osm) Extract*
- *Intermediate Transformed Data: KML(.kml), and COLLADA(.dae)*
- *Output Data for Visualization Use: glTF files(.glTF), and glb files(.glb).*
- **Source Data: osm**: A sample area of Moscow was downloaded from OSM ready extracts[1]. The OpenStreetMap file covers the buildings layer of the city, represented in footprints: a 2D file containing features and representing the buildings outlines. An extrusion operation on these outlines is needed. The input buildings OSM were successfully imported in Blender using the importer plugin, extruded, and then exported as a COLLADA file.

  After converting the OSM into the COLLADA file, we moved on to the second iteration of conversion. At this stage, we are going to keep the buildings assembled in one group and test the visualization in Cesium, without any kind of optimization, or texturing. Splitting the file into small group of buildings is crucial at a later stage, the reason is because the models will be more adaptable to processing such as simplifying and editing.

- **glTF** This format is very promising for the future 3D standards for high-performance visualization of massive models on the web. It is the GL Transmission Format, which is the open-standard runtime asset format for WebGL, OpenGL ES, and OpenGL from Kronos group [RC14]. However, this format requires adaptation and scaling to standards for massive models, such as terrain, points clouds, and

[1] download.bbbike.org/osm/bbbike/Moscow/

buildings in our case.

- **COLLADA2GLTF** CesiumJS has made available an open source component for 3D model conversion. The COLLADA2GLTF model converter: is a model converter developed by KronosGroup, that allows to generate a glTF 3D model starting from a COLLADA file. It also supports textured 3D models. There are two output files of the COLLADA2GLTF: the first is a file system-based collection of glTF 3D models, one for input building. The second is a JSON file, that is essential for ensuring the geo-reference of each component stored in it. The purpose of the Json file is to describe the node hierarchy, meshes, materials, and how to access the binary block. The accessors which constitute a part of the Json content tell the parser how to read and interpret the binary data and how to extract data required for rendering triangle meshes. It also allows for creating larger data chunks and for optimization techniques that will greatly improve rendering performance The conversion from Collada to glTF was eventually successfully established, and the city was visualized in Cesium.

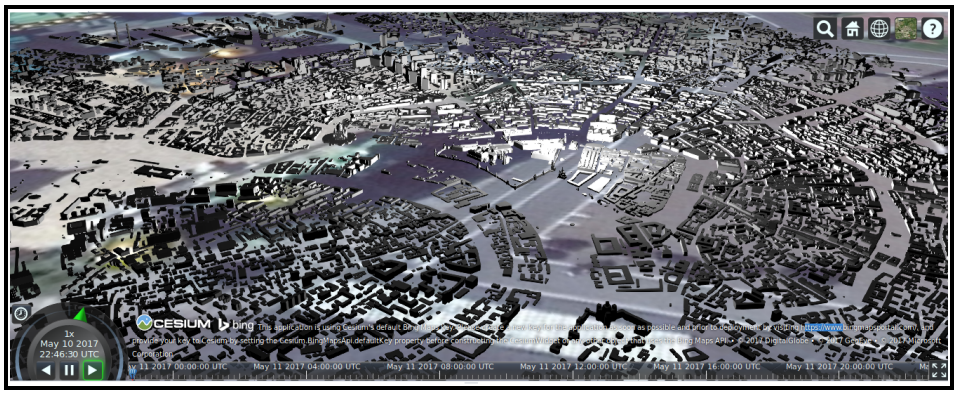


Figure 3.1: First test of a converted glTF file visualization of Moscow

.

The conversion output formats is obtained using the OpenSource converter: COLLADA2GLTF.

3.2.1 Creating Chunks

Streaming the city using one glTF file is very constraining due to the number of polygons to be processed at once in Cesium. this is why the file needed to be reduced into many smaller files. For this, the software "Meshmixer"[2] is found to be handy, and

[2]http://www.meshmixer.com/

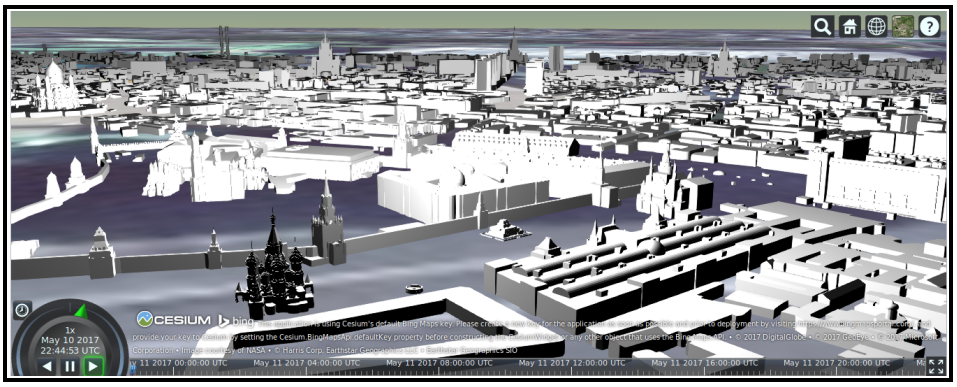


Figure 3.2: A more zoomed view to the distinguished buildings

.

easy to use for this purpose. The software was created by Autodesk, and allows the creation of mashups for an already existing 3D models. The function "Plane Cut" is used to chunk the 121.5 MB obj file through a cutting plane. The cutting plane could be adjusted, transformed, and oriented according to the user's needs.

This process led to six chunks of the obj file, which were exported from "Meshmixer" into COLLADA files for a further conversion to glTF, the same way we did with the "Non Chuncked" COLLADA file of the city. The slices have the following file sizes: Slice1: 14.9 MB, Slice2: 10.7 MB, Slice3: 4.2 MB, Slice4: 6.2 MB, Slice5: 12 MB, and Slice6: 28.9 MB. Below are the chunks obtained from the random cuts executed,and imported into 3Ds max. They are visualized respectively with their bounding volumes in the perspective view.

As we can notice from figure, some of the chunks bounding volumes interfere with each other. This is normal since some of the buildings are quite close to each other, or have a certain spatial relationship.

3.3 Data Simplification

Usually, cartographic generalization is carried out manually by cartography experts. With the development of GIS related technologies, the volume of data for 3D city models is increased drastically. Different researches have been done on 3D generalization in order to reduce the complexity of 3D city models [3]

The generalization of the data could be applied in any stage of the visualization

[3]Open Geospatial Consortium OGC:http://www.opengeospatial.org/standards/

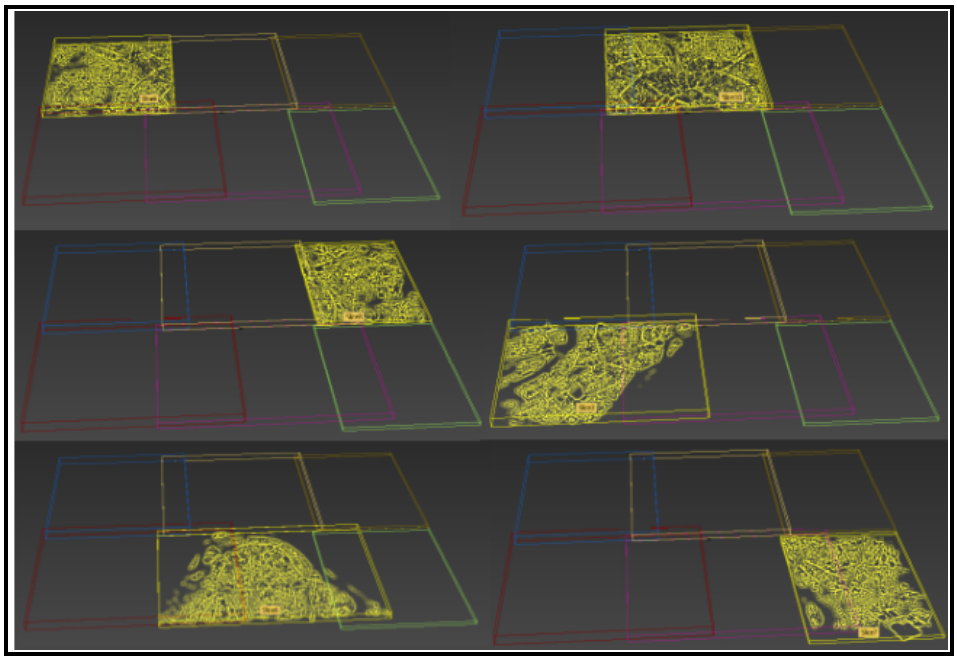


Figure 3.3: The different chunks created with their Bounding Boxes.

pipeline *the Filtering, the Mapping, and the Rendering*, see 2. In our case, the generalization will be applied in the rendering phase and will be adapted to our visualization strategy.

3.3.1 Our Simplification Approach

Our simplification approach of the buildings consists in generalizing the buildings according to the method explained in Chapter2. We need to provide adaptive and dynamically generalized models for the interactive 3D visualization. The challenges are discussed in accordance to our system. Other visualization techniques may address other challenges and therefore, other generalization techniques.

To start with, we are going to use an algorithm with generalization in order to restrict lines, curves, and corners of the distinguished buildings ground plan. This is done with the help of*MathWorks libraries* implemented in Matlab. Working with the ground plan of the building is a practical approach when it comes to generalization, first because we can deal with the outer structure of the buildings, and second the generalization rules are applied as 2D rules, which makes them easier to compute. Footprints of 3D buildings may contain corners, offsets, and intrusions or extrusions. That is why we are going to fix certain criterion and conditions in order to facilitate the elimination of these unwanted details:

- Geometry-Based rules inspired from the CityGML specifications[4]
- Threshold value: if the considered edge is shorter than this value, then it is removed. Else, it will be extended to the neighbouring larger edge.
- If the removal or the modification affects the neighbouring feature or features, they should be treated together.
- If two buildings are separate by a distance of 10m or less, these buildings are merged or aggregated.

We are going to consider the different cases and shapes that we encountered in our buildings and how they are dealt with:

Removal of Shortest Edge The criterion concerning the edge removal (the second item) is shown in one of the buildings below. The difficult part when removing the shortest edge is to adjust the neighbouring edges, while maintaining the length, orientation, and association with each other.The process is repeated under other smaller edges are removed. The simplification process is shown in the algorithm 1

Simplification of Curved Shapes Similar to lines or polygons, building footprints could contain curves. The best way to deal with this kind of shape is to divide it into nodes, then relate these nodes with straight lines. Then, the lengths are compared to the threshold value.

Aggregation The aggregation of the buildings close to one another is established in the same spirit of the framework proposed by Mao et al [Mao11b], who applied the aggregation by connecting 2D footprints of neighbouring buildings using the *Trimmed Convex Hull*. The aggregated height is then stored in order to support reconstruction of extruded 3D shapes for visualization.

3.3.2 Reconstruction of the Simplified Building

The ground plan simplification shape result and the roof structure simplification shape result are merged as part of the reconstruction of the resulting building. Each wall is considered as a polygon of 4 vertices and its heights are increased until they touch the roof polygons. The polygon of a new wall can be acquired by removing the line segments above the intersection with roof.

[4]http://www.citygmlwiki.org/index.php/Specifications

Algorithm 1 Shortest Edge Removal

1: $Input \leftarrow \{poly(2, n)\}$
2: $Input \leftarrow \{numvertices\}$
3: $Output \leftarrow \{Finalpoly\}$
4: $numvertices \leftarrow \{length(poly)\}$
5: **foreach** $vertex \in numvertices$ **do**
6: CALCULATE(Vertex_Importance)
7: VERTEX_IMPORTANCE($v, poly, numvertices$)
8: ITERATE
9: **while** $numvertices > num$ **do**
10: REMOVE(vertex with least importance)
11: RECALCULATE(importance for vertices neighbouring the removed ones)
12: CALCULATE(Vertex_Importance$v_p, poly, numvertices$)
13: Clip Polygon to the final length num
14: Find adjacent vertices
15: Obtain adjacent line segments and lengths: $len1$= NORM($dir1$)
16: $len2$= norm($dir2$)
17: $dir1$ = $poly$(v_p,v)
18: $dir2$= $poly$(v_p,v_m)
19: Calculate the angle between the vectors and multiply by segment lengths
20: $acos(dir1 * dir2/len1 * len2) * len1 * len2$
21: return a

The generalization algorithms has been implemented and tested on the distinguished buildings in the center. It shows good results for simple models, and the more complicated ones. The advantage is that the complexity of the objects could be reduced without destroying the overall appearance of the building. Some of the buildings have erroneous symmetries. Even if they do not affect the topology of the resulting building model, they could be adjusted manually.

Our method should be extended to fit the quite complex structures of some of the buildings that could be found in the city. Subsequently, the generalization procedures were applied and tested on a dataset of buildings in the center. The reduction in the size of the generalized building models is calculated. The result of the data simplification is presented in Table 3.1 based on different threshold values. In addition, the graph shows that the percentage of data reduction is inversely proportional to the value of threshold. Moreover, the buildings located closer to the viewing point of a user seems less generalized as compared to the ones located far from user in street view.

Case	Footprint Original	Footprint Generalized	3D Model Original	3D Model Generalized
Vertex Number	361	156	970	725
Polygon number	44	20	230	97
Reduction Percentage	31	25	41.5	31.9

Table 3.1: Comparison of polygon numbers before and after generalization

3.4 The Simplified Distinguished Buildings Representation

The distinguished buildings, or landmarks, are crucial to facilitate orientation in the 3D city. In this section, we are going to confirm the utility of the generalization part by placing it in the context of a visualization scenario. It is also to improve the visibility of the generalized landmarks. The problem with the 3D rendering is that it sometimes implies perspective effects such as foreshortening and occlusion, which could hide the landmark objects. An algorithm adapted from Winter et al. [WTES08] creates a balanced landmark as follows:

(i) Triangulate landmarks using their centroids for a *Delaunay Triangulation* [DBVKOS00].

(ii) Compare each landmark's saliency with direct neighbors in the DT and vote for the highest saliency landmark among them.

(iii) Promote landmarks that have at least one vote to the next level.

(iv) Iterate until only one landmark is left. These interest values are computed for all visible and non-visible parts of a feature. After being normalized, these are used as blend values to compose the final image.

We start from the assumption that all of the*Distinguished Buildings* have a minimal size allowing the user to clearly see them. The idea is focused on increasing the buildings scale and thus reducing the space of the surrounding models. The scaling process will depend on the camera distance. It is also calculated in real-time based on pre-defined distance intervals for each landmark object. This scaling process could, however, result in an overlapping problem between the objects. Therefore, the *Surrounding Objects* are dealt with by applying the *Displacement* operation.

First, the input data consists of a part of the city or a 3D model chunk. This part is generalized using the aforementioned generalization approach. Features are rendered multiple times using different levels of details, which are continuously blended. To each level of details, weight values are assigned. These values are computed at rendering time using the three saliency metrics: *the viewing distance, the view angle, and the region of interest.*

As we have discussed in the state of the art of city models generalization, the systems use heavy algorithms to find a suitable solution for displacing objects. However, these algorithms are not applied to real-time visualization. As the user moves in the scene, the camera distance and the viewing angle change respectively. An algorithm that allows a computation of *Scaling of the Distinguished Buildings*, and *Displacing of the Surrounding Buildings* should be implemented. Another assumption that we are going to make is that in a camera view, the number of non distinguished buildings is much greater than the distinguished ones.

The implementation starts with taking the 3D city model as input, and augmenting it with weights defined per model during the tagging phase to separate distinguished from non distinguished buildings. In our proof of concept, we use the height as the landmark weight.

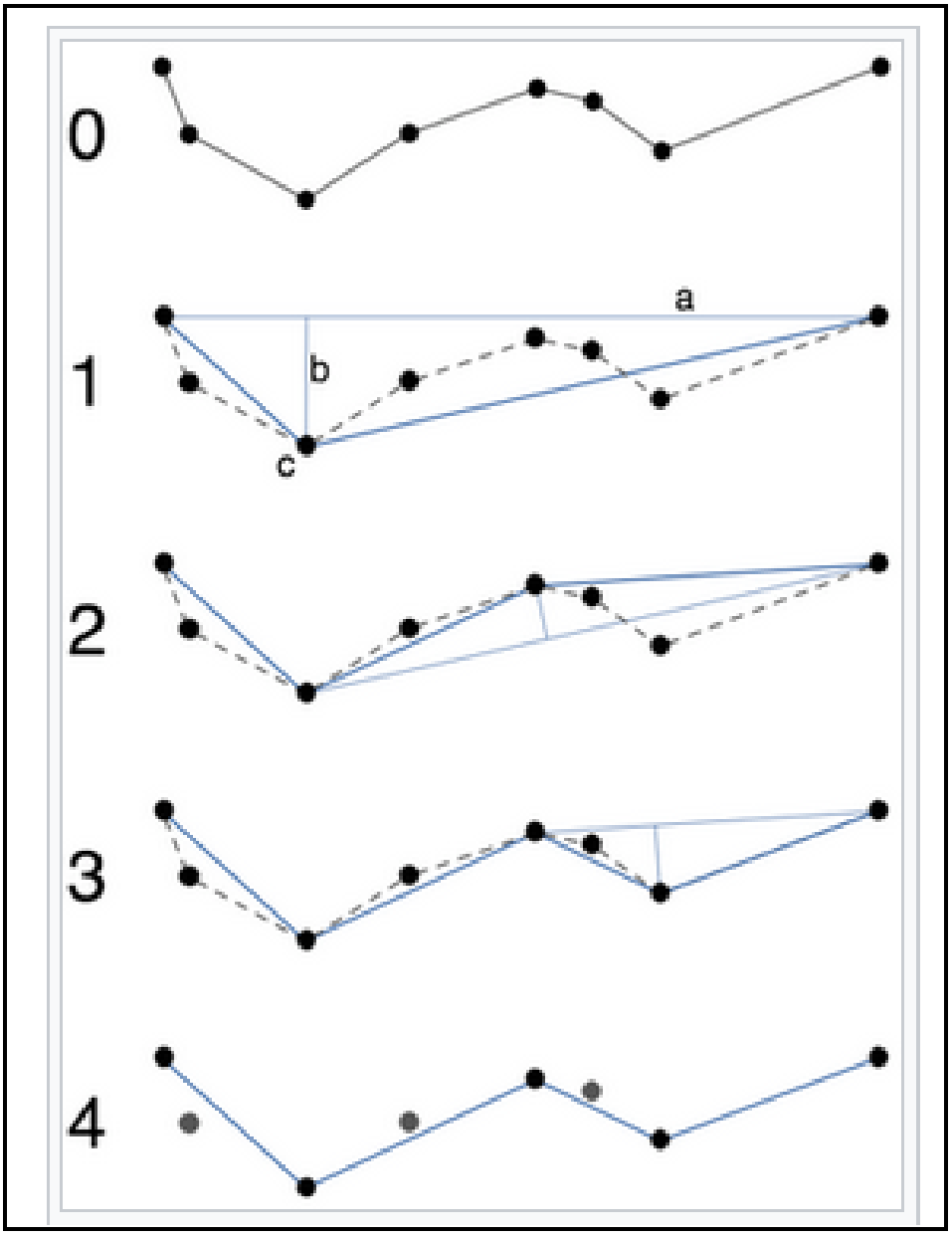


Figure 3.4: Curve Simplification explained

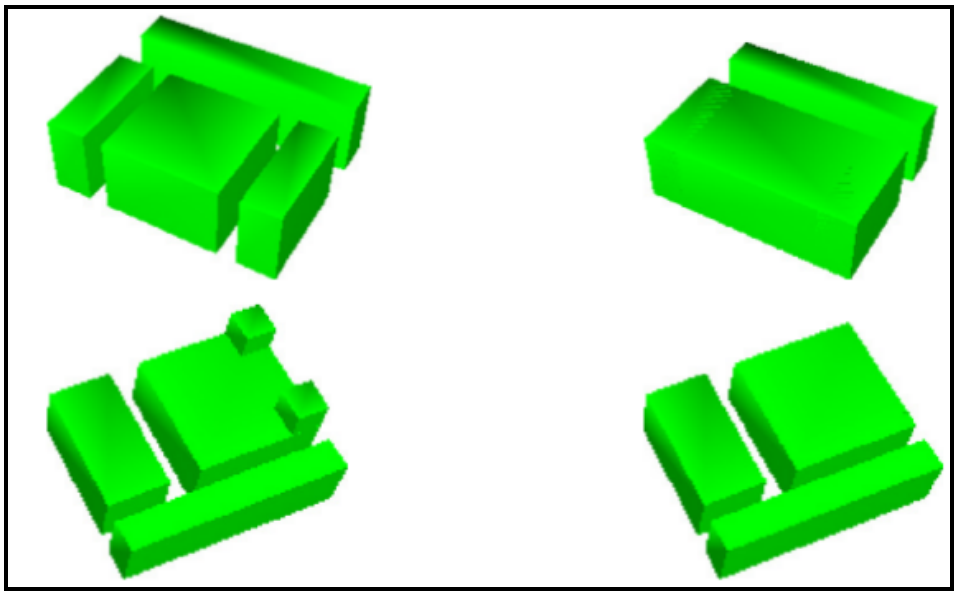


Figure 3.5: The process of aggregating 2 buildings

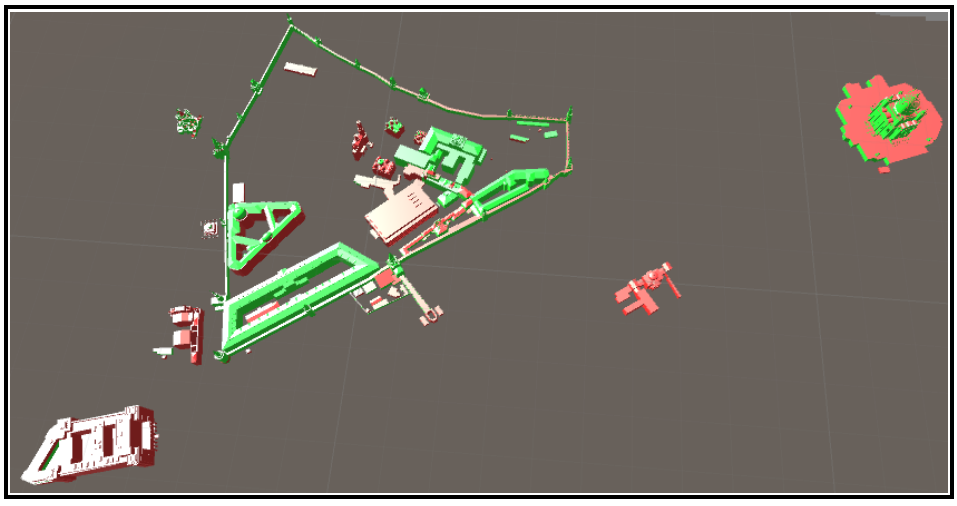


Figure 3.6: Buildings of the center to be generalized.

Chapter 4

Texturing in the Navigation Context

Contents

This chapter introduces multi-resolution texture atlases, an approach for real-time rendering of city models with high texture complexity in terms of number of textures and texture size. It describes the texture atlas tree, a data structure that forms the core of the approach, as well as the related construction process and real-time rendering. We will show that the triangulation of the buildings is very helpful when applying the textures, and the textures levels of details management utility in an interaction context. The efficiency of the approach is demonstrated in

performance tests. Finally, some improvements are explained to optimize the implementation.

4.1 Related Works

4.1.1 Texturing Techniques

Huge urban models often require textures that exceeds the main memory capacity. The common method for dealing with large textures requires subdividing a huge texture image into small tiles of sizes directly supportable by typical graphics hardware. This approach provides good paging granularity for the system both from disk to main memory and from main memory to texture memory. The problem of texture complexity has been addressed in several approaches:

Clip Maps of Tanner et al. [TMJ98] use dynamic texture representation that efficiently caches textures of arbitrarily large size in a finite amount of physical memory. Cline and Egbert [CE98] proposed a software approach for large texture handling. At runtime, they determine the appropriate mipmap level for a group of polygons based on the projected screen size of the polygons and the corresponding area in texture space. In another terrain viewing application, Lindstrom et al. [LKH+95] use the angle at which textures are viewed to reduce texture requests over using a simple distance metric.

In the context of texture management, Lefebvre et al. [LDN04] proposed a GPU-based approach for large-scale texture management of arbitrary meshes. The novel idea of their approach is to render the texture coordinates of the visible geometry into texture space to determine the necessary texture tiles for each frame. However the method requires a cost-intensive fragment shader and the geometry has to be rendered multiple times per frame. Carr and Hart [CH02] introduced a texture atlas for real-time procedural solid texturing. They partition the mesh surface into a hierarchy of surface regions that correspond to rectangular sections of the texture atlas. This structure supports mipmapping of the texture atlas because textures of triangles are merged only for contiguous regions on the mesh surface.

Frueh et al. focused on the visualization in the animation context [FSZ04] by describing an approach to create texture maps for 3D city models. The technique includes the creation of a specialized texture atlas for building facades and supports efficient rendering for virtual fly-through. Another approaches of textures rendering

were more restricted on the camera movement. For instance, the approach of Hesina et al. [HMT04] which describes a texture caching approach for complex textured city models, only in case of an interactive walk-through.

Hierarchical Data Lakhia [Lak04] proposed an out-of-core rendering engine which applies the cost and benefit approach of the *Adaptive Display algorithm* proposed by Funkhouser and Séquin [FS93] to hierarchical levels of detail (HLOD) [EMBI01] achieving interactive rendering of detailed city models. To support texturing, they store down-sampled versions of the original scene textures with each HLOD[1]. In other terms, the obvious method is to set the pixels of the output image to the average of the corresponding n*n blocks in the input image. The algorithm adjusts image quality dynamically in order to maintain a user-specified frame rate, selecting a level of detail and an algorithm with which to render each potentially visible object to produce the best image possible within the target frame time.

Another level-of-detail texturing technique creates a hierarchical data structure, such as in the work of Buchholz and Döllner [BD05] who presented a level-of-detail texturing technique that creates a hierarchical data structure for all textures used by scene objects (4.1), and it derives texture atlases at different resolutions. At runtime, their texturing technique requires only a small set of these texture atlases, which represent scene textures in an appropriate size depending on the current camera position and screen resolution.

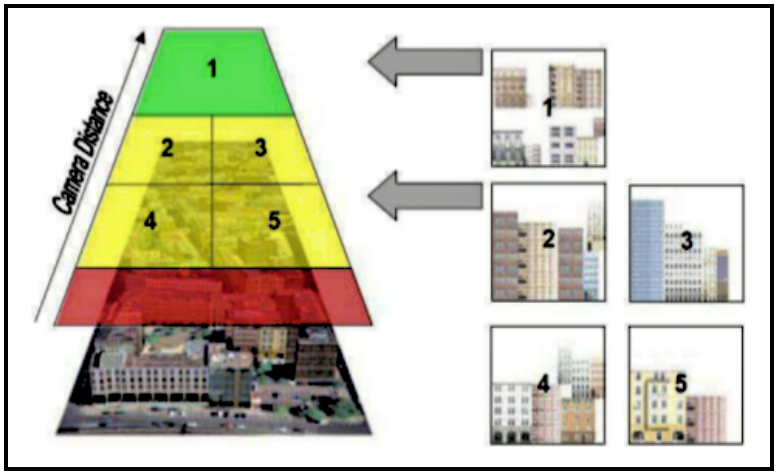


Figure 4.1: Applying textures with LOD consideration: original textures are used for the near area (red), four texture atlases are used for the center area (yellow), and a single combined texture atlas is used for the far area(green). Courtesy: Buchholz and Döllner

.

[1]Down-Sampling an image is reducing its size by multiplying the image by an integer factor n

4.1.2 Navigation in a 3D City Model

In geovirtual environments, navigation represents a key functionality in the interaction with virtual environments. Navigation and real-time rendering techniques are strongly related to each other in two ways:

- *Usability*: Effective navigation tools are essential to make real-time rendering techniques usable. If the navigation fails, an application is not used any longer, independent of the underlying rendering techniques. The reason a navigation fails, for instance, is when the user becomes disoriented in the virtual space and is unable to control the virtual camera to a desired viewing perspective.

- *Technical Requirements:* Freedom of movement provided by the user affects the demands on a real-time rendering technique, for instance:

  - If the navigation is restricted to slow movements, the corresponding small changes in the resulting images can be exploited to distribute the computational effort over several subsequent frames [Sch95].
  - If the navigation does not allow movements on the ground, restricted visibility can be exploited to optimize rendering by occlusion culling algorithm [BWPP04].
  - If the navigation enforces a minimum flying altitude, textures of a city model can be created at a lower resolution [FSZ04].
  - If the user can only navigate along predefined paths and can neither alter the viewing direction nor modify the environment, real-time rendering is not required at all, since everything can be provided as precomputed animations.

Navigation Techniques Different ways exist that allow user navigation in city models. The variety of different existing navigation techniques motivates the development of a rendering technique that does not depend on restricting assumptions on the navigation. A Navigation Technique Maps user inputs received from input devices such as mouse or keyboard to parameters of a virtual camera.

Navigation techniques have to achieve conflicting goals: on one hand, they should provide sufficient freedom of movement to allow the user to achieve all desired viewing perspectives, and on the other hand, a navigation technique should be intuitive, and, therefore, not too complicated to handle from the user's perspective. Controlling a virtual camera means to control at least five degrees of freedom simultaneously, taking

only camera position and viewing direction into account. Therefore, a navigation technique must provide an appropriate compromise between both goals. Correspondingly, a single navigation technique that is optimal for all applications does not exist and according to Ware [War12], *the optimal navigation method depends on the exact nature of the task*. In addition, different user skills and experiences as well as individual preferences of users are relevant [CCM00].

Examples of navigation techniques that are useful for city models are the following:

- *Pedestrian Navigation*: it supports exploration of a virtual environment from the point of view of a virtual avatar. Forward, backward, and sideward movements are performed by keys, while the head rotation and the walking direction is controlled by mouse movement. This technique has become quite popular due to its frequent use in computer games.
- *Fly-Over*: it is frequently used in geovirtual environments.
- *Trackball Navigation*: rotates the virtual camera around a focus point. For this, the mouse movement is mapped to a movement of the virtual camera along the surface of an invisible sphere. The focus point is determined by shooting a ray from the camera position towards the viewing direction and choosing the first intersection point with the environment(4.2).
- *Zoom Navigation*: is also based on a focus point and allows for controlling the distance between focus point and virtual camera, e.g. by mouse wheel or mouse movement.
- *Focus-Point Selection*: enhances trackball and zoom navigation by the ability to select a new focus point. The technique rotates the camera in a way that the focus point moves smoothly to the center of the screen during a short animation period.
- Panning Navigation: moves the camera parallel to the horizontal plane at a constant height. By dragging an arbitrary point on the ground along the screen, the camera is moved in a way that the currently dragged point follows the mouse pointer.

Pedestrian and Flyer navigation are well suited for presentation and entertainment applications. A combination of trackball and zoom navigation is commonly used for the investigation of single objects. In addition to the above mentioned techniques, there are also semi-automatic navigation in which the user navigation is interpreted to trigger on-demand camera animations:

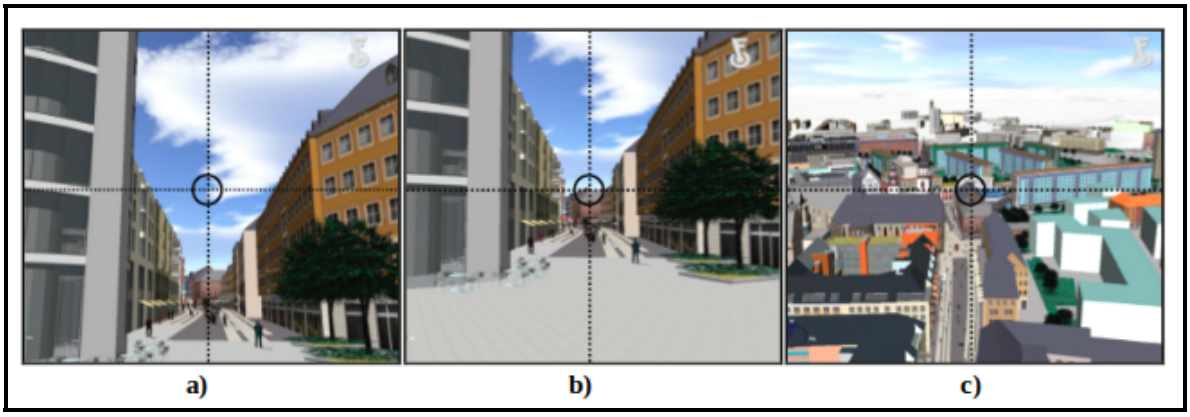


Figure 4.2: Image sequence illustrating the Focus-Point Selection. a) No focus point is defined, therefore the trackball is unable to work. b) The perspective has been modified by the adjustment strategy to obtain a valid focus point. c) The trackball can now be used for leaving the pedestrian perspective.

- *Landmark Selection*: invokes on-demand camera paths from the current camera perspective to predefined viewpoints [HS00], [SGLM03].
- *Path-Drawing:* allows the user to specify camera paths spontaneously by drawing strokes onto the screen [IKMT98]. The strokes are projected back into the 3D space, and a smooth camera path is invoked along the resulting curve.
- *Semantic-Based Navigation*: comprises a collection of navigation techniques for city models which enhance the path-drawing approach. These techniques initiate camera paths based on user gestures taking into account the semantics of currently visible city models entities. For example, drawing a stroke along a street followed by clicking onto a building is evaluated by flying along the street and finally rotating the camera towards the selected building.

4.2 The Problem: Interactive Visualization VS Textures Rendering

As we said before, there is a tight relationship between navigation and real-time rendering techniques. Many navigation techniques have been used in applications examples. An interesting implementation of these navigation technique is based on a navigation framework proposed by [BBD05]. Currently, large city models face a challenge between the **Interactive Visualization** and the **Textures Rendering**. The large amount of textures presented in the scene makes the visualization through the user interaction (walkthrough, flyover) less efficient and could be burdening.

They could only be rendered efficiently if the amount of textures is relatively small. Another problem that we noticed is the quality of the lighting in the scene. In most of the existing visualization applications, the viewer provides automatic creation of the lighting textures to improve the rendering quality compared to standard real-time lighting (Figure 4.3). Currently, these textures are rendered in a conventional way, so that the quality of the lighting is restricted by the available texture resolution. In addition, the required texture memory restricts its application to small models.

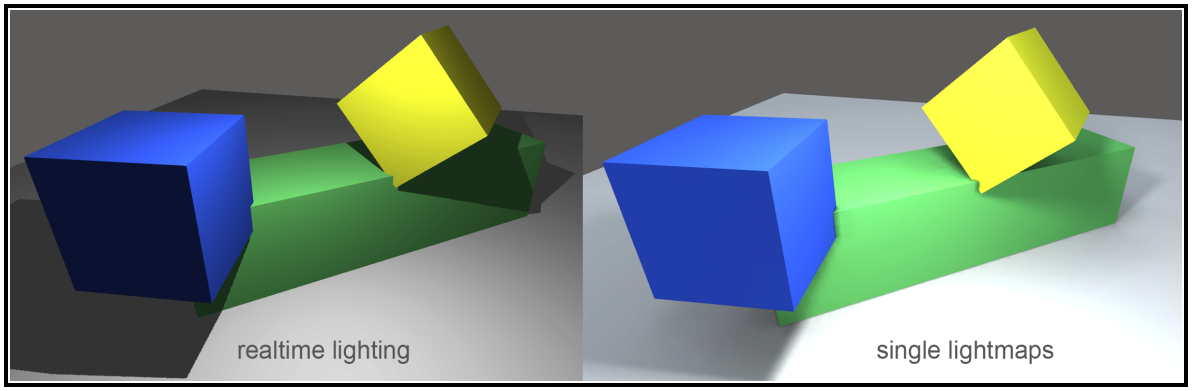


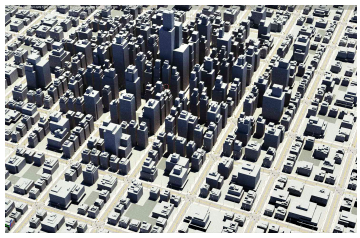
Figure 4.3: Comparison of standard real-time lighting (left) and precomputed lighting textures (right).

4.2.1 The solution: Necessity of Textures Optimization for Different Interactions

Optimizing the textures for rendering is essential in terms of reducing the number of WebGL draw calls, in other terms in reducing the number of meshes as much as possible. A texture atlas is an image containing a collection of smaller images, usually packed together to reduce the atlas size. An atlas can usually consist of a uniformly-sized sub-images, or they can consist of images of varying dimensions. A sub-image is drawn using custom texture coordinates to pick it out of the atlas. In an application where many small textures are used frequently, it is often more efficient to store the textures in a texture atlas which is treated as a single unit by the graphics hardware.

The texture atlas algorithms address the problem of two-dimensional image packing [CGJT80] and building a texture atlas from a set of rectangular images involves allocating this set of rectangular images into a larger rectangular image by minimizing the unused space. This is commonly known as the bin packing problem [LMM02]. The most efficient known algorithms use heuristics to accomplish results which, as good as they are, are not always the optimal solution. Three common algorithms for texture packing are used:

(a) Application using LandXplorer



(b) Lighting Simulation

(c) 3D Visualization

(d) Fly-over a city

(e) Berlin3D

(f) A 3D building viewer

Figure 4.4: Different applications of 3D city models

- The Basic: it divides the texture atlas into empty and filled regions. The first item (1) is placed in the top left, the empty region remaining is split into two rectangles along the side of the items. To insert the next item (2), we will check if it can fit in the empty space (above A), and left of B. If it is full (which is the case here), then the item will be on the right of B. If it fits on this place, B is exactly split like we split the original texture, otherwise, we insert and split below A. Doing this in a recursive way builds a binary tree representing the texture atlas. When adding an item, there is no information on the sizes of the items that are going to be packed after this one. This keeps the algorithm simple and fast. The items could also be rotated when inserted into the texture atlas[2]

- The Maximal Rectangles: the Maximal Rectangles algorithm stores a list of free rectangles that represent the free are of the bin (texture packing format). For splitting purposes, this algorithm performs an operation that essentially corresponds to picking two split axes at the same time. When an input rectangle R is placed in the bottom-left of a free rectangle F, the two rectangles F_1 and F_2 (that cover the L-shape region of F/R) are computed. The "Maximal" refers to the property that these new rectangles F_1 and F_2 are formed to be of maximal length in each direction, That is, at each side they touch either the bin edge or some rectangle already placed into the bin.

- The Polygon: like the name shows, this algorithm looks for the polygons in the 2D texture and cuts it around the external boundaries. It is particularly effective with textures having irregular polygons and complex polygons contents.

The Maximal Rectangle algorithm provides a fast but often the non optimal solution, involving placing each item into the first bin in which it will fit. It requires $O(n \log n)$ time, where n is the number of elements to be packed. The algorithm can be made more effective by first sorting the list of elements into decreasing order, although this does not guarantee an optimal solution, and for longer lists may increase the running time of the algorithm. Basic has the shortest generation time and produces good results. As for efficiency, TPIM[3] comes in the first level with the $(HighestusedArea/TotalSizeRatio)$. To see the three algorithms in practice, we packed the available textures for the buildings in the part of the city that contains the distinguished buildings three times, applying the algorithms explained above successively. This is what we get as result:

[2] http://blackpawn.com/texts/lightmaps/default.html
[3]Touching Perimeter Algorithm

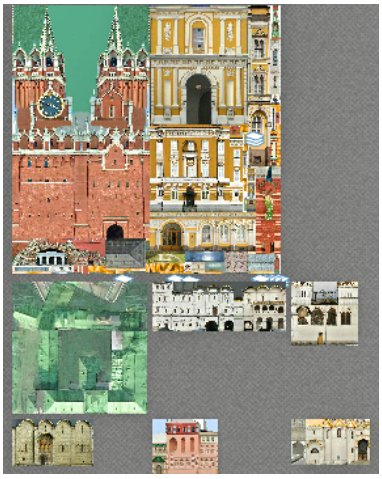


Figure 4.5: Polygon Algorithm

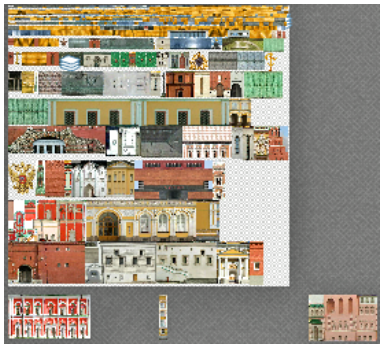


Figure 4.6: Basic Algorithm

Figure 4.7: Maximum Rectangle Algorithm

However, for a bigger range of flexibility and for designing texture atlases with different resolutions, we are going to implement our own texture packing method.

4.3 Our Proposal: LODs Textures Representation

Usually, LODs textures packing are commonly used to handle large textures in real-time, where the amount of simultaneously visible texture data can be arbitrary high at full resolution. As part of the optimization process we are carrying during the visualization, applying a technique that adjusts the texture LOD according to the building distance is very important. This will largely reduce the textures memory allocation since the texture regions that are viewed from large distances will only cover a small part of the screen. Hence, it is not necessary to use high texture resolutions for such regions. Based on these facts, the solution to handle large terrain textures would be to use high texture LOD only for terrain regions close to the camera and to decrease the texture LOD with increasing camera distance.

Our technique derives texture atlases at different LODs. At runtime, the texturing technique requires only a small number of texture atlases. The textures are represented in an appropriate size depending on the current camera position and screen resolution. This technique aims at solving the problem of uneven texture distribution in a 3D scene. Therefore, a focus should be shifted towards a scene subdivision that is based on the textures workload, and not only on the scene geometry, which in most cases, is established in the 2D representation of the scene.

4.3.1 The Data Parameters

To begin with, we need to construct a tree or a hierarchical representation of the texture atlases. In general, the input data from which a texture atlas tree is constructed consists of:

- An arbitrary 3D scene, given as a set of triangles G. Each triangle in G has 2 vectors: a position vector and a 2D texture coordinate vector.
- A set of related textures, T. Each texture is specified in the form of a 24-Bit RGB image. If it is not the case, the textures are managed in a different texture atlas tree rendered simultaneously as the first one (24-bit images).
- An identifier that assigns to each triangle in G a single texture in T.
- A minimum distance parameter d. d should be chosen small enough so that the geometry within a distance $>= d$ from the camera will be rendered with original textures.
- A ratio of pixel per texel[4] equal to 1.0 which is the ideal case. This value could be changed if it is necessary to trade image quality for performance. The higher this value is, the lower the resolution to be rendered.
- Finally, a texture atlas size defined by $W * H$ width and height. The best atlas generation occurs when $W = H$, and the size is a large as possible. We will show further that $W = H = 512$ is the most appropriate choice.

Creating the Atlas Tree In order to create a texture atlas, each node N or a group of buildings needs to be triangulated. The atlas is created in such a way that for a given distance d, there should be no magnification in the texture atlas, provided that the camera has a mimimum distance d from all triangles. d is chosen with respect to two criterias:

- It should be small enough, so that all triangles from a given viewpoint within a distance d are rendered with the **Original Textures**. In case of larger d values, the nodes should contain more full-resolution textures, which could slow the rendering performance.
- The smaller the distance d, the bigger the size of the tree data structure, and the higher preprocessing time.

[4]Textures per Pixel, the smallest element of a texture applied to a surface

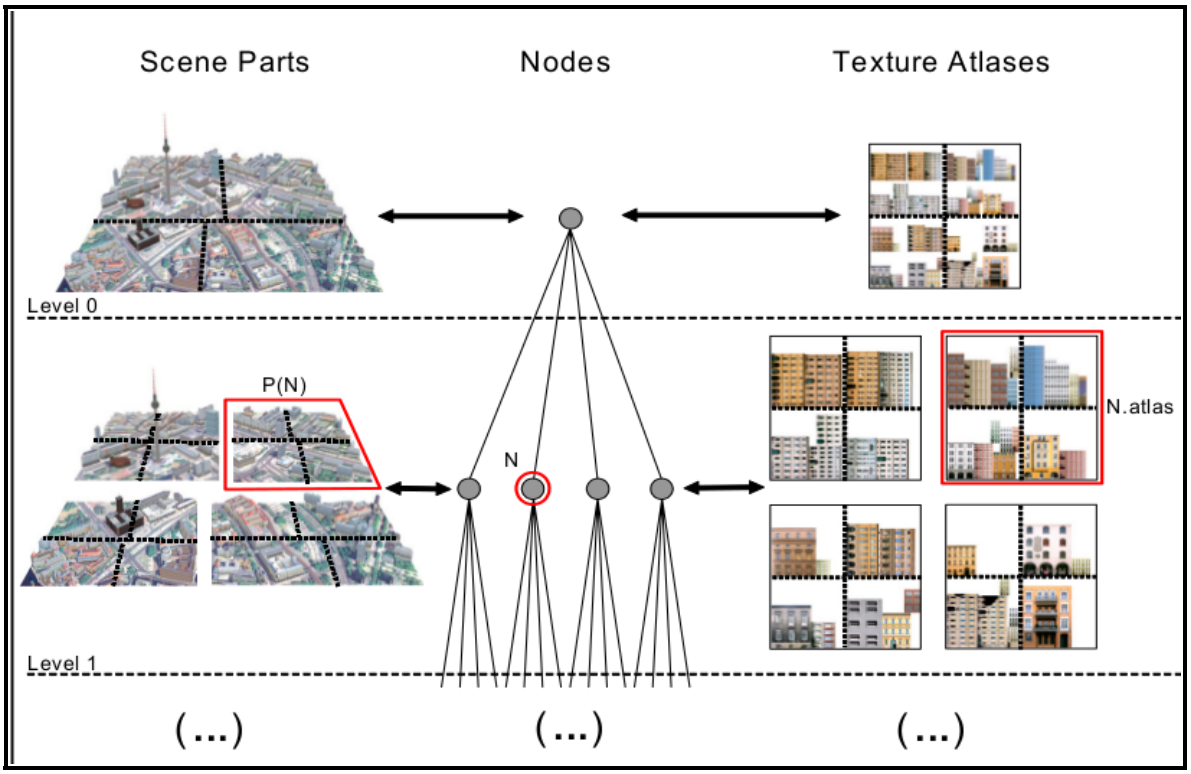


Figure 4.8: The nodes of the atlas

Now, we will use the predefined size 512*512 of the atlas and, for the calculations, a perspective camera projection is assumed with a defined angle of view α. Three calculations are established:

- For each texture, the minimum resolution at which the texture should be represented in the texture atlas to avoid magnification at a distance d.
- For each texture, the size of the required area within the texture atlas and in parallel the source area in texture space.
- The single texture atlas of the size A of the textures, found in the second step.

The basic structure of the texture atlas tree is shown in figure 4.9. The tree defines a hierarchical subdivision of both the scene and the textures atlases.

Texture Size Calculation Now that we have the original textures and the set of triangles that use these textures, we are going to manage the LOD of the textures to avoid the distortion at a camera distance of d. The obvious thing to do is to calculate each texture height and width required for each triangle in the scene. The maximum value of all the calculated widths will be the required width and the maximum value of

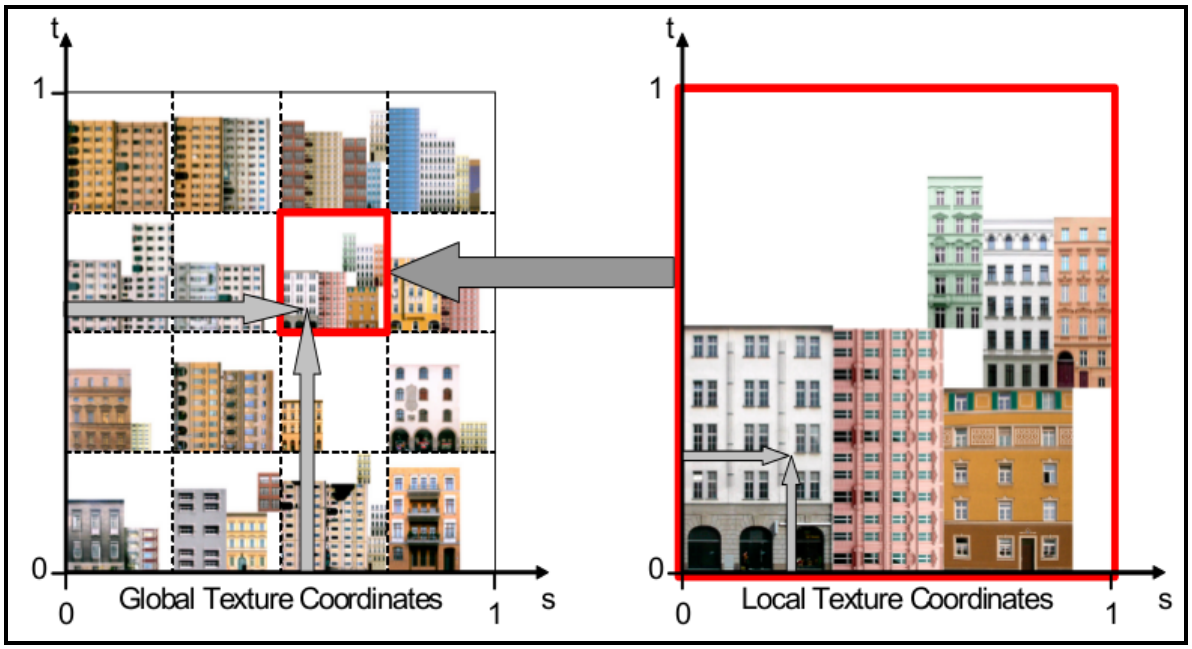


Figure 4.9: Local coordinates versus global coordinates

the calculated heights will be the required height. For a triangle T at a camera distance $> d$, the texture dimension is calculated according to the screen-space projections of the vectors $s(1,0)$ and $t(1,0)$ in the texture space. If the screen has a length of x pixels, the texture must have a width of x texels. In the same way, if the screen has a height of y pixels, the texture must also have a height of y texels.

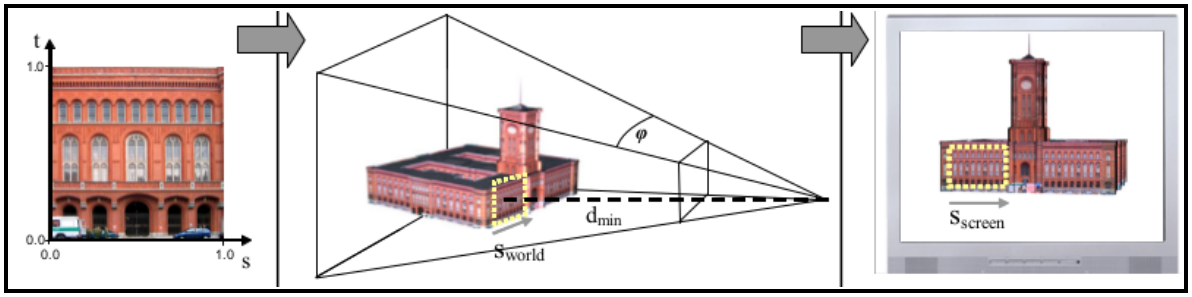


Figure 4.10

Node Triangulation The next logical step after the tree atlases creation is to render this tree, or in other words, apply the corresponding texture atlas to each node in the scene.

The triangles of the node have to be subdivided into four groups, one group for each node. For a good subdivision, we will try to meet two important conditions:

- The overlap of the group's bounding boxes should be small.
- The summed number of texture pixels for all required texture areas of a group should be approximately equal for all groups.

For establishing the triangle subdivision, we consider that the texture mapping is unique: for each texture, each point in texture space is mapped to at most one point in world space. Therefore, if the same area of a texture is mapped to two different triangles, these triangles are not considered for the subdivision. Also as a result of the unique texture mapping, the summed number of texture pixels for all the required texture area is proportional to the summed area of all the triangles in world space. That is said, condition b) can be substituted by the condition b') The summed surface area of all triangles of a group should be approximately equal for all groups.

The triangulation algorithm chooses two of the three main coordinate axes along which B has the largest bounding box formed by the triangle set to be divided. For faster implementation, we worked on small sets of the buildings to implement the triangulation method successively. We decided to implement the algorithm based on **Non-Constraint Delaunay Triangulation**. Reasons for this choice are the following: this type of algorithm is the most popular and most used in practice. It is also relatively easy to implement and can be modified for another types of triangulations. [LLL17].

- first, the research data is focused on a small set of 60 buildings, to better explain the process and effect of the algorithm. The algorithm starts by locating the centers of the buildings footprints. The connection between these points will create a large sets of triangles. Only the outside triangles are kept, and the inside ones (included in the building shape) are removed.
- Second, the triangulated set is split into two groups along the x-axis. The first group contains the first set of triangles with incrementing x coordinates, and the second group contains the remaining ones. The set of triangles are chosen in a way that the sum of their surfaces are equal in both groups.
- Another iteration of splitting is done, this time along the y-axis, to obtain another two groups, having a total of four groups which have equal summed areas.
- Usually, the splitting iteration process should stop when single triangles have a largely small surface compared to the triangle set to be subdivided.

We know that, for a given set of triangles, the iteration steps are successful if the texture atlas creation shows no errors in each of the resulting groups, meaning that the texture atlas area alone will not exceed the predefined maximum texture atlas size.

Algorithm 2 Triangulation Algorithm

1: **procedure** TRIANGULATION ALGORITHM(X_i, X_k)
2: $X_i \leftarrow$ list of points in the triangulation set
3: $X_k \leftarrow$ a set of seed points
4: **for** each item X_i in X_k **do**
5: $dist \leftarrow (X_i$ - $X_0)$
6: **return** *dist*
7: Sort the points in the set X_K by increasing X coordinates
8: **return** the three first points X_0, X_1, X_2
9: Link the points to form a triangle
10: Iterate for points X_i, X_{i+1} and X_{i+2}
11: **return** C
12: Flip the edge of the newly formed triangle and the existing neighbouring one if necessary

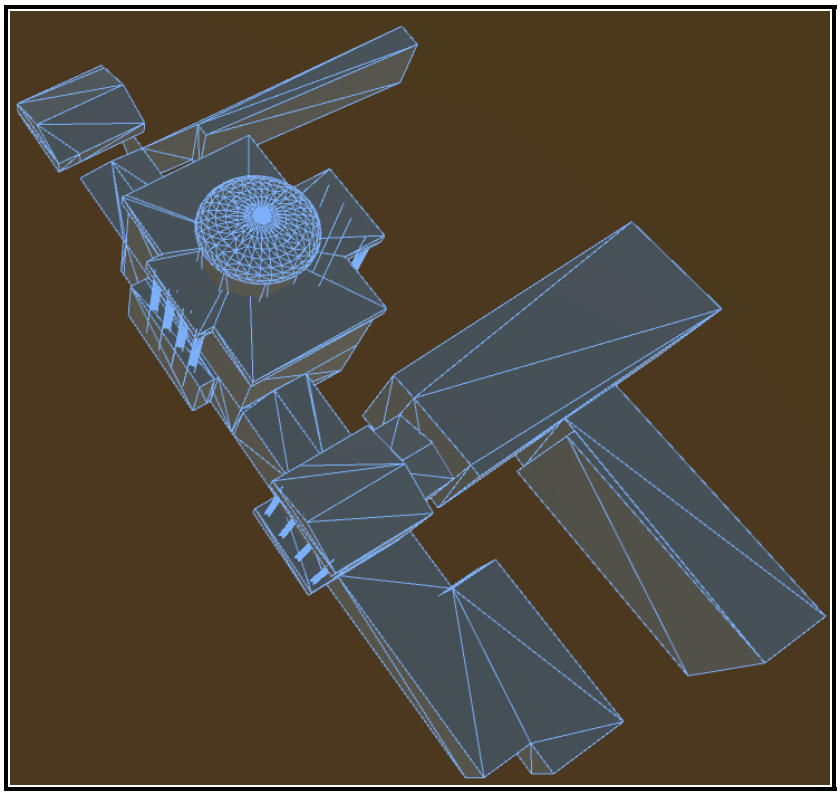


Figure 4.11: Result of Triangulation Algorithm for a complex building

4.3.2 Textures Mapping

Once we have all the textures packed into a single texture atlas and the nodes were triangulated, we want to wrap the texture with the triangles , including the self-repeating textures. The aim is to achieve a wrapping scheme able to compress efficiently the repetitive patterns. At this stage, it seems interesting to work with mipmapping: a pre calculated, optimized collections of images that accompany a main texture, intended to increase rendering speed and reduce aliasing artifacts [Wil83]. We used a specialized texture coordinate that compactly represents the textures. We first discuss the process of mapping of the original textures, a way to use mipmapping, and finally the requirements to well adapt the wrapping.

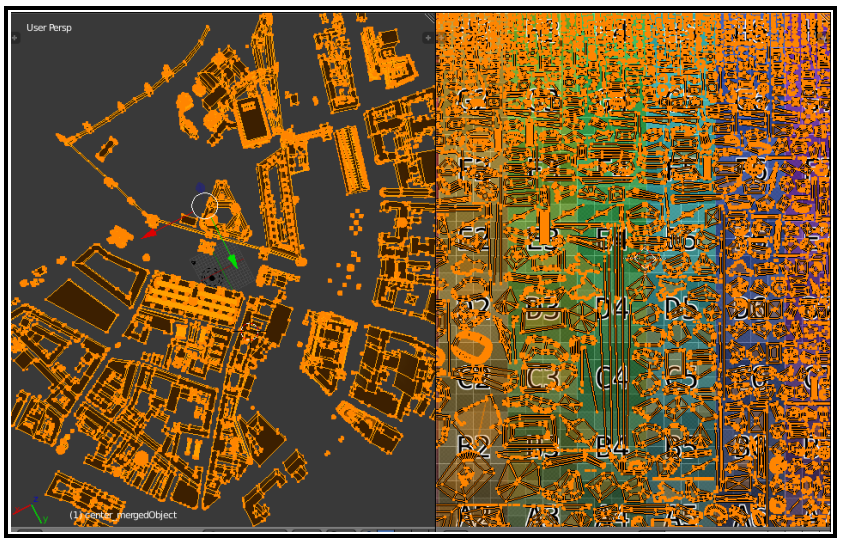


Figure 4.12: Mapping the textures to the model

Let us assume that a building, composed by several meshes, has a set of planes having this equation:

$$f(v) = \{\, (a, b, c, d) | ax + by + cz + d = 0 \,\} \tag{4.1}$$

The roof and facade planes were already clustered based on the normal of the plane. Assuming a plane P, the zenith angle angle θ of the normal is defined as :

$$\theta = \arccos(\frac{c}{\sqrt{(a^2 + b^2 + c^2)}}) \tag{4.2}$$

If the angle is less than 10 degrees, then this plane is defined as roof. The characteristics of the texture pixels can be evaluated by texture mapping. Triangle $a(V_4, V_5, V_6)$ has three vertices in the model mesh, and the triangle $b(u_4, u_5, u_6)$ is the corresponding

in the texture image. The texture coordinates of the vector V_4 for example are (x, y), ranging between 0 and 1. The coordinates of u_4 are (u_x, u_y)

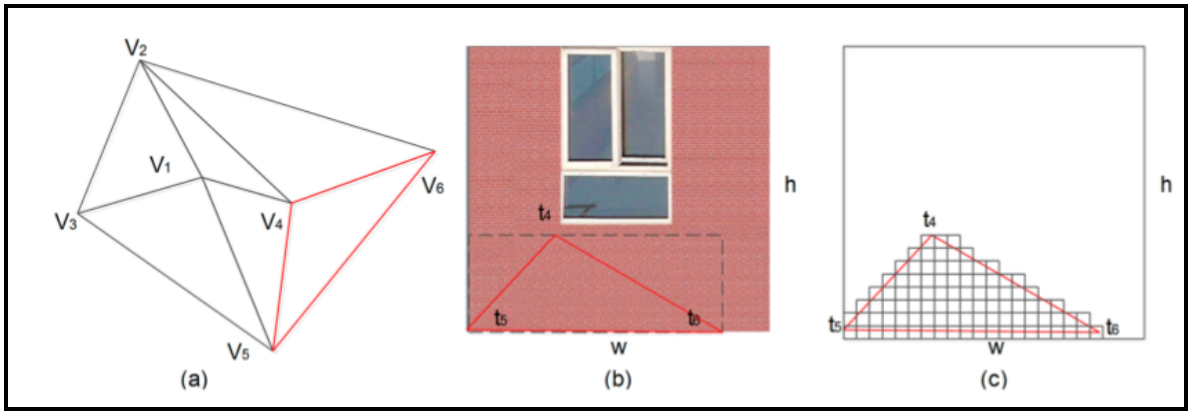


Figure 4.13: The concept of mapping

4.4 Rendering with LODs: the procedure

Rendering scene parts very close to the camera at full texture resolution frequently consumes a significant amount of the overall rendering time. Therefore, it is useful to accelerate rendering with full-resolution textures by using conventional texture atlases as long as the textures are small enough and non-repetitive. As a key advantage, LOD texture atlases facilitate the rendering of large city models that are completely covered with individual textures. Each triangle can be textured independently. This provides a technical basis to use surfaces of city model objects to visualize various kinds of thematic data that result from computational models. For this, these data are encoded into surface textures, i.e., textures that are draped onto the surfaces of city model objects.

Usually, when rendering a frame, there is no need for all the texture data to be at full resolution, only the texture data viewed from the current viewpoints are accessed by the renderer. Since the screen resolution is bounded, less detailed versions of textures can be used for distant geometry. In other words, displayed texture LOD does not need to exceed screen resolution. Automatic LOD is a powerful Unity extension that allows you to quickly generate and manage multiple levels of detail for your 3D models.

On the first level, the domain of the texture coordinates ranges from the center of the first element tot the center of the last one. It could happen that a voxel is aligned on different levels, which results in separation of the texture between two adjacent voxels. The width of texture elements on each level is calculated as we explained

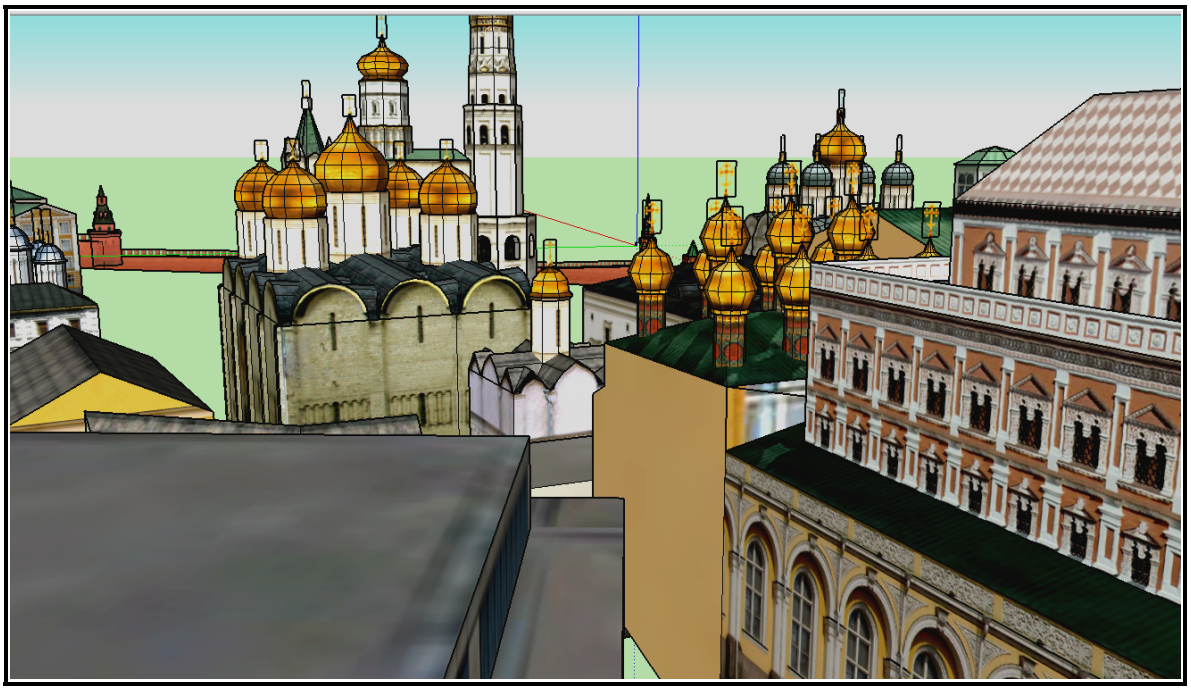


Figure 4.14: Original textures without LOD Rendering

before, and since the chunks have a constant shape, texture coordinates can be always determined. In order to account for a continuous levels of details transition, texture pieces boundaries need to be adapted whenever the hierarchy is changed by moving the focus point. As long as there are no level transitions greater than one are specified, our approach allows to use arbitrary texture resolution for every chunk. According to the OpenGL specification textures can be stored in different internal formats in the graphics subsystem. In general, the texture maps used for rendering can either consist of color values or color indices, which are then transformed to color values using texture lookup tables. The hierarchical decomposition does not imply any restrictions on the internal format of the textures. Down-sampling at textures boundaries can be performed in any case.

4.5 Discussion

Our work differs from the cited works above is that we develop a multi-resolution structure that allows smooth transition between levels. We also address the rendering artifacts that still persist when rendering two differing but adjacent levels. Our multi-resolution hierarchy is not strictly based upon an octree. Our work uses slicing planes which are parallel to the image plane rather than spherical shells for rendering efficiency.

The goal of our hierarchical representation is twofold: first, to convert the volume

data into a multi-resolution representation that entirely fits into the limited texture memory and second, to minimize texture lookups by adaptively resampling the data with respect to the selected level-of-detail. Each brick locally stores approximations of the original data at an ever coarser resolution. These copies are then used to adaptively render arbitrary regions with reduced detail size. Even if the multi-scale representation does not entirely fit into texture memory, texture loading will be reduced. In summary, by taking advantage of a level of detail representation as described, our goal is improved rendering performance. We initially decomposed the data into chunks. After constructing the texture hierarchy these bricks can be rendered at arbitrary resolutions. Some additional expenses is required to guarantee continuous transitions between adjacent textures pieces at different levels.

One dimensional example of data set with four levels of detail. On every level the size of texture elements increases by a factor of two. Note that at chunk boundaries on the same level texture elements have to be included in multiple chunks in order to guarantee continuous texture interpolation (filled areas). The original data set is represented by textures of level 0. Ascending level indices indicate coarser texture resolution. The width of texture elements on level k is twice the width of elements on level k-1, whereas the texture size is reduced by a factor of two. Texture elements on different resolution levels are aligned in such a way, that their centers on a certain level corresponds to an even element on the next finer level. Since the geometry or shape of chunk will be retained throughout the hierarchy, the domain of the underlying texture function, necessary to compute appropriate texture coordinates, has to be adapted accordingly.

Part III

Quantitative Results and Discussion

Chapter 5

Quantitative Results and Case Studies

Contents

Our overall approach to tackle the complexities of virtual 3D city models is to design and implement methods for generalization and texturing optimization of large urban data, originally in OSM. For this, we applied and combined general principles and fundamentals from 3D visualization systems. The main reason for this process is that raw data can not be directly used for rendering. It was transformed into graphics representation basically that are rendered by Cesium, a 3D rendering engine.

Building data were originally provided as an OSM file. The processing of this kind of data included three major stages and tasks:

- Data treatment: including geometry parsing, conversion, and creating 3D chunks to facilitate the analysis of the groups of buildings.
- Geometry Optimization: generalizing the structures of the buildings, while applying a certain LOD control to the distinguished buildings to preserve their general structure.
- Texture Optimization: includes texture atlasing and texture resolution control depending on animation frame in the scene.

5.1 Quantitative Evaluation

Studies were carried out to test the efficiency of our methods regarding the visualization format, the generalization, and the texturing in the context of scene animation. Frame rates calculation and performance evaluation are shown next.

5.2 Texturing in the Animation Context

The frame rates for the camera animation were measured in three parts:

- In the first part, the rendering was done with the original scene textures. The results are indicated by the thin solid line. The whole model of the city fitted in main memory so that the frame rate measurements could be done for comparison.
- In the second part, the scene was rendered without texture switches, so instead of several textures, a single white texture has been used for the complete scene. In this part, the model did not appear correctly. However, the measured frame rates provide an approximate upper bound for the rendering performance. The rendering performance could theoretically be achieved by reducing texture switches and texture workload. The result is shown by the dot line.
- In the third part, the scene was rendered using two texture atlas tree. Results indicated by the thick line.

The average frame rates were 17 fps for the original textures, 30 fps without texture switches and 27 fps for the atlas tree. At the overview perspective the original texture rendering dropped below 5 fps, while the atlas tree was permanently over 14 fps. Around frame 200, the second run surprisingly yielded the maximum performance. This effect can be explained as follows: when rendering with a single white texture or with LOD texture atlases, several small objects share a single texture atlas. Therefore, they can be combined to larger objects and rendered with a single rendering call. This, generally improves the rendering performance for most perspectives.

Around frame 200, however, the camera was in a perspective for which only a very small number of triangles and textures were inside the view frustum. In this special case, the rendering of separate small objects became advantageous because all objects outside the view frustum could be excluded from rendering.

Another test was made on the chunks created in Chapter3, which include the distinguished buildings. It consists of the buildings in the city center, with the following textures characteristics:

- For each building, a 512*512 resolution texture is repeated around the facade.
- The textures were given an ID to each. The size of the texture on the facade is 50 MB.

This test shows the rendering speed in fps[1] measured using a small animation consisting of 300 frames, in the context of a user navigation (in walk-through and fly-over modes). The average frame rate is 100 fps. A frame-rate minimum of 18 fps occurred at an overview perspective. Remarkably, particularly the overview perspective is uncritical from the point of view of textures because there is no building close to the camera. In this case, only a small number of texture atlases is required for rendering.

For example, when the frame rate minimum occurred, the number of used texture atlases was 34, while in the beginning of the camera path, where the frame rate was between 40 and 50 fps,the number of texture atlases was in the range of 60. This indicates that the bottleneck of switching between textures has been successfully overcome. Therefore, we could deduce that the frame rate drop is caused by other factors, such as the complexity of the model, or the fill rate, which means the number of number of triangles or the number of fragments that have to be processed by the graphics hardware.

These performance measurements change from one scene to another, depending on the amount of visible triangles per frame. This could lower the average frame rate in general. Tests made on the generalized models will drastically improve the frame rate, since the text-data contains less amount of redundantly high tessellated geometry, for example, rectangular walls that were represented by several triangles instead of just two. For the test we explained in this section, the redundancies had been removed from the model. Therefore, the frame rate minimum was higher in the new test (with 18 fps instead of 12 fps), although twice as many block buildings were rendered.

5.2.1 Texture Atlas Performance

The generation of the Texture Atlas in our city has made it possible to render the data set with drastic reduction of elements with respect to manual texture by texture implementation. Our rendering test was capped by 60 fps. A smooth rendering was performed in case of the atlas generation.

[1] Frames per Second

LOD Management Efficiency The efficiency of LOD management of the texture atlas has also been tested. Our test configuration started with the following parameterization:

- The resolution of the texture atlas is set to *512*512*, followed by the fact that each block should not exceed *130 KB*.
- The summed size of all blocks in memory was limited to 100 MB.
- The screen resolution of *800*600 pixels* with a *pixel-per texel ratio of 1.0* on an *HP ELITEBOOK Laptop with Intel(R) Core(TM) i5-4300U at 1.9 GHz, 4GB Main Memory, and Mobility Graphics Card with sufficient Graphics Memory.*

The test is only carried out for the scene parts rendered with LOD management. The roofs are not considered in the specification of texture complexity mentioned above nor in the following frame rate measurements.

We needed to specifically check the ability of the LOD texture atlases approach to cope with scenes in which the texture workload is irregularly distributed over the scene. Snapshots of the model are shown in The geometric complexity of texture data is about 36 million texture pixels, which could, theoretically, be handled directly by the graphics hardware. The model contains, however, about 2000 different textures, which are all visible at once from an overview perspective. Therefore, when rendering the scene in a conventional way is caused by the number of texture switches. In addition, the scene contains several repeated textures with high repetition counts such as the tile texture on the ground in the lower part of This makes it difficult to reduce the number of textures switches by applying conventional texture atlases

Repeated Textures Repeated textures must also be considered when using LOD texture atlas. Textures repetition is applied for our city model. In contrast to conventional texture atlases, LOD texture atlases allow using texture repetition without problems even for large repetition counts because the texture resolution does only depend on the screen space area covered by each triangle at a distance *d¿dmin*. That is, it is not relevant for the required texture space whether a triangle contains 100 repetitions of a tile texture or a single one. It is because the repeated tiles appear correspondingly smaller.

This made it easier to apply texture repetition to our model. In contrast, using conventional texture atlases for the model, texture repetition would have increased the total amount of texture data to multiple billion texture pixels, leading to an unacceptably large number of necessary texture atlases.

5.3 Saliency Metrics

We are going to evaluate in this section the proposal of Chapter4 by creating the saliency maps of the output buildings, for different saliency parameters. The focus of the important buildings brings out aspects related to *Saliency*[2]. It is used to differentiate between information of *Hight Interest* and those of *Low Interest*. It could be applied to images, as well as to 3D models. To identify areas of high details, we are going to compute the interest value for each visible feature using saliency metrics. These metrics are: *The view Distance, the Vertical View Angle, and the Region of Interest*. Other metrics can be added as long as they are normalized. For example, view metrics can be defined by *Normalized Euclidean Distances and Angles.The Region of Interest* is represented by a distance map that is computed using the *Jump-Flooding Algorithm* [RT06], and is used to visualize ROIs[3] or routes through a virtual 3D city model [TBPD11, STKD12b].

Other metrics can be added as long as they are normalized. For instance, view metrics can be defined by *Normalized Euclidean Distances* and angles as we showed in figure:

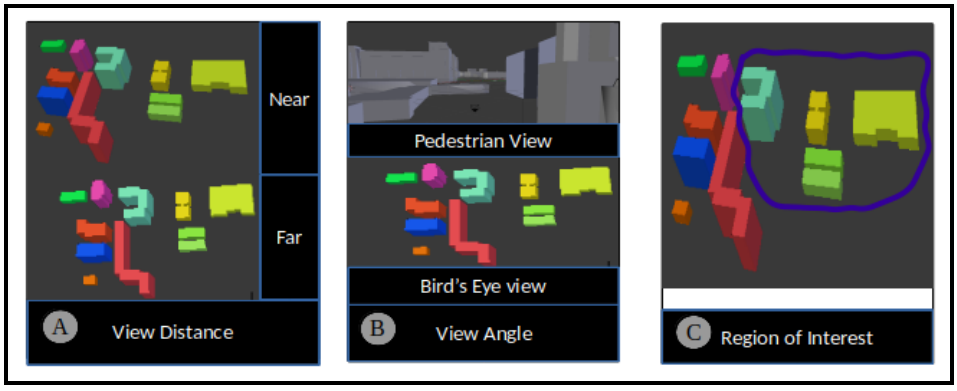


Figure 5.1: The different saliency metrics parameters

.

As can be seen, visual saliency of homogeneous graphic styles is distributed across focus and context regions. By contrast, the visualization method with texturing presented in Chapter4 yields to concentrated high saliency within a circular region of interest, and for single landmarks in the context area.

[2]The quality of being particularly noticeable, important, or prominence
[3]Regions of Interest

5.4 Discussion and Limitations

During development and authoring of the use cases, it was noticed that the parametrization of the system can be heavy. Based on this, we need to configure a way to configure the transitions between one use case to the other. A parametrization of the graphic styles could be also helpful in order to deploy usage scenarios.

Another observation concerns testing the texturing method on the generalized buildings too. Although this part was not developed due to time limits, we could have proposed a way that allows the modification of the textures we have in order to adapt the change of vertices and geometries when the generalized method is in action. The way we proceeded with applying the textures is applicable on the non generalized buildings. It could be interesting as a future work to test its efficiency on the simplified model of our city, with numbers showing the time consumption between both cases: Simplified Vs Non Simplified.

Moreover, the performance evaluation indicates that a single-threaded rendering engine has impact on the system's interactivity.

This work presents a concept and an implementation of a system that visualizes a 3D city model in an optimized manner. This includes generalization of the distinguished buildings, applying the textures by using the concept of LOD texture atlas depending on the building distance from the camera. The texturing work is evaluated by using saliency metrics, as explained before. We also established use cases and evaluate the performance in case of animation. The system is extensible and could be used to improve existing visualization techniques, such as focus+context zooming [QWC+09] and deformation. Moreover, the generalization technique proposed gives a potential to our system to be applied on mobile devices, for the simplification of the buildings reduces information compression on displays with limited size.

Chapter 6

Conclusion and Perspectives

Contents

This chapter discusses the methodologies explained in the previous chapters with the advantages and disadvantages, answers the thesis objectives, and offers proposals for future improvement.

6.1 Technologies, Tools, and Methods Overview

Our work includes various techniques and tools that allow the optimization of the data visualization, without the loss of important information of the city. These techniques and tools have their advantages and disadvantages, which we are going to elaborate. First, we consider our proposal in this thesis as an effective optimized rendering solution for streaming 3D heavy urban data in web environments. This is done under the premise of using the open source software Cesium, international standard, and royalty-free transmission formats (in Chapter3).

6.2 Using glTF for Web Rendering

Modern web browsers have powerful built-in functionalities such as WebGL, which can be leveraged in order to create online applications including 3D GIS and Virtual

Globes. WebGL is supported by all modern browsers at least since 2013 and is currently further developed to support OpenGL. Hence, web browsers provide a reliable platform for publishing and viewing 3D city models. However, the overall performance of WebGL still lags with technical limitations.

The data processing by the client and 3D rendering must be streamlined as much as possible in order to avoid rendering problems. The advantage of using WebGL implemented in Cesium is that all necessary software components are already installed on the client, and the integration with other web content and layout is straight-forward. In this context, using glTF as a rendering format in Cesium 3D.js has the following advantages:

- Meshes can be merged effectively across multiple geometries without losing the possibility to identify single features in order to increase rendering performance.
- Cesium.js allows interacting with single features easily such as selecting, highlighting, and extracting features properties.
- Vertex data is stored as binary data, which is more efficient than text representations both in terms of file size and parsing speed. Compressing files for network transmission is not necessary.

6.3 Answers for Thesis Objectives

In this section, we present the attempts to answer the research of the introduction in the same order: Question1: How accurate is the choice of the openstreetmap standard as a starting point of the city raw data?
Answer: Choosing the osm format to work with as a starting point invoked questions about the accuracy of this choice. Of course dealing with large-scale open source database has its constraints. In the case of osm the data quality could be better, the coverage could be incomplete, and we could find inconsistencies from place to place, as stated by Hakla [Hak10]. However, these issues are gradually decreasing and the advantages offered by OpenStreetMap outweights these issues for many applications done. Yet all of these issues are gradually being out. The advantages of Open Street Map outstrips these issues for many applications already. These advantages include:

- **R**apid updates of new projects
- **G**reater range of attributes
- **A**bility to publicly share data without the need of specific license

3D geo-databases are needed to handle surface and volume models and new directions for the development of 3D geo-databases to open new fields for interdisciplinary research are addressed In this domain, it is important to look at the nature of the modeled objects, which differs from one application to another. The task of geo-database researchers is to find implementations for geo-models that provide efficient storage adn retrieval of the models in geo-databases. Transformation is also necessary to convert a geo-model used for visualization into a geo-model stored in the geo-database.

Consistent Reduced Information: Optimized representation of the city needs to have reduced information, while maintaining sufficient both the general shapes of the buildings, and a focus on the important buildings throughout the animation.

Automatic Reduction: The optimization technique, especially the generalization should have some degree of automation, especially ragarding the buildings which are more or less similar.

Saliency Control: Saliency metrics should be used to keep the viewpoint with respect to the distinguished buildings. That is why a method for distinguishing these buildings should be applied.

The second research question occurs in the context of texturing rendering: How can we render the different *Texture Atlases* with the different resolutions? This question involves the following requirement:

Multi-Resolution Texturing: It is crucial to make the textures adapt the camera viewpoint. The textures display should take into consideration the distance of the buildings from the camera, where the textures of the far buildings should have a lower resolution than the ones of the close buildings.

There are, however, some disadvantages of using glTF:

- The internal structure of the features such as defined in CityGML could not be represented so far, for example the subdivision into thematic surfaces can not be represented by B3DM batches because it is not possible to store hierarchy information. TO However, experiments are done showing that with workarounds it is possible to extend the concept of the batch table as specified by B3DM so that model hierarchies can be included, such as in the work of [SBN16].

- LODs are stored statically in glTF. Mixed LOD scenes using HLOD (or Hierarchical LODs) CityGML filtering can be stored in a single B3DM file. Switching between different building representations according to the distance to the viewer is, nevertheless, not possible.

Question2: Is it possible to apply some modifications for better texturing outcomes? Answer: the answer opens the door for many titles:

Model Simplification could be one of the answers. It could surely be used in combination with the texture atlas, so instead of using the original geometry for all tree levels, simplified versions of the scene geometry could be stored in the inner node of the texture atlas tree. And as we said before, our simplification proposal works best for smooth continuous surfaces, and simplifying the very detailed city models will consist of large numbers of small disjoint geometric objects, so that drastic simplification will lead to distortions in the visual representation. Reducing the original geometry will certainly reduce the geometric workload, and therefore what is handled by the graphic hardware will only be the texture complexity instead of having both the textures and the geometric complexity.

Reducing Texture Atlas File Size The file that stores the original textures of a texture atlas tree, sometimes reaches gigabytes. A possible way to reduce its size is to reduce the amount of unused textures, by arranging the textures in a more efficient way. Concerning the optimization of the atlas itself, there are not lots of things that could be done, since it is already an optimized version of the original textures. It is also considered a considerable fraction of the original textures is when the scene contains many repeated textures. It is due to the fact that usually, repeated textures in a texture atlas are handled by a fragment shader, introduced by [Wlo05]. Integrating such shader in the rendering process reduces the file size and preprocessing time at the cost of additional computational effort involved by the fragment shader and the restriction to recent graphics hardware.

Also, Improvements need to be done as for having a standard relational data model in database systems, although there are remarkable attempts to standardize the geo-data, for example by the OpenGeoSpatial Consortium (OGC), and exchanging data in an unified way. Anyhow, Geo-databases will also play an important role in bridging the geometric modeling of man-made and natural geo-objects. Therefore, it is useful to provide geometric primitives for both man-made and natural objects eo construct more complex objects, with the database serving as a tool to access the geometric features of the objects. Important directions include the integration of 2D and 3D data models and the development of dimension-independent topological and geometric data models. These data models should be uniformly usable for both 2D and 3D worlds. If these spatial data integration challenges are not solved, we expect the development of many separate 2D and 3D solutions in the future. Such solutions would avoid seamless data integration. Thus, challenges for future research include spatial data integration, new user interfaces for geo-databases, and development of 3D/4D geo-information systems, and geo-sensor databases.

textit Question3: To what extent is a city database creation important in large

data and how could it facilitate the web management?
*A*nswer: In case the 3D city models have to be continuously managed, a database creation is a must. Many databases have been created for storing, managing, and maintaining the information about the city. Some GIS databases exist and are always researched to produce city models semi or fully automatically.

6.4 Conclusion

The huge amount of data, rapid development, and the increasing data supplements from map agencies has led us to experiment the fusion of different datasets and formats. We considered the example of the city of Moscow, and for better performance purposes, we focused on procedurally texturing the distinguished buildings. The structure of the application consisted in communicating between the server, the editor, and the webpage. Many visualization engines require the presence of topological primitives (such as nodes and edges) to support the display of 3D objects, with the data primarily stored in proprietary file formats for performance purposes. It is therefore possible to provide support for visualization in 3D as part of the implementation of topological functionality, by modifying and enhancing the primitives model to support visualization. In addition to this, the use of standard databases such as currently utilized to hold GIS data, should also be considered to open the topologically structured data to both query and visualization. This may provide one possible solution to the problem of passing large data volumes to a visualization engine.

As a consequence of data volume issues, many 3D visualization applications use varying levels of detail to describe objects at different scales. This process reduces the quantity of polygon mesh data passed to the visualization engine as the user moves away from a particular object. All the processing steps explained in 3, 4, and 5 which consider chunking, simplifying, and procedural texturing have been implemented and integrated in a workflow, which uses the 3dcitydb and enables preparing raw data city models for being efficiently published and viewed in a web browser. Using open standards not only for storing and exchanging 3D city models but also for web streaming will also improve interoperability and open up new possibilities for including external data models and thus creating a mashup of standards and rules.

6.4.1 Perspectives

Regarding the completeness and accuracy about city datasets, we think that city layers formats in general and CityGML in particular could get more advanced, especially in terms of LODs. Tags used in the file content could integrate information like roof

types, roofs orientation, and eaves heights. For this goal, a database scheme should be setup. Setting up a database for large datasets has big advantages:

- The data could be accessed by a variety of third party programs.
- Spatial queries could be done by clients accessing the database, which widens the functionality of the database.
- Huge datasets could be stored in a database: the data is only transfered to RAM when only a specific query is called.

The explained techniques and concepts used in our project have been successfully applied to dynamically visualize the city data in specific use cases.

Nevertheless, multiple aspects are to be discussed:

Maintenance of multiple representations to be used as Levels- Of-Details is critical aspect of large three-dimensional models (Thomsen et al., 2008). The consistent management of multiple representations is still far from being formalised. Currently, there are no formal rules for transition from one 3D LOD to another.

Development of such rules might be a breakthrough in 3D generalisation techniques. Such rules will permit only the most detailed model to be persistently stored because any other LOD could be derived on demand. Although not critical for most natural objects, management of texture and mechanisms for texture mapping and texture draping are critical for management of realistic 3D topographical object models. Man-made 3D features are often textured with images from the real world. As 3D models continue to develop, textures will be required for maintenance. Textures can be understood as 'presentation' attributes of 3D objects.

Appropriate indexing of texture images is required. To model more real-world phenomena, 3D models must be extended with new representations such as parametric shapes, free-form curves and surfaces, as discussed in (Breunig and Zlatanova, 2006). The 3D spatial functionality regarding such complex 3D spatial data types must be further defined. The role of databases remains an open question: can they be seen primarily as repositories of data or as systems that provide certain functionality? In our opinion, functionality should be developed (or extended) according to the extent of objects to be maintained in the database. As soon as 3D geometry (with extended primitives) and topological models are supported, a database should provide functions and operations for validity, generic analysis and seamless conversion between models. Import and export should be also synchronized with 3D geo-data standards.

Bibliography

[And05] Karl-Heinrich Anders. Level of detail generation of 3d building groups by aggregation and typification. In *International Cartographic Conference*, volume 2, 2005. 32, 35

[BAR13] Siddique Ullah Baig and Alias Abdul-Rahman. Generalization of buildings within the framework of citygml. *Geo-spatial Information Science*, 16(4):247–255, 2013. 25

[Bat07] Michael Batty. Model cities. *Town Planning Review*, 78(2):125–151, 2007. 18

[BBD05] Henrik Buchholz, Johannes Bohnet, and Jurgen Dollner. Smart and physically-based navigation in 3d geovirtual environments. In *Information Visualisation, 2005. Proceedings. Ninth International Conference on*, pages 629–635. IEEE, 2005. 66

[BD05] Henrik Buchholz and Jurgen Dollner. View-dependent rendering of multiresolution texture-atlases. In *Visualization, 2005. VIS 05. IEEE*, pages 215–222. IEEE, 2005. 63

[Bec03] Michael Beck. Real-time visualization of big 3d city models. *International Archives of the Photogrammetry Sensing and Spatial Information Sciences*, 34, 2003. 38

[BJF95] L. Bundy, B. Jones, and E. Furse. Holistic generalisation of large-scale cartographic data. In *GIS and Generalisation, Gisdata1, Taylor*, pages 106–119. Muller, J.C., Lagrange, J.P., and Weibel, 1995. 35

[BJPS13] Sergiy Byelozyorov, Rainer Jochem, Vincent Pegoraro, and Philipp Slusallek. From real cities to virtual worlds using an open modular architecture. *The Visual Computer*, 29(2):141–153, 2013. 40

[BWPP04] Jiří Bittner, Michael Wimmer, Harald Piringer, and Werner Purgathofer. Coherent hierarchical culling: Hardware occlusion queries made useful. In *Computer Graphics Forum*, volume 23, pages 615–624. Wiley Online Library, 2004. 64

[CCM00] C Chen, M Czerwinski, and R Macredie. *Individual Differences in Virtual Environments- Introduction and Overview*, volume 51(6). Wiley InterScience, 2000. 65

[CE98] David Cline and Parris K Egbert. Interactive display of very large textures. In *Visualization'98. Proceedings*, pages 343–350. IEEE, 1998. 62

[CG02] Alesandro Cecconi and Martin Galanda. Adaptive zooming in web cartography. *Computer Graphics Forum*, 21(4):787–799, 2002. 42

[CGJT80] Edward G Coffman, Jr, Michael R Garey, David S Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980. 67

[CH02] Nathan A Carr and John C Hart. Meshed atlases for real-time procedural solid texturing. *ACM Transactions on Graphics (TOG)*, 21(2):106–131, 2002. 62

[Coo01] V. Coors. Feature-preserving simplification in web-based 3d-gis. *Proceedings on International Symposium on Smart Graphics*, page 5, 2001. 30

[DB98] Victor J Deleon and H Robert Berry. Virtual florida everglades. In *4 th International Conference on Virtual System and Multimedia VSSM98–FutureFusion Application Realities for Virtual Age, 1998–Japan. Pg. 458*. Citeseer, 1998. 19

[DBB06] J. DÃ–LLNER, K. BAUMANN, and H. BUCHHOLZ. Virtual 3d city models as foundation of complex urban information spaces. *Proceedings of CORP*, 2006. 38

[DBVKOS00] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000. 56

[DDA04] R Dogan, S Dogan, and MO Altan. 3d visualization and query tool for 3d city models. In *Proceeding of XXth ISPRS Congress*, 2004. 19

[DP11] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Classics in Cartography: Reflections on Influential Articles from Cartographica*, pages 15–28, 2011. 34

[EMBI01] Carl Erikson, Dinesh Manocha, and William V Baxter III. Hlods for faster display of large static and dynamic environments. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 111–120. ACM, 2001. 63

[EP08] Birgit Elias and Volker Paelke. User-centered design of landmark visualizations. *Map-based mobile services*, pages 33–56, 2008. 41

[FLM09] H. Fan, Meng L., and Jahnke M. Generalisation of 3d buildings modeled by citygml. In *Lecture Notes in Geoinformation and Cartography, Advances in GIScience, Paelke V., Sester M.,and Bernard L.*, pages 387–405. Springer, Berlin, Heidelberg, 2009. 35, 36

[FM02] Andrea Forberg and Helmut Mayer. Generalization of 3d building data based on scale-spaces. *INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY REMOTE SENSING AND SPATIAL INFORMATION SCIENCES*, 34(4):225–230, 2002. 28

[FM12] H. Fan and L. Meng. A three-step approach of simplifying 3d buildings modeled by citygml. *International Journal of Geographical Information Science*, 26(6):1091–1107, 2012. 36

[FOJ04] Martin Fornefeld, Peter Oefinger, and Kathrin Jaenicke. Nutzen von geodateninfrastrukturen. *Studie, MICUS Management Consulting GmbH*, 2004. 50

[For07] Andrea Forberg. Generalization of 3d building data based on a scale-space approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(2):104–111, 2007. 28

[FS93] Thomas A Funkhouser and Carlo H Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 247–254. ACM, 1993. 63

[FSZ04] Christian Frueh, Russell Sammon, and Avideh Zakhor. Automated texture mapping of 3d city models with oblique aerial imagery. In *3D Data*

Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on, pages 396–403. IEEE, 2004. 62, 64

[GB09] Richard Guercke and Claus Brenner. A framework for the generalization of 3d city models. In *Proceedings of 12th AGILE Conference on GIScience*, 2009. 31

[GBS09] R. Guercke, C. Brenner, and M. Sester. Generalization of semantically of semantically enriched 3d city models. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII(3):28–34, 2009. 31, 34

[GD09] Tassilo Glander and Jürgen Döllner. Abstract representations for interactive visualization of virtual 3d city models. *Computers, Environment and Urban Systems*, 33(5):375–387, 2009. 29

[GH97] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997. 42

[GTD11] Tassilo Glander, Matthias Trapp, and Jürgen Döllner. Concepts for automatic generalization of virtual 3d landscape models. *Digital Landscape Architecture Proceedings*, pages 127–135, 2011. 43

[GVB+15] Jérémy Gaillard, Alexandre Vienne, Rémi Baume, Frédéric Pedrinis, Adrien Peytavie, and Gilles Gesquière. Urban data visualisation in a web browser. In *Proceedings of the 20th International Conference on 3D Web Technology*, pages 81–88. ACM, 2015. 40

[GZBZ12] Richard Guerckea, Junqiao Zhaob, Claus Brennera, and Qing Zhu. Generalization of tiled models with curved surfaces using typification. *Advances in Geo-Spatial Information Science*, page 33, 2012. 33

[Hae14] KH Haefele. Citygml model of the fjk-haus. In *3D Geoinformation Science*. Springer, 2014. 20

[Hak10] Mordechai Haklay. How good is volunteered geographical information? a comparative study of openstreetmap and ordnance survey datasets. *Environment and planning B: Planning and design*, 37(4):682–703, 2010. 90

[Har99] Lars E Harrie. The constraint method for solving spatial conflicts in cartographic generalization. *Cartography and geographic information science*, 26(1):55–69, 1999. 32

[HGM94] Günter Hake, Dietmar Grünreich, and Liqiu Meng. *Kartographie: Visualisierung raum-zeitlicher Informationen*. Walter de Gruyter, 1994. 41

[HMM12] Shuang He, Guillaume Moreau, and Jean-Yves Martin. Footprint-based 3d generalization of building groups for virtual city visualization. In *The Fourth International Conference on Advanced Geographic Information Systems, Applications, and Services, Valencia, Spain [online] Available from: http://www. thinkmind. org/index. php*, 2012. 25

[HMT04] Gerd Hesina, Stefan Maierhofer, and Robert F Tobler. Texture management for high-quality city walk-throughs. *Proceedings of the CORP–GeoMultimedia*, pages 305–308, 2004. 63

[Hop96] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM, 1996. 42

[Hop98] Hugues Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Visualization'98. Proceedings*, pages 35–42. IEEE, 1998. 42

[HS00] Ralf Helbing and Thomas Strothotte. Quick camera path planning for interactive 3d environments. *Smart Graphics 2000 Demo Session*, 2000. 66

[HWT+05] Andy Hamilton, Hongxia Wang, Ali Murat Tanyer, Yusuf Arayici, Xiaonan Zhang, and YH Song. Urban information model for city planning. *Journal of Information Technology in Construction (ITCon)*, 10:55–67, 2005. 21

[IKMT98] Takeo Igarashi, Rieko Kadobayashi, Kenji Mase, and Hidehiko Tanaka. Path drawing for 3d walkthrough. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 173–174. ACM, 1998. 66

[Kad02] Martin Kada. Automatic generalization of 3d building models. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIV-4, 2002. 26

[Kad06] M. Kada. 3d building generalization based on half-space modeling. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(2):58–64, 2006. 26

[Kad07] M. Kada. Generalisation of 3d building models by cell decomposition and primitive instancing. In *Proceedings of the Joint ISPRS Workshop on Visualization and Exploration of Geospatial Data*, Stuttgart,Germany, 2007. 28

[Kad09] Martin Kada. The 3d berlin project. In *Photogrammetric week*, pages 331–340, Wichmann Verlag, Heidelberg, 2009. 28

[KG15] Michel Krämer and Ralf Gutbell. A case study on 3d geospatial applications in the web using state-of-the-art webgl frameworks. In *Proceedings of the 20th International Conference on 3D Web Technology*, pages 189–197. ACM, 2015. 38

[KNH13] Thomas H Kolbe, Claus Nagel, and Javier Herreruela. 3d city database for citygml. *Addendum to the 3D City Database Documentation Version*, 2(1), 2013. 39

[Kol09] Thomas H Kolbe. Representing and exchanging 3d city models with citygml. In *3D geo-information sciences*, pages 15–31. Springer, 2009. 21

[Lak04] Ali Lakhia. Efficient interactive rendering of detailed models with hierarchical levels of detail. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 275–282. IEEE, 2004. 63

[LDN04] Sylvain Lefebvre, Jérome Darbon, and Fabrice Neyret. *Unified texture management for arbitrary meshes*. PhD thesis, INRIA, 2004. 62

[Lin00] Peter Lindstrom. Out-of-core simplification of large polygonal models. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 259–262. ACM Press/Addison-Wesley Publishing Co., 2000. 33

[LKH+95] Peter Lindstrom, David Koller, Larry F Hodges, William Ribarsky, Nick L Faust, and Gregory Turner. Level-of-detail management for real-time rendering of phototextured terrain. Technical report, Georgia Institute of Technology, 1995. 62

[LKPM01] YC Lee, Angela Kwong, Lilian Pun, and Andy Mack. Multi-media map for visual navigation. *Journal of Geospatial Engineering*, 3(2):87–96, 2001. 41

[LKR+96] Peter Lindstrom, David Koller, William Ribarsky, Larry F Hodges, Nick Faust, and Gregory A Turner. Real-time, continuous level of detail rendering of height fields. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 109–118. ACM, 1996. 42

[LLL17] Po Liu, Chengming Li, and Fei Li. Texture-cognition-based 3d building model generalization. *ISPRS International Journal of Geo-Information*, 6(9):260, 2017. 74

[LMM02] Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European journal of operational research*, 141(2):241–252, 2002. 67

[Lyn60] Kevin Lynch. *The image of the city*, volume 11. MIT Press, 1960. 29

[MA76] E.M. Mikhail and F.E. Ackermann. *Observations and Least Squares*. New York: IEP, 1976. 109

[Mao11a] Bo Mao. *Visualisation and generalisation of 3D City Models*. PhD thesis, KTH Royal Institute of Technology, 2011. 38

[Mao11b] Bo Mao. *Visualisation and generalisation of 3D City Models*. PhD thesis, KTH Royal Institute of Technology, 2011. 54

[May05] H. Mayer. Scale-spaces for generalization of 3d buildings. *International Journal of Geographical Information Science*, 19(8-9):975–997, 2005. 36

[MF07] L. Meng and A. Forberg. 3d building generalisation. *Challenges in the portrayal of geographic information. Amsterdam: Elsevier Science*, 2007. 25, 36

[MK01] Alan M MacEachren and Menno-Jan Kraak. Research challenges in geovisualization. *Cartography and geographic information science*, 28(1):3–12, 2001. 19

[PD16] I Pispidikis and E Dimopoulou. Development of a 3d webgis system for retrieving and visualizing citygml data based on their geometric and semantic characteristics by using free and open source technology. *ISPRS*

Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, pages 47–53, 2016. 39

[PDS+15] F Prandi, F Devigili, M Soave, U Di Staso, and R De Amicis. 3d web visualization of huge citygml models. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(3):601, 2015. 38

[PHF07] Michael Peter, Norbert Haala, and Dieter Fritsch. Preserving ground plan and facade lines for 3d building generalization. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XXXVII-B2, 2007. 26

[QWC+09] Huamin Qu, Haomian Wang, Weiwei Cui, Yingcai Wu, and Ming-Yuen Chan. Focus+ context route zooming and information overlay in 3d urban environments. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1547–1554, 2009. 88

[RB06] Nora Ripperda and Claus Brenner. Reconstruction of façade structures using a formal grammar and rjmcmc. In *Joint Pattern Recognition Symposium*, pages 750–759. Springer, 2006. 27

[RC14] Fabrice Robinet and P Cozzi. Gltf—the runtime asset format for webgl, opengl es, and opengl, 2014. 50

[RCT+06] Jiann-Yeou Rau, Liang-Chien Chen, Fuan Tsai, Kuo-Hsin Hsiao, and Wei-Chen Hsu. Lod generation for 3d polyhedral building model. In *Pacific-Rim Symposium on Image and Video Technology*, pages 44–53. Springer, 2006. 26

[Reg01] N Regnauld. Contextual building typification in automated map generalization. *Algorithmica*, 30-no2:312–333, 2001. 32

[RFC13] José IJ Rodrigues, Mauro JG Figueiredo, and Celso P Costa. Web3dgis for city models with citygml and x3d. In *Information Visualisation (IV), 2013 17th International Conference*, pages 384–388. IEEE, 2013. 38

[RHG+01] José Ribelles, Paul S Heckbert, Michael Garland, Tom Stahovich, and Vinit Srivastava. Finding and removing features from polyhedra. In *Proceedings of Design Engineering Technical Conference (DETC)*, pages 1–10. Pittsburgh, PA, USA, 2001. American Association of Mechanical Engineers (ASME), 2001. 26

[RHSS98] Stefan Röttger, Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. Real-time generation of continuous levels of detail for height fields. In *International Conference in Central Europe on Computer Graphics and Visualization*, pages 315–322. Václav Skala-UNION Agency, 1998. 42

[RSV01] Philippe Rigaux, Michel Scholl, and Agnes Voisard. *Spatial databases: with application to GIS*. Morgan Kaufmann, 2001. 20

[RT06] Guodong Rong and Tiow-Seng Tan. Jump flooding in gpu with applications to voronoi diagram and distance transform. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 109–116. ACM, 2006. 87

[Rua98] Anne Ruas. A method for building displacement in automated map generalisation. *International Journal of Geographical Information Science*, 12(8):789–803, 1998. 41

[SB05] Monika Sester and Claus Brenner. Continuous generalization for visualization on small mobile devices. In *Developments in spatial data handling*, pages 355–368. Springer, Berlin Heidelberg, 2005. 34, 36

[SBN16] Arne Schilling, Jannes Bolling, and Claus Nagel. Using gltf for streaming citygml 3d city models. In *Proceedings of the 21st International Conference on Web3D Technology*, pages 109–116. ACM, 2016. 91

[Sch95] G. Schaufler. Dynamically generated impostors. In *Proceedings of the Workshop Modeling-Virtual Worlds-Distributed Graphics(MVD'95*, pages 129–136, 1995. 64

[Ses00] Monika Sester. Generalization based on least squares adjustment. *International archives of photogrammetry and remote sensing*, 33:931–938, 2000. 34

[Ses02] Monika Sester. Application dependent generalization-the case of pedestrian navigation. *INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY REMOTE SENSING AND SPATIAL INFORMATION SCIENCES*, 34(4):291–296, 2002. 32

[Ses05] Monika Sester. Optimization approaches for generalization and data abstraction. *International Journal of Geographical Information Science*, 19(8-9):871–897, 2005. 41

[SF99] Anthony Steed and Emmanuel Frécon. Building and supporting a large-scale collaborative virtual environment. *Proceedings of 6th UKVRSIG, University of Salford*, pages 59–69, 1999. 39

[SGLM03] Brian Salomon, Maxim Garber, Ming C Lin, and Dinesh Manocha. Interactive navigation in complex environments using path planning. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 41–50. ACM, 2003. 66

[SM13] Heidrun Schumann and Wolfgang Müller. *Visualisierung: Grundlagen und allgemeine Methoden*. Springer-Verlag, 2013. 37

[STKD12a] Amir Semmo, Matthias Trapp, Jan Eric Kyprianidis, and Jürgen Döllner. Interactive visualization of generalized virtual 3d city models using level-of-abstraction transitions. In *Computer Graphics Forum*, volume 31, pages 885–894. Wiley Online Library, 2012. 42

[STKD12b] Amir Semmo, Matthias Trapp, Jan Eric Kyprianidis, and Jürgen Döllner. Interactive visualization of generalized virtual 3d city models using level-of-abstraction transitions. In *Computer Graphics Forum*, volume 31, pages 885–894. Wiley Online Library, 2012. 87

[TBPD11] Matthias Trapp, Christian Beesk, Sebastian Pasewaldt, and Jürgen Döllner. Interactive rendering techniques for highlighting in 3d geovirtual environments. In *Advances in 3D Geo-Information Sciences*, pages 197–210. Springer, 2011. 87

[Thi02] Frank Thiemann. Generalization of 3d building data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(4):286–290, 2002. 26

[TKRL07] Jacques Teller, Abdel Kader Keita, Catherine Roussey, and Robert Laurini. Urban ontologies for an improved communication in urban civil engineering projects. presentation of the cost urban civil engineering action c21 towntology. *Cybergeo: European Journal of Geography*, 2007. 21

[TLC12] Fuan Tsai, Wan-Rong Lin, and Liang-Chien Chen. Multi-resolution representation of digital terrain and building models. *Advances in geospatial information science*, pages 233–242, 2012. 32

[TMJ98] Christopher C Tanner, Christopher J Migdal, and Michael T Jones. The clipmap: a virtual mipmap. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 151–158. ACM, 1998. 62

[TS05] Frank Thiemann and Monika Sester. Interpretation of building parts from boundary representation. In *Proceedings of the 1st International Workshop on Next Generation 3D City Models,Bonn*. Whittles Publishing, 2005. 31

[TS06] Frank Thiemann and Monika Sester. 3d-symbolization using adaptive templates. *Proceedings of the GICON*, 2006. 27

[Vin99] Norman G Vinson. Design guidelines for landmarks to support navigation in virtual environments. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 278–285. ACM, 1999. 41

[VK01a] Marc Van Kreveld. Smooth generalization for continuous zooming. In *Proceedings of the ICA, 4th Workshop on Progress in Automated Map Generalization, Peking, China*, 2001. 32

[VK01b] Marc Van Kreveld. Smooth generalization for continuous zooming. In *Proc. 20th International Cartographic Conference (ICC'01)*, pages 2180–2185, 2001. 42

[WA09] Markus Wolff and Hartmut Asche. Towards geovisual analysis of crime scenes–a 3d crime mapping approach. In *Advances in GIScience*, pages 429–448. Springer, 2009. 18

[War12] Colin Ware. *Information visualization: perception for design*. Elsevier, 2012. 43, 65

[Wei96] Robert Weibel. A typology of constraints to line simplification. In *Advances in GIS Research II. 7th International Symposium on Spatial Data Handling, UK*, pages 533–546, 1996. 32

[Wil83] Lance Williams. Pyramidal parametrics. In *Acm siggraph computer graphics*, volume 17, pages 1–11. ACM, 1983. 76

[Wlo05] Matthias Wloka. Improved batching via texture atlases. *Shader X3: Advanced Rendering with DirectX and OpenGL*, pages 155–167, 2005. 92

[WTES08] Stephan Winter, Martin Tomko, Birgit Elias, and Monika Sester. Landmark hierarchies in context. *Environment and Planning B: Planning and Design*, 35(3):381–398, 2008. 56

[ZZD+12] J. Zhao, Q. Zhu, Z. Du, T. Feng, and Y. Zhang. Mathematical morphology-based generalization of complex 3d building models incorporating semantic relationships. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:95–111, 2012. 36

Appendix A

Index of terms

k-d Trees: A k-d tree is created when each tile has two children separated by a splitting plane parallel to the x, y, or z axis (or longitude, latitude, height). Usually, a k-d tree does not have uniform subdivision like typical 2D geospatial tiling schemes and, therefore, can create a more balanced tree for sparse and non-uniformly distributed datasets.

R-Trees: R-Trees are data structures used for spatial access methods. The key is to group nearby objects and represent them with their minimum bounding rectangle in the next higher level of the tree. At the leaf level, each rectangle describes a single object. At a higher level, it describes the aggregation of an increasing number of objects. R-Trees do not guarantee good worst-case performance, but generally perform well with real-world data. The spatial structure of the R-Tree is more suitable for non homogeneous distributed data than regular data.

Quadtrees: A quadtree is created when each tile has four uniformly subdivided children(e.g., using the center longitude and latitude) similar to typical 2D geospatial tiling schemes. Empty child tiles can be omitted. Traditional quadtree subdivision, used in TMS for example, is sufficient for map tiles and 2D, but it is suboptimal for 3D and non-uniform dataset distributions. 3D tiles take advantage of empty tiles: those tiles that have a bounding volume, but no content. Since a tile's content property does not need to be defined, empty non-leaf tiles can be used to accelerate non-uniform grids with hierarchical culling. This essentially creates a quadtree or octree without hierarchical levels of detail (HLOD).

MipMapping: In computer graphics, mipmaps (also called MIP maps or pyramids), are a pre-calculated, optimized sequences of images, each of which is a progressively lower resolution representation of the same image. The height and width of

each image, or level, in the mipmap is a power of two smaller than the previous level. Mipmaps do not have to be square. They are intended to increase rendering speed and reduce aliasing artifacts. A high-resolution mipmap image is used for high-density samples, such as for objects close to the camera. Lower-resolution images are used as the object appears farther away. This is a more efficient way of downfiltering (minifying) a texture than sampling all texels in the original texture that would contribute to a screen pixel. Mipmap textures are used in 3D scenes to decrease the time required to render a scene. They also improve the scene's realism, at the cost of 1/3 more memory per texture.

Back-Face Culling: In computer graphics, back-face culling determines whether a polygon of a graphical object is visible. It is a step in the graphical pipeline that tests whether the points in the polygon appear in clockwise or counter-clockwise order when projected onto the screen. If the user has specified that front-facing polygons have a clockwise winding, but the polygon projected on the screen has a counter-clockwise winding then it has been rotated to face away from the camera and will not be drawn. The process makes rendering objects quicker and more efficient by reducing the number of polygons for the program to draw. For example, in a city street scene, there is generally no need to draw the polygons on the sides of the buildings facing away from the camera; they are completely occluded by the sides facing the camera.

Bulk Loading: Bulk loading is a way to load data (typically into a database) in large chunks, where the user enters information once at a time into the system. It takes a file of this same sort of information and could load hundreds, thousands, and even millions of records in a short period of time. It is therefore to import and export large amounts of data. This improves performance, for large amounts of data, quite significantly. In case of dealing with a B-tree, indexes are usually optimized for inserting rows one at a time. In case of adding a large amount of data, inserting rows one at a time may be inefficient, for instance, the optimal way for a B-tree is to insert a single key is very poor way of adding a bunch of data to an empty index. Instead, a different strategy is pursued with B-trees. All the data is presorted, and are grouped in blocks. Then, a new B-tree is built by transforming the blocks into tree nodes. Although both techniques have the same asymptotic performance, O(n log(n)), the bulk-load operation has much smaller factor.

First Fit Algorithm: In the Bin Packing problem, objects of different volumes must be packed into a finite number of bins or containers each of volume V in a way that minimizes the number of bins used. Optimal solutions to many instances of the

Bin Packing problem can be produced with sophisticated algorithms. In addition, many heuristics have been developed: for example, the First Fit Algorithm provides a fast but often non-optimal solution, involving placing each item into the first bin in which it will fit. It requires $theta(nlogn)$ time, where n is the number of elements to be packed. The algorithm can be made more effective by sorting the list of elements into decreasing order (sometimes known as the First Fit Decreasing Algorithm). Although this still does not guarantee an optimal solution, and for longer lists may increase the running time of the algorithm. It is known, however, that there always exists at least one ordering of items that allows First Fit to produce an optimal solution.

Least Square Adjustment: The least square adjustment is a model for the solution of an overdetermined system of equations based on the principle of least squares of Least squares adjustment is a model for the solution of an overdetermined system of equations based on the principle of least squares of observation residuals. It is used extensively in the disciplines of surveying, geodesy, and photogrammetry, the field of geomatics, collectively. The conventional Least Square Adjustment Techniques in practice is depending on categorizing different groups of variables which appear in the mathematical model. The measured variables are called observations, while the unknown variables that should be determined are called unknown parameters. The rapid development in Geomatics, Photogrammetry, Computer Vision, and Laser Scanning in the last three decades necessitate the development of modified adjustment techniques. These adjustment techniques could handle the integration of different collected data in the mathematical models in a more generic and unified way which is called the unified approach of least square adjustment as mentioned in [MA76].

Morphing: It is a special effect in motion pictures and animations that changes (or morphs) one image or shape into another through a seamless transition. Morphing algorithms continue to advance and programs can automatically morph images that correspond closely enough with relatively little instruction from the user. This has led to the use of morphing techniques to create convincing slow-motion effects. Morphing has also appeared as a transition technique between one scene and another, even if the contents of the two images are entirely unrelated. The algorithm in this case attempts to find corresponding points between the images and distort one into the other as they cross-fade. Morphing is heavily used today, where morphing effects are most designed to be seamless and invisible to the eye.

Selection: It chooses which objects to include in the representation (and which to omit), e.g., by considering only major roads and omitting small building shapes.

Aggregation: It is the combination of several objects into one shape, e.g., by representing several single neighboring buildings by a single shape.

Simplification: It removes minor details of a shape, e.g., small protrusions of a building footprint or small variations of a curve representing a road.

Saliency Map: Representing a Saliency Map of an image is a part of image segmentation. In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves,...) in images. More precisely, image segmentation is the process of assigning a label to every pixel in a image such that pixels with the same label share certain characteristics. The result of image segmentation is a set of segments that collectively cover the entire image or a set of contours extracted from the image. Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture.