



Technische Universität München  
Faculty of Civil, Geo and Environmental Engineering  
Department of Cartography  
Prof. Dr.-Ing. Liqiu Meng

# Procedural 3D modeling and visualization of geotypical Bavarian rural buildings in Esri CityEngine software

Ieva Dobraja

Master's Thesis

**Submitted:** January, 2015

**Study Program:** Cartography (Master)

**Supervisors:** Dr.-Ing. Mathias Jahnke (TUM)  
Dr.-Ing. Özgür Ertac (Esri Deutschland GmbH)

**Cooperation:** ESRI Deutschland GmbH

2015



## **Declaration of Authorship**

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

This paper was not previously presented to another examination board and has not been published.

---

Place and date

---

(Signature)



## **Abstract**

The aim of the Master's Thesis "Procedural 3D modeling and visualization of geotypical Bavarian rural buildings in Esri CityEngine software" is to establish a 3D model library for rural Bavarian buildings and to create a 3D model of geotypical buildings for the case study area. The purpose of this thesis is to test if the software CityEngine, which is based on procedural modeling and mainly used for creating a 3D content of urban areas, can be applied for the 3D modeling of rural areas.

As the area of interest with typical Bavarian buildings chosen for this thesis was Seefeld, a municipality in the district of Starnberg, located southwest of Munich. It lies on the northern coast of the Pilsen Lake.

The task of the thesis is to create a 3D model of geotypical Bavarian rural buildings for the case study area by using procedural modeling. This task includes four main steps. The first step is the collection of all necessary data and attributes of buildings to create a database for modeling. This step includes also the photo collection of typical façade textures. The second step is to generate CGA rule file. The next step is the creation of the 3D model in Esri CityEngine software, using the rule files created. At the end, the final results of the 3D model generated are published as CityEngine WebScene.

The final result is a procedurally generated 3D model of the case study area.

## **Key Words**

3D modeling, Procedural modeling, CityEngine, CGA, rule file, WebScene, Bavarian rural buildings



## Table of Contents

Abstract .....	i
Key Words.....	i
Table of Contents .....	ii
List of Figures .....	iv
List of Tables.....	vi
Abbreviations .....	vii
1 Introduction.....	1
1.1 Motivation .....	1
1.2 Objectives .....	1
1.3 Structure of the Thesis .....	1
2 Fundamentals and Applications of 3D Modeling .....	3
2.1 Related Work .....	3
2.2 Definition of 3D modeling .....	8
2.3 Procedural 3D modeling .....	8
2.4 Esri CityEngine as a tool for procedural 3D modeling .....	10
3 Case Study Area.....	15
3.1 Description of the case study area .....	15
3.2 Description of necessary data .....	19
4 Data Processing.....	22
4.1 Data preparation before import into Esri CityEngine.....	22
4.2 Data import into CityEngine.....	27
4.3 Steps of procedural modeling using CityEngine .....	28
4.4 Problems during the modeling process.....	30
5 Results.....	37
6 Summary .....	40
6.1 Conclusion .....	40
6.2 Problem overview .....	40
6.3 Future work.....	41
7 References .....	42



Table of Contents

---

Appendix .....45

Appendix A: Rule file for buildings.....46



## List of Figures

Figure 1-1: Structure of the thesis .....	1
Figure 2-1: Image-Based Modeling and Rendering (IBMR) (Luan et al. 2008) .....	4
Figure 2-2: The pipeline of the city creation tool. The dark boxes show the results and data structures of the tools in the white rectangles (Parish & Müller 2001).....	5
Figure 2-3: The scope of the shape (Müller et al. 2006) .....	5
Figure 2-4: A typical workflow of CityEngine (Watson et al. 2008).....	6
Figure 2-5: Principle of the procedural modeling (Radies 2013a) .....	7
Figure 2-6: 2D zoning plan (Radies 2013a) .....	7
Figure 2-7: 3D zoning plan (Radies 2013a) .....	8
Figure 2-8: Principles of the procedural modeling (Schubiger 2012).....	9
Figure 2-9: Efficiency of a procedural modeling in comparison with a manual modeling (Schubiger 2012).....	9
Figure 2-10: Overview of import and export data formats in the CityEngine software (Radies 2013b).....	10
Figure 2-11: The CityEngine user interface (Esri 2014b).....	11
Figure 2-12: Example of 2D data (Esri n.d.(a)) .....	12
Figure 2-13: A graphical representation of a rule file in CityEngine.....	13
Figure 2-14: Application possibilities of 3D city models created in CityEngine (Schubiger 2012) .....	13
Figure 2-15: Creation steps of a 3D city model in the CityEngine software (Esri n.d.(b)) .....	14
Figure 3-1: Location of Seefeld (Google 2014) .....	15
Figure 3-2: Case study area .....	16
Figure 3-3: Seefeld Palace (Seefeld24.de 2000) .....	17
Figure 3-4: Church in Seefeld (Gemeinde Seefeld 2012) .....	17
Figure 3-5: Town hall of Seefeld .....	18
Figure 3-6: Examples of typical buildings in Seefeld .....	19
Figure 3-7: Attribute table of buildings.....	20
Figure 3-8: Imagery base map.....	20
Figure 3-9: Height map .....	21
Figure 3-10: Examples of vegetation models (Esri 2014c).....	21
Figure 4-1: Attribute table of buildings with the calculated values .....	22
Figure 4-2: Building footprints in ArcMap before the simplification.....	23
Figure 4-3: Building footprints before the simplification in CityEngine.....	23
Figure 4-4: Generated buildings in CityEngine before the simplification of footprints .....	24
Figure 4-5: Building footprints after the simplification in ArcMap.....	24
Figure 4-6: Buildings generated in CityEngine after the simplification of footprints .....	25
Figure 4-7: Generated points for the locations of trees .....	25



Figure 4-8: Bavarian church model downloaded from 3D Warehouse (Goesgen 2014).....	26
Figure 4-9: Bavarian Church model edited for the Seefeld project .....	26
Figure 4-10: The 3D model of Bavarian church imported into the CityEngine scene.....	27
Figure 4-11: The building footprints and the street network imported into CityEngine scene.....	28
Figure 4-12: Representation of rule files.....	29
Figure 4-13: Standard CityEngine workflow (Viinikka 2014) .....	29
Figure 4-14: Doors of the buildings on the wrong façade.....	30
Figure 4-15: “Set First Edge” function .....	31
Figure 4-16: Result after changing “the First Edge” .....	31
Figure 4-17: Complex shape buildings with too many doors .....	32
Figure 4-18: Result of complex shape buildings.....	32
Figure 4-19: Building and street network mismatch .....	33
Figure 4-20: Result after moving the streets .....	33
Figure 4-21: Windows at the intersections of buildings .....	34
Figure 4-22: Buildings after avoiding window creation at intersections .....	34
Figure 4-23: Roof overhang problem.....	35
Figure 4-24: Building with created roof overhangs .....	36
Figure 5-1: 3D model of the case study area.....	37
Figure 5-2: Final results of the 3D model of the case study area.....	39
Figure 5-3: A car model and models of people added to the final result .....	39



**List of Tables**

Table 4-1: Overview of the data imported .....27



## **Abbreviations**

CAD – Computer Aided Design

CE – CityEngine

CGA – Computer Generated Architecture

DTM – Digital Terrain Model

DXF – Drawing Interchange File Format

GIS – Geographic Information Systems

JPEG – Joint Photographic Experts Group (a method used of lossy compression for digital images)

KMZ – Zip-compressed KML (Keyhole Markup Language) file

LoD – Level of Details

OBJ – a 3D format that describes 3D geometry

OSM – OpenStreetMap

SHP file – Shape file

TIFF – Tagged Image File Format (a computer file format for storing raster graphics images)



# 1 Introduction

## 1.1 Motivation

Nowadays 3D modeling and visualization of large cities have become more and more popular. Accordingly there are new solutions to make these processes faster and cheaper. The reason for their growing popularity is the wide variety of usage areas for 3D city models. They can be used further not only for research and education purposes in urban planning and design, but also for virtual reality in the movie and gaming industries (Parish & Müller 2001).

The best method for need to create a large size 3D city model in a short time and with low expenses is procedural modeling, which is a 3D model creation using rules and algorithms. It is a cost-effective solution for generating 3D urban contents (Watson et al. 2008).

Procedural modeling in the software Esri CityEngine has been mainly used for creating a 3D model of urban areas. The modeling is based on CGA (Computer Generated Architecture) rules. CGA is a scripting language quite appropriate for generating architectural 3D content (Parish & Müller 2001).

Previous researches in this field have showed that the software CityEngine is appropriate for modeling cities. The purpose of this Master's thesis is to test if these procedural modeling principles can be applied for modeling a 3D content of rural areas as well.

## 1.2 Objectives

The aim of the thesis is to establish a 3D model library for rural Bavarian buildings and to create a geotypical 3D model of the chosen case study area using procedural modeling.

To achieve the aim of the thesis following tasks were completed:

- Literature review of related work in 3D modeling and procedural modeling
- Data collection for the case study area
- Data preparation for the modeling process
- 3D model creation of the case study area
- Publication of the results as CityEngine WebScene

By means of the above mentioned steps, we can obtain an overview of typical Bavarian rural buildings in the chosen case study area.

## 1.3 Structure of the Thesis

This Master's Thesis consists of introduction, a theoretical part, a practical part, results and summary (figure 1-1).

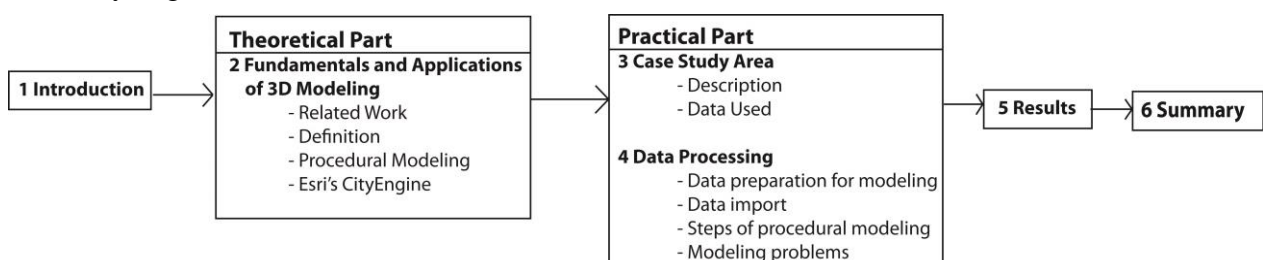


Figure 1-1: Structure of the thesis



The thesis starts with the introduction, which gives an overview of the topic, stated aim and the structure of the thesis.

The theoretical part of the thesis describes fundamentals and applications of 3D modeling. It includes an overview of related work in this field, a general explanation of 3D modeling, advantages of procedural modeling in comparison to manual modeling and an overview of Esri CityEngine as a tool for procedural 3D modeling.

The practical part of this thesis is focused on 3D modeling of rural areas. As the final result of the practical part, a 3D model of Seefeld, a municipality in the district of Starnberg located southwest of Munich, has been created using Esri's ArcGIS software for data preparation and CityEngine for the modeling process. This part of the thesis starts with a chapter about the case study area. It includes a description of the chosen area and of the data which were used for modeling.

The next chapter explains the data processing and the modeling process. It covers the steps of the data preparation before the import into modeling software and the data import, then explains the steps of procedural modeling and describes the problems which occurred and their solutions.

After the practical part of the thesis comes a chapter which gives an overview of the results achieved and describes publishing the results on the Web.

The last chapter of this Master's Thesis is the summary. It consists of conclusion, problem overview and future work. This part summarizes the results achieved in the thesis and gives suggestions for improvements and further research.



## 2 Fundamentals and Applications of 3D Modeling

### 2.1 Related Work

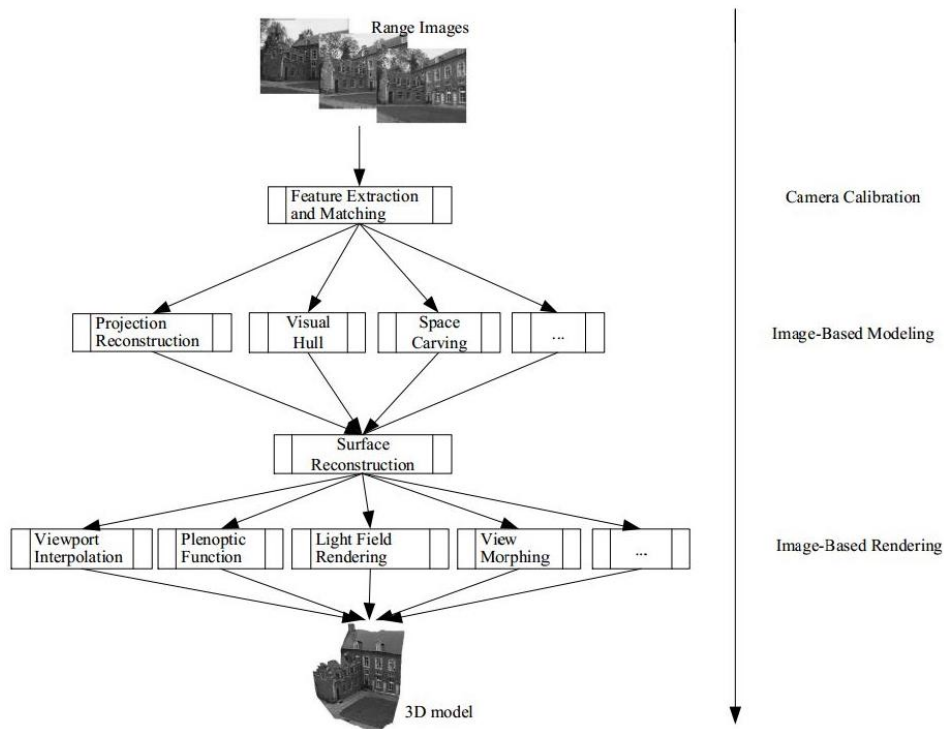
As a 3D modeling is being used in different application areas, such as animation, archeology, architecture, engineering and others, Luan et al. (2008) in their research have stated it as a key technique for many researches and applications.

Research by Luan et al. (2008) gives an overview of the process of 3D modeling which includes 3D data acquisition, modeling and rendering steps. There are given application possibilities of the 3D modeling as well.

In the 3D data acquisition section authors introduced with advantages and disadvantages of two methods which can be used to collect 3D data. One of the methods is Laser scanning system which can be applied to collect the 3D data of objects' surface as point clouds. The advantages of this method are its high scanning speed and accuracy. The disadvantages of this method include required equipment which is expensive and may be dependent on the weather conditions, for example, it may not work properly in the sun or rain. Another disadvantage of this method is the required post-processing of the data acquired. Post-processing step is needed for a better final result and usually has to be done manually. The second method of the data acquisition is Close Range Photogrammetric System which can be used for 3D data acquisition of the objects using calibrated cameras. Cameras can be fixed or hand-held. Calibration process (acquisition of camera's internal and external parameters) and image matching is easier for fixed camera system but systems with hand-held cameras can get object's 3D data of all sides. The advantage of this method is its relatively low costs.

To introduce with 3D modeling and rendering process, authors have described Image-Based Modeling and Rendering (IBMR) method which can model objects with different sizes and get the information of depth and texture at the same time. Figure 2-1 shows the steps of this method to create a 3D model from 2D images. The first step after the acquisition of range images is camera calibration by feature extraction and matching. This step is done by selecting and matching at least 7 pairs of points which are corresponding between two images to calculate a fundamental matrix and get a projection matrix. The next step in the modeling process is to estimate the depth information from a curvature information which is further used to reconstruct the surface of a model. Usually it is constructed by polygon mesh. After the surface reconstruction, the rendering process can be applied. There are several Image-Based Rendering methods available, for example, viewpoint interpolation, light field rendering, view morphing and others. This step is important for improving the final 3D model (Luan et al. 2008).

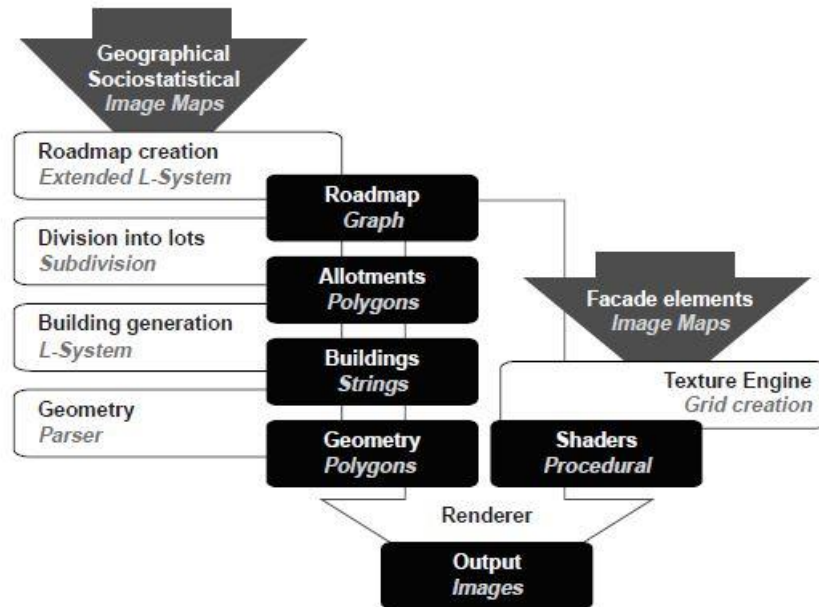




**Figure 2-1: Image-Based Modeling and Rendering (IBMR) (Luan et al. 2008)**

There are several researches in the field of procedural modeling for cities and buildings already made. For example, research by Parish and Müller (2001) in which they introduce the system “CityEngine” which is appropriate for modeling the whole city using comparatively small set of statistical and geographical input data. This system is highly controllable by the user and uses a procedural approach based on L-systems to model cities. Figure 2-2 shows the pipeline of the creation of city models using CityEngine system tools. At the beginning the roadmap has been created using L-system according to input data. Various image maps, for example, land-water boundaries and population density, can be used as input data. In the next step, the areas between the roads are subdivided and defined as the lots of buildings. After the lots of buildings have been defined, the generation of the buildings can be done by applying L-system. They are generated as a string representation of boolean operations on solid shapes. In the last step, a parser interprets all the results for the visualization software which should be able to process polygonal geometry and texture maps. Additionally, the mechanism to generate the façades of buildings can be incorporated into pipeline.

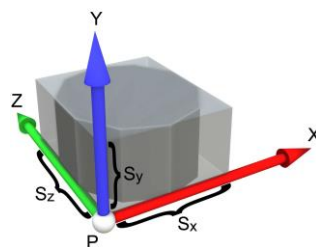




**Figure 2-2: The pipeline of the city creation tool. The dark boxes show the results and data structures of the tools in the white rectangles (Parish & Müller 2001)**

Research paper describes two different L-systems which can be used for the creation of the complete city. One L-system for a street creation and other one for a building generation (Parish & Müller 2001).

The other research related to the field of procedural modeling of cities and buildings is made by Müller et al. (2006). It gives an overview and a detailed description about the creation process of buildings using the shape grammar (CGA shape) and compares it with the L-systems. CGA shape is a shape grammar used for procedural modeling of CG architecture to produce building shells including high visual quality and geometric details. It is based on production rules to develop a design by creating more and more details of the shapes which consist of symbol, geometry (geometric attributes) and numeric attributes. Geometric attributes such as the position  $P$ , the coordinate system describing orthogonal vectors  $X$ ,  $Y$ ,  $Z$  and the size define an oriented bounding box called scope (figure 2-3). Scopes are used in the rules to make modifications, for example, a rotation, of the shapes.



**Figure 2-3: The scope of the shape (Müller et al. 2006)**

The procedural modeling process of buildings starts with the creation of a mass model which is a crude volumetric model of a building, and then the details of façades and roof are created.

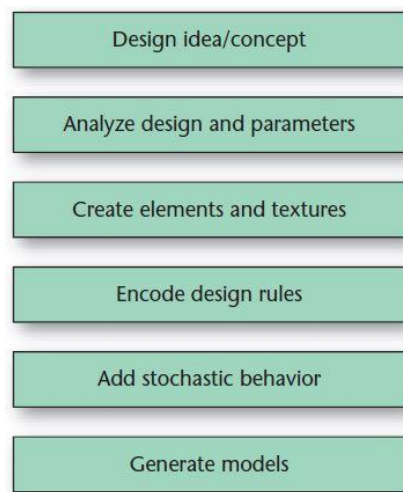


In comparison to the L-systems that capture a growth over time, the CGA shape is a sequential grammar and is suitable for characterization of the structure of a shape, for example, its spatial distribution of features and components. In their research Müller et al. (2006, p.622) have found out that: *“a direct application of L-Systems to architecture overemphasizes the idea of growth, a concept that is often counterproductive for the procedural modeling of buildings.”* The reason for this statement is that originally the L-systems were invented to simulate the growth of plants and they do not relate well to the modeling of buildings.

There are mentioned and explained issues which should be taken into account when modeling the buildings, for example, overlap (occlusion) and snapping. The purpose for the occlusion query is to test if there are intersections between shapes. It helps not to place façade elements such as doors and windows on the part of the façade which is occluded. The main function of snapping is to improve the layout of a façade structure. It can be implemented by altering the existing shape rules to snap to a dominant face or line in the repeat and subdivision rules.

There are described modeling strategies for a simple building model, an office buildings and single family homes in this paper (Müller et al. 2006).

Research by Watson et al. (2008) describes the typical workflow (figure 2-4) and the applications of CityEngine, and how the created models can be used in game industry. A typical workflow in the CityEngine begins with an idea or a concept. The idea can be stemmed from a photograph, a drawing, an architectural figure or it can be a completely new design concept. When the idea of a project is defined, analyses of a design to find the most important parameters for the creation of model elements and textures are made. In the next step, design rules with the previously collected parameters and elements can be encoded. After that randomization of parameters for stochastic rules to generate the whole city can be done. It is important to add randomness gradually to avoid a chaotic and uninteresting design. After these steps the 3D models can be generated (Watson et al. 2008).



**Figure 2-4: A typical workflow of CityEngine (Watson et al. 2008)**

To create a better and more realistic 3D urban content, several suggestions should be taken into account because as Watson et al. (2008, p.19) have stated: *“Urban content is something more than building placement and shape”*. To create more realistic urban content weathering and



wear elements can be applied to materials and models, for example, rust and dirt accumulation. Created city model looks more real if it populated with cars and pedestrians.

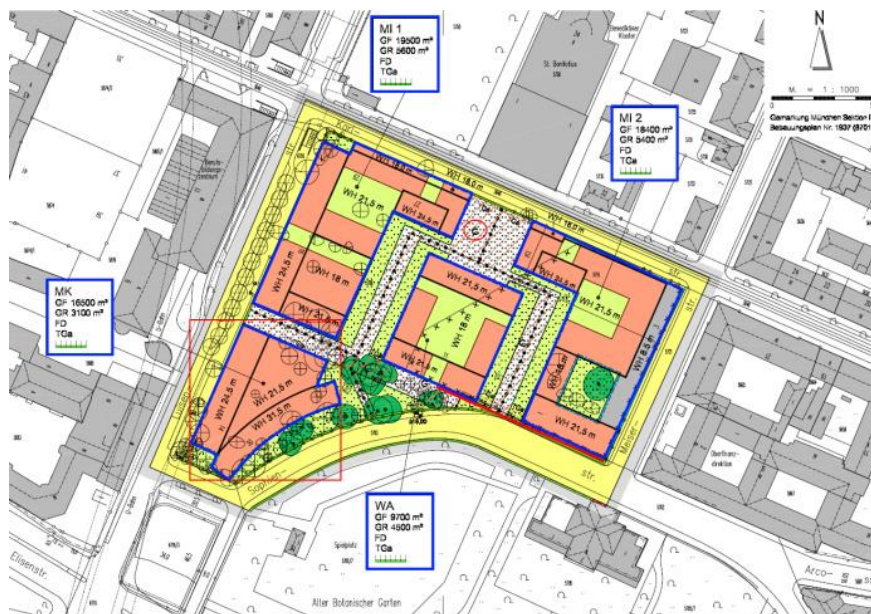
3D models created in the CityEngine have different applications, for example, in a cultural heritage, urban planning and in a movie industry. Procedural modeling can be applied in the 3D urban content creation for video games as well (Watson et al. 2008).

Research by Radies (2013a) describes 3D building modeling principle of using procedural modeling in the software CityEngine. Procedural modeling principle (figure 2-5) is based on shape grammars with production rules to generate detailed 3D objects out of simple initial geometric features. It starts with the extrusion of the initial shape to a block model which is later divided into shapes, for example, facades and a roof. To create a high level detailed model, created shapes are split into elements, for example, windows and doors, and textures or colors are added to these elements.



**Figure 2-5: Principle of the procedural modeling (Radies 2013a)**

There is given an example of procedurally generated 3D dynamic model of a zoning plan. 2D zoning plan (figure 2-6) which was used as a basic information for the generation of this plan in 3D, includes information of floor and space ratio, maximum permitted height, building types and other parameters. Zoning plans are used for planning a development of a city (Radies 2013a).



**Figure 2-6: 2D zoning plan (Radies 2013a)**

Taking into account information from the 2D zoning plan and using procedural modeling in the software CityEngine, the 3D zoning plan (figure 2-7) was created as the final result.





Figure 2-7: 3D zoning plan (Radies 2013a)

### 2.2 Definition of 3D modeling

According to Govil-Pai (2004), modeling is the first and the most critical pillar of 3D graphics. It can be defined as (Govil-Pai 2004, p.83) *“the process of creating a 3D model in the computer”*.

The process includes three main steps. The first step is a 3D data acquisition, the next step is modeling and the last step is rendering. In the data acquisition step, data about object's size and depth are collected. There are two main methods of data acquisition – contact (for example, measurements using tape) and non-contact (for example, laser scanning and measurements taken with total stations). The modeling step includes refinement of the initial shape, for example, surface construction. Main elements of this process are the shape of an object, spatial layout and connectivity of components that compose the object. The rendering of the model includes smoothing, morphing and other processing methods to make the final result look more natural (Luan et al. 2008). The model created in this process is a representation of a concrete or abstract entity to visualize the structure of it.

There are different types of representation possible:

- Quantitative representation in which equations are used to represent and describe system behavior
- Organizational representation when hierarchies are being used to represent classification schemes
- CG (Computer Graphics) in which models refer to geometrical representation of the entity

(Govil-Pai 2004)

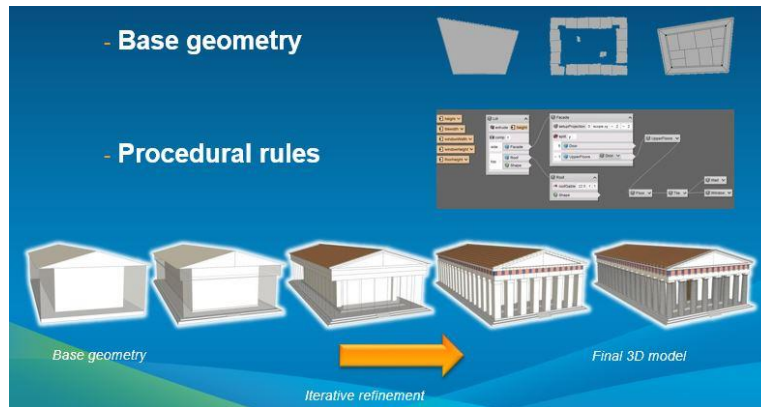
Nowadays the applications of 3D modeling are used in a wide range of areas, for example, animation, architecture, education, design, movies, engineering and others. It allows to model and simulate the real world (Luan et al. 2008).

### 2.3 Procedural 3D modeling

Procedural modeling is a 3D model creation process using rules and algorithms. It consists of a base geometry, for example, building footprints with the information about building heights

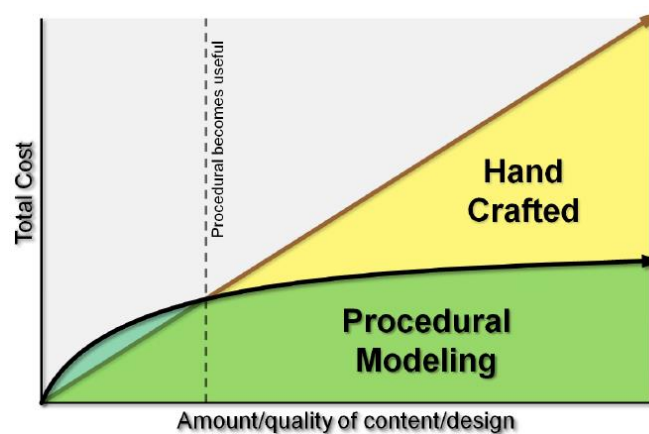


and roof types, and procedural rules. It allows progressively and iteratively to refine the model (Figure 2-8) (Esri 2012). In the process of generating a building procedurally, firstly the production rules create a primitive volumetric model (a mass model), secondly a façade is being structured and finally in the last steps, details for windows, doors and ornaments are added (Müller et al. 2006).



**Figure 2-8: Principles of the procedural modeling (Schubiger 2012)**

According to Müller et al. (2006), modeling of large size 3D environments requires a lot of financial resources and several man years' worth of labor. For this reason in creation of a large sized city model saving time and money is important. To fulfill these requirements, procedural modeling is one of the solutions. As the figure 2-9 shows, there is a definite cost and time saving benefit for using the procedural modeling when a lot of 3D modeling or design iterations are needed (Esri 2012).



**Figure 2-9: Efficiency of a procedural modeling in comparison with a manual modeling (Schubiger 2012)**

Procedural modeling has several advantages. Firstly, it makes the modeling process faster and randomness can be controlled. It is easy to update the rule files, for example, adding floors to the buildings and changing the roof type can be done by making the changes in the parameters and not manually modeling the object, and they are re-usable for other projects because the process of creation has been recorded (Piccoli 2013).



The next advantage of CityEngine is its variety of data formats supported for import and export. Figure 2-10 shows the overview of the data formats which can be imported into CityEngine for a modeling process and the data formats which can be exported as results. CityEngine supports geospatial data, 2D data, 3D assets (polygonal meshes) and Python scripting as input data types and offers a wide choice of generated 3D data export formats.

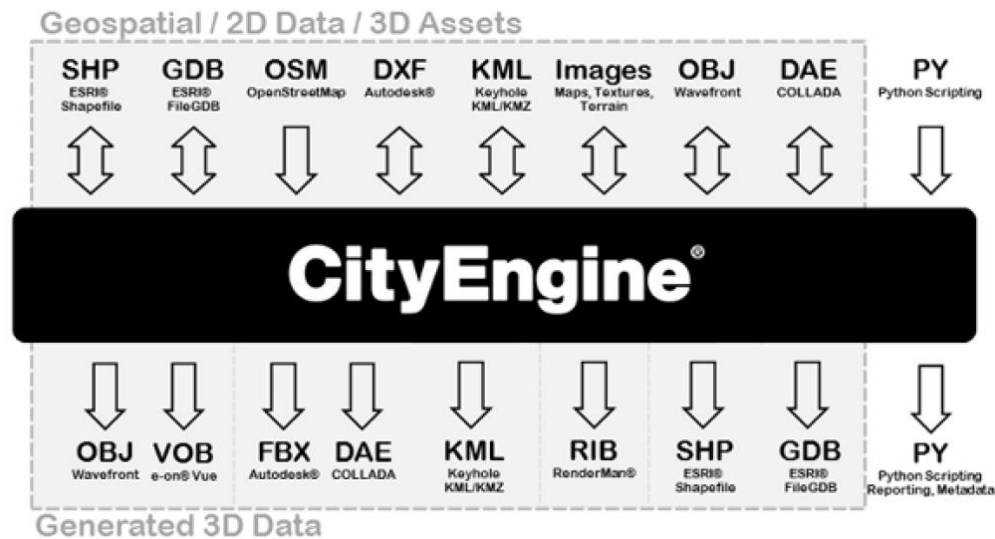


Figure 2-10: Overview of import and export data formats in the CityEngine software (Radies 2013b)

Further advantage of the procedural modeling is its possibility of dynamic editing. For example, when a street network is imported, software will generate parcels within the streets and 3D buildings will be generated within parcels and if the street is moved, then buildings will be moved as well because they have been connected procedurally (Esri 2012).

The disadvantage of the procedural modeling is that it is less user friendly than the manual modeling techniques (Piccoli 2013). Required knowledge of a CGA scripting to write rule files makes procedural modeling process more complicated at the beginning if a user has no experience with it.

### 2.4 Esri CityEngine as a tool for procedural 3D modeling

According to Watson et al. (2008), Esri CityEngine is one of the most mature procedural modeling tools which are available. It is a stand-alone software which transforms 2D GIS data into smart 3D City models. It improves urban planning, architecture and design (Esri n.d.(c)).

CityEngine combines procedural modeling methods with shape and split grammars to generate a 3D content. Shape grammars perform recognition of a particular shape and its possible replacement. The shapes which will be replaced and the manner in which it will be done are specified by using rules. Split grammars split a 3D object into its components, for example, faces, edges or vertices (Edvardsson 2013).

For a successful use of shape and split grammars in the modeling process two following steps should be taken into account and completed. The first step is the acquisition of a good understanding of basic building elements such as windows, doors, columns, walls and others.



The next step is the creation of grammar rules to combine these basic elements (Watson et al. 2008).

The user interface of CityEngine consists of several windows. A typical modeling session includes (figure 2-11):

- 1 “Scene Editor” window where the user can manage the scene, it is organized in layers.
  - 2 “Rule Editor” window where the CGA scripts are generated. “Rule Editor” can have a text and a graphical view.
  - 3 & 6 “Viewport” window where the generated 3D content is displayed.
  - 4 “Inspector” window which gives a detailed view editing possibilities of selected objects.
  - 5 “Navigator” window where all the files are stored. In this section user can manage and preview files in the workspace.
  - 7 “Log” window shows the log records. They are classified as information messages, warnings and errors.
  - 8 “Console” window.
- (Esri 2014b)

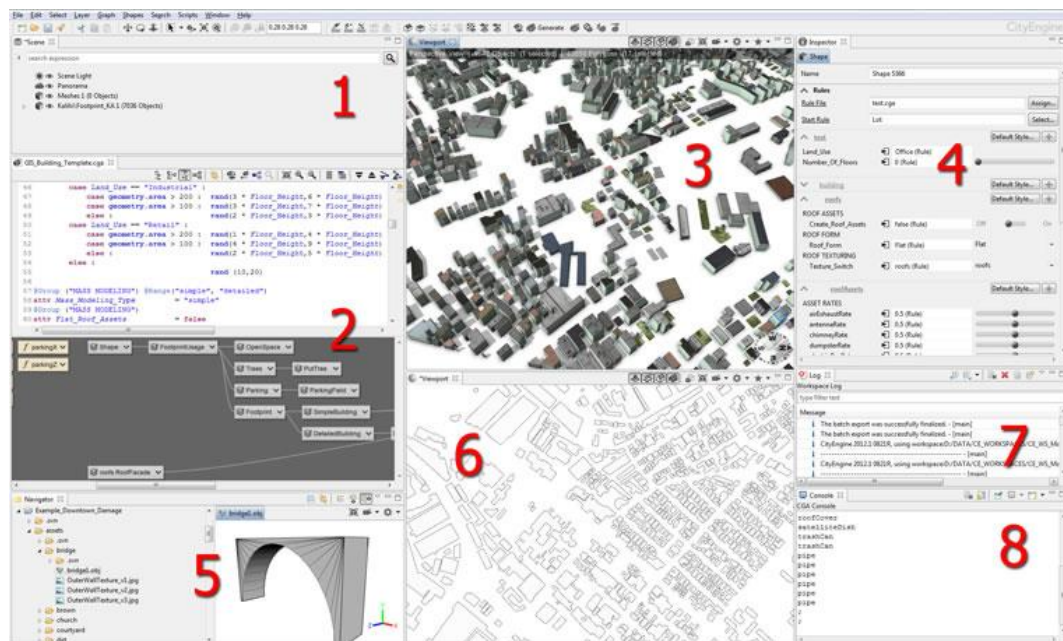


Figure 2-11: The CityEngine user interface (Esri 2014b)

Some of the elements which are required for a city modeling process in CityEngine are a terrain and control map layers, a street network, building footprints and a CGA rule file. Terrain consists of a height map or a texture map and map layers (images). A street network and building footprints can be automatically or manually created in CityEngine or imported from DXF and SHP files (Piccoli 2013).

The scripting language CGA (Computer Generated Architecture) which defines the rules or procedures to create a 3D content is the core of this software. CityEngine can be used for the whole procedural modeling development cycle because it includes tools for setting up the scene



with pre-existing data, modeling street and lot geometries, writing and analyzing the CGA script files, applying the CGA files to create 3D models and exporting the created models for further processing (Viinikka 2014).

Two primary attributes which connect the initial shape to the procedural generation process are “Rule file” which defines the CGA rule file assigned to the initial shape and “Start rule” which defines the production rule from which the creation process begins. Other attributes which describe the initial shape (for example, buildings heights, number of floors, roof shapes, etc.) can be used as parameters in the CGA production rules (Viinikka 2014).

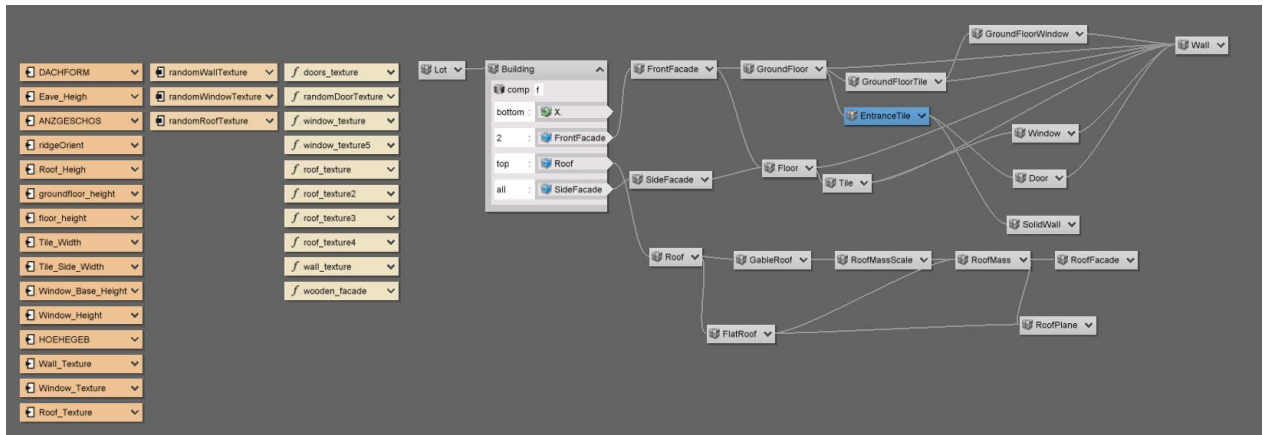
The advantage of CityEngine is the ability to create 3D urban scenes based on existing GIS data easily. A high-quality 3D content can be created by using only the combination of 2D data, attributes and procedurally defined rules. Components of CityEngine are feature geometry, feature attributes and procedurally defined rules. Figure 2-12 shows the example of 2D data for City of Philadelphia which consists of street centerlines, building footprints, multipatch footprints, street furniture and vegetation data.



**Figure 2-12: Example of 2D data (Esri n.d.(a))**

The complexity and the real-world accuracy of the generated model depends on the amount of the details provided for each of the CityEngine components (Esri n.d.(a)). The figure 2-13 shows an example of the graphical representation of a rule file in the CityEngine software. It consists of the attributes, the constants and the textures defined and the rule.





**Figure 2-13: A graphical representation of a rule file in CityEngine**

When the 3D city content using parametric rules is generated, a 3D city design creation and further analysis (for example, visibility and shadow analysis) of the model are possible (figure 2-14).



**Figure 2-14: Application possibilities of 3D city models created in CityEngine (Schubiger 2012)**

To sum up, a creation of a 3D city model in CityEngine can be done in 5 steps (figure 2-15):

- First of all, there should be a 2D geodatabase or 2D GIS data, CAD data or data from other formats which will be imported into CityEngine.
- After the import, GIS data can be edited and manipulated. This step includes 3D street, block and parcel creation as well.
- Next step includes 3D extrusion of buildings, generation of roofs, street furniture and other elements in the model.
- When the models have been created, the refinement of the model details using procedural rules, selection of the Level of Details (LoD) and adding vegetation can be done.
- The last step of the process is the export of the created city model to another 3D application. It can be rendered in another application, exported to a game engine and



shared online using Esri WebScene. It is also possible to export the created models back to a file geodatabase for more accurate 3D analysis in ArcScene (Sohail 2013), (Esri n.d.(b)).



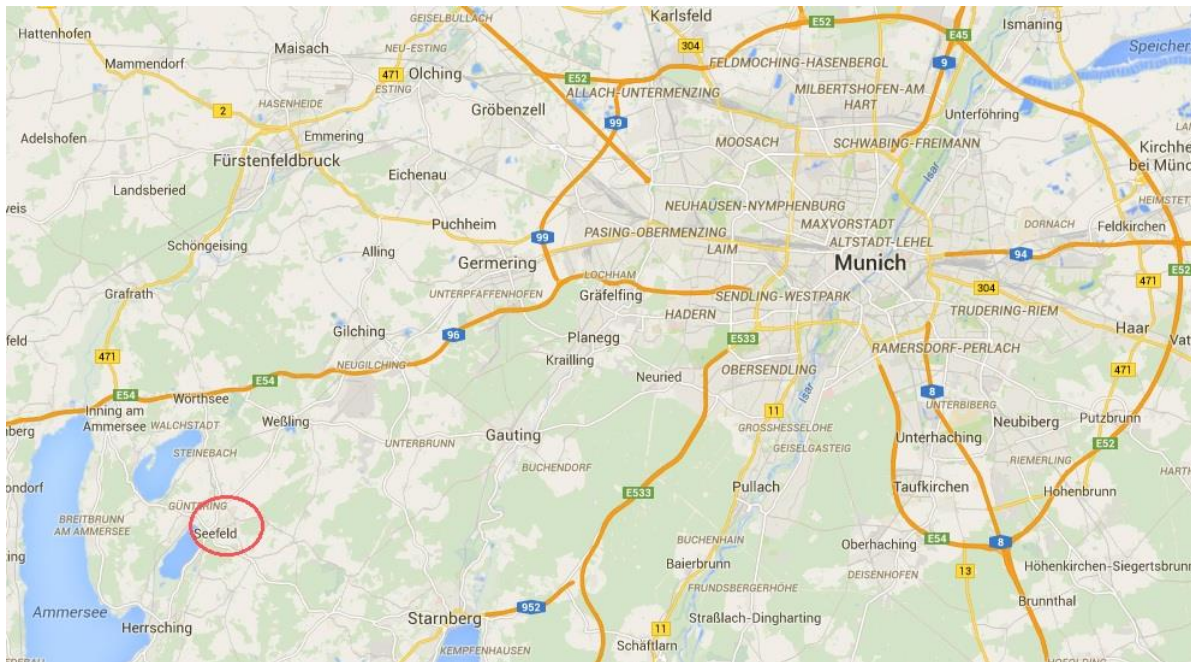
**Figure 2-15: Creation steps of a 3D city model in the CityEngine software (Esri n.d.(b))**



### 3 Case Study Area

#### 3.1 Description of the case study area

After exploration of the available geodatabase of buildings and consultations with the supervisors, Seefeld was chosen as the case study area. It is a municipality in the district of Starnberg, located southwest of Munich. It lies on the northern coast of the Pilsen Lake (figure 3-1). This area was chosen as the area of interest because typical Bavarian buildings can be found there and it is conveniently reachable by public transport.



**Figure 3-1: Location of Seefeld (Google 2014)**

As can be seen in the figure 3-2, Seefeld is a so called scattered (nucleated) village. This type of village is typical for Bavaria and the southern part of Germany.





**Figure 3-2: Case study area**

Nucleated villages are villages in which the houses cluster around a physical or conceptual nuclei, for example, the castle or the local church (Curtis 2013). These villages generally have an irregular shape, but they roughly circle around a central place and/ or church. In central Europe nucleated villages emerged as small settlement farmsteads which grew together with growing population (Wikimedia Foundation 2010).

Other types of villages are linear settlements and polyfocal settlements. For the linear type of village, houses are laid out in a linear row, usually along a dike, a main street or a waterway (Curtis, 2013). Polyfocal settlements have been developed through expansion of two or more small nuclei which originally were separate (Williamson 2012).

According to available information about the Seefeld municipality, important buildings in Seefeld are Seefeld Palace (figure 3-3), Sankt Peter and Paul Church (figure 3-4) and the town hall (figure 3-5). Seefeld Palace is a popular tourism object in this area. It was built in the 18<sup>th</sup> century and consists of several buildings (Gemeinde Seefeld 2012).





**Figure 3-3: Seefeld Palace (Seefeld24.de 2000)**

Sankt Peter and Paul Church is an important religious place and was built in baroque neo-gothic style (Gemeinde Seefeld 2012).



**Figure 3-4: Church in Seefeld (Gemeinde Seefeld 2012)**

The Town Hall of Seefeld is a significant building for questions related to city administration (Gemeinde Seefeld 2012).





**Figure 3-5: Town hall of Seefeld**

After the review of the database of buildings in Seefeld, typical parameters were found, for example, the number of floors and the roof shape. In this case study area, the number of floors for the buildings varies between 1 and 6 floors. Most of the buildings have 1 or 2 floors. Buildings which belong to the palace have 6 floors. The most typical roof type for the buildings in this area is the gable roof. There are some flat roofs as well.

From the photos taken during field work, we can distinguish the typical colors of façades and the window types. As can be seen in the figure 3-6, most of the façades of the buildings are light colored, for example, white, light yellow and grey. For this case study area typical are windows with shutters (figure 3-6 a and b). Flowers in front of windows are common as well (figure 3-6 c and d).



**a**



**b**





**Figure 3-6: Examples of typical buildings in Seefeld**

## 3.2 Description of necessary data

To create the 3D model of Seefeld, different types of data were used. At the beginning it was planned to use data downloaded from OpenStreetMap (OSM) (© OpenStreetMap contributors n.d.) for modeling the buildings, but after the download and exploration of the data, it was discovered that using it for modeling in CityEngine was impossible because it did not contain any information about the heights of the buildings and the roof shapes. Data for buildings downloaded from OSM contained only the footprints of the buildings. This information was not detailed enough for modeling because building heights and roof shapes are one of the most important attributes of the buildings. As a solution to this problem, ready prepared geodatabase from Esri<sup>1</sup>, containing the necessary data, was used. This database for buildings includes shapes of the building footprints and the attribute table with the building heights, the number of floors, the roof shapes (classified with codes) and other attributes (figure 3-7).

---

<sup>1</sup> In this case Esri Deutschland GmbH



### 3 Case Study Area

FID	Shape	GEID	DQDACH	DQLAGE	DQBODEN	BPDACH	DACHFORM	GEBFCT	HOEHEGRUND	HOEHEDACH	HOEHEGEB	ALKISOID	LAGEHNR	GDESCHL	ANZGESCHOS	STANDLK	Er
622	Polygon	3691044	1000	1000	2000	2200	1000	1001	559.56	576.641	17.081		Schloßhof 18 c b a	188132	6	12/3/2009	
738	Polygon	3734363	1000	1000	2000	2200	1000	1001	565.608	577.034	11.426		Hauptstraße 23	188132	3	12/3/2009	
1197	Polygon	3734675	1000	1000	2000	2200	1000	1001	548.775	556.037	7.262		Jahnweg 1	188132	2	12/3/2009	
0	Polygon	3734728	1000	1000	2000	2200	3100	1001	549.411	557.504	8.093		Ludwig-Thoma-Straße 3 a	188132	3	12/3/2009	
1	Polygon	3734729	1000	1000	2000	2200	3100	1001	551.092	559.008	7.916		Ludwig-Thoma-Straße 5 a	188132	2	12/3/2009	
2	Polygon	3734730	1000	1000	2000	2200	3100	1001	545.953	554.654	8.701		Ludwig-Thoma-Straße 6	188132	3	12/3/2009	
3	Polygon	3734731	1000	1000	2000	2200	3100	1001	544.638	553.28	8.642		Ludwig-Thoma-Straße 6 a	188132	3	12/3/2009	
4	Polygon	3734732	1000	1000	2000	2200	3100	1001	546.269	553.965	7.696		Ludwig-Thoma-Straße 4	188132	2	12/3/2009	
5	Polygon	3734733	1000	1000	2000	2200	3100	1001	546.705	555.504	8.799		Ludwig-Thoma-Straße 2 a	188132	3	12/3/2009	
6	Polygon	3734734	1000	1000	2000	2200	3100	1001	546.705	555.534	8.829		Ludwig-Thoma-Straße 2	188132	3	12/3/2009	
7	Polygon	3734735	1000	1000	2000	2200	3100	1001	546.528	554.995	8.467		Münchner Straße 11 a	188132	3	12/3/2009	
8	Polygon	3734736	1000	1000	2000	2200	3100	1001	547.158	555.2	8.042		Münchner Straße 11	188132	3	12/3/2009	
9	Polygon	3734737	1000	1000	2000	2200	3100	1001	548.229	555.8	7.571		Münchner Straße 9 a	188132	2	12/3/2009	
10	Polygon	3734738	1000	1000	2000	2200	3100	1001	548.451	555.87	7.419		Münchner Straße 9	188132	2	12/3/2009	
11	Polygon	3734739	1000	1000	2000	2200	3100	1001	545.514	553.178	7.664		Münchner Straße 21 a	188132	2	12/3/2009	
12	Polygon	3734740	1000	1000	2000	2200	3100	1001	545.514	552.968	7.454		Münchner Straße 21	188132	2	12/3/2009	
13	Polygon	3734741	1000	1000	2000	2200	3100	1001	546.016	553.258	7.242		Münchner Straße 19 a	188132	2	12/3/2009	
14	Polygon	3734742	1000	1000	2000	2200	3100	1001	545.052	552.208	7.156		Münchner Straße 15 a	188132	2	12/3/2009	
15	Polygon	3734743	1000	1000	2000	2200	3100	1001	546.016	553.378	7.362		Münchner Straße 19	188132	2	12/3/2009	
16	Polygon	3734744	1000	1000	2000	2200	3100	1001	546.228	553.871	7.643		Münchner Straße 17 a	188132	2	12/3/2009	
17	Polygon	3734745	1000	1000	2000	2200	3100	1001	545.07	552.158	7.088		Münchner Straße 15	188132	2	12/3/2009	
18	Polygon	3734746	1000	1000	2000	2200	3100	1001	544.246	552.268	8.022		Münchner Straße 13 a	188132	3	12/3/2009	
19	Polygon	3734747	1000	1000	2000	2200	3100	1001	544.246	552.238	7.992		Münchner Straße 13	188132	2	12/3/2009	
20	Polygon	3734748	1000	1000	2000	2200	3100	1001	546.228	553.841	7.613		Münchner Straße 17	188132	2	12/3/2009	
21	Polygon	3734749	1000	1000	2000	2200	3100	1001	559.794	568.331	8.537		Kirchenstraße 11	188132	3	12/3/2009	
22	Polygon	3734750	1000	1000	2000	2200	3100	1001	558.485	566.665	8.18		Anner 1	188132	1	12/3/2009	

**Figure 3-7: Attribute table of buildings**

For the street network, data downloaded from OSM was used. These data contain information about street width, type, name and more.

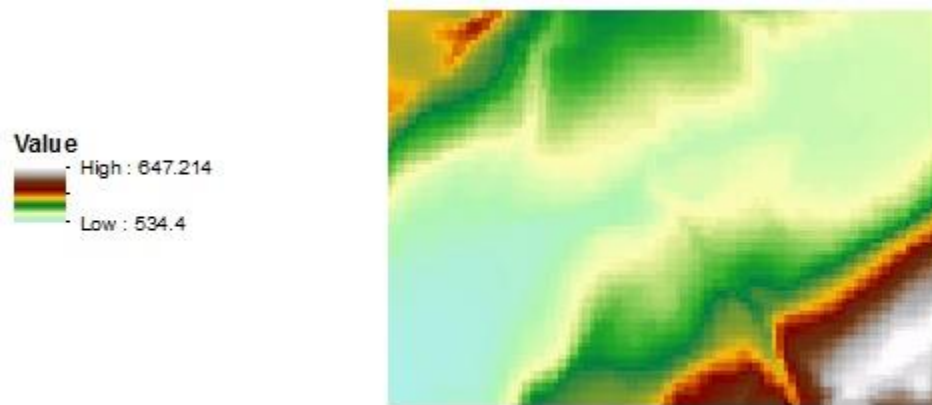
The terrain model consists of imagery base map (figure 3-8) and height map (figure 3-9). The imagery base map was taken from the ArcGIS base map collection. Relief data of DTM25 to create the height map were provided by the Bavarian State Office for Survey and Geoinformation. According to BKG (2012, p.3), DTM25 is a Digital Terrain Model which “*describes the terrain forms of the earth’s surface according to point quantity arranged in a regular grid*” with a width of 25 meters. The grid is georeferenced to planimetry and altimetry.



**Figure 3-8: Imagery base map**

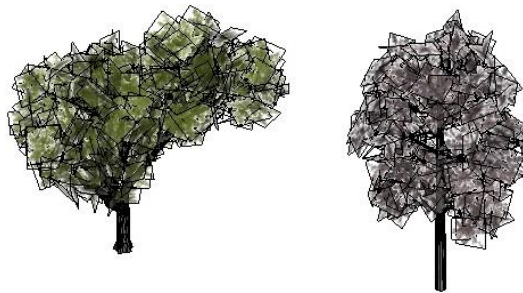


The height map shows the altitude in the area. As can be seen in the figure 3-9, in the case study area it varies between 534.4 and 647.214 meters.



**Figure 3-9: Height map**

For vegetation objects, elements from Esri 3D Vegetation Library were used (figure 3-10).



**Figure 3-10: Examples of vegetation models (Esri 2014c)**

Typical textures of façades, windows and doors for buildings were taken from real photos. For the façades of buildings lower than 4 meters, wooden façade texture from web source was used (ArchiExpo n.d.).

Typical roof textures for buildings were taken from Google Earth.

For the creation of the street network, textures were taken from web sources (Adnin 2005), (Vehicledonation100 2014) and the template “The Modern Street Example” (Esri 2013).



## 4 Data Processing

### 4.1 Data preparation before import into Esri CityEngine

Before import and start to model, it is necessary to make data preparation. It should be done for more successful modeling process and for better results. In this project data preparation included:

- calculation of necessary parameters of buildings,
- generalization of the building footprint shapes,
- creation of location and attributes for vegetation elements,
- adapting the downloaded 3D model of Bavarian church model to Sankt Peter and Paul Church in Seefeld.

For data preparation software Esri ArcGIS 10.2.2 was used. Data preparation started with the calculation of additional parameter values for the buildings. For modeling process it was necessary to calculate the roof height and the eave height. Figure 4-1 shows a part of the attribute table of the buildings with highlighted values for a building total height (HOEHEGEB), an eave height (Eave\_Heigh) and a roof height (Roof\_Ht).

DQ_LAGE	DOBODEN	BPDACH	DACHFORM	GEFBKT	HOEHEGRUND	HOEHEGEB	ALKISOID	LAGEHNR	GDESCHL	ANZGE	STANDLK	Eave_Heigh	Shape_Leng	Shape_Area	InBlid_FID	BLD	Roof_Ht
1000	2000	2200	3100	1002	581.949	583.777	1.828	Anger	188132	1	03.12.2009	1.828	32.575645	66.14505	1509	5	0.5
1000	2000	2200	3100	1002	574.101	575.935	1.834	Schreyeggsstraße 8	188132	1	03.12.2009	1.834	61.923722	212.808146	447	2	0.5
1000	2000	2200	3100	1002	558.872	560.742	1.87	Mellinger Weg 6	188132	1	03.12.2009	1.87	25.89793	34.562974	1700	5	0.5
1000	2000	2200	3100	1002	577.856	579.741	1.883	Schreyeggsstraße 18	188132	1	03.12.2009	1.883	53.628641	153.835457	445	1	0.5
1000	2000	2200	3100	1002	587.079	588.99	1.911	Höhenstraße 23	188132	1	03.12.2009	1.911	30.23704	51.927671	414	5	0.5
1000	2000	2200	3100	1002	583.259	585.184	1.925	Scheidegger Weg 4	188132	1	03.12.2009	1.925	26.875909	39.993058	119	5	0.5
1000	2000	2200	3100	1002	550.947	552.874	1.927	Ludwig-Thoma-Straße 1	188132	1	03.12.2009	1.927	26.843787	39.954681	281	5	0.5
1000	2000	2200	3100	1002	550.103	552.056	1.953	Ludwig-Thoma-Straße 1	188132	1	03.12.2009	1.953	35.790299	80.058042	280	5	0.5
1000	2000	2200	3100	1002	583.259	585.214	1.955	Nähe Scheidegger Weg	188132	1	03.12.2009	1.955	26.888969	40.017774	124	5	0.5
1000	2000	2200	3100	1002	589.859	591.828	1.969	Nähe Schützenstraße	188132	1	03.12.2009	1.969	30.581503	55.182229	1523	5	0.5
1000	2000	2200	3100	1002	567.771	569.771	2	Hedwigstraße 2	188132	1	03.12.2009	2	25.053349	35.350382	47	5	0.5
1000	2000	2200	3100	1002	557.913	559.913	2	Mellinger Weg 11	188132	1	03.12.2009	2	29.805946	46.653192	223	5	0.5
1000	2000	2200	3100	1002	577.035	579.035	2	Riedfeld	188132	1	03.12.2009	2	12.871342	9.715062	559	2	0.5
1000	2000	2200	3100	1002	544.959	546.959	2	Herrschinger Straße 1	188132	1	03.12.2009	2	12.662342	9.421149	1122	2	0.5
1000	2000	2200	3100	1002	552.136	554.136	2	Münchener Straße 6 c	188132	1	03.12.2009	2	25.965759	35.338611	1351	5	0.5
1000	2000	2200	3100	1002	554.848	556.848	2	Nähe Hartstraße	188132	1	03.12.2009	2	26.905726	40.125523	1653	5	0.5
1000	2000	2200	3100	1002	546.877	548.877	2	Saganger	188132	1	03.12.2009	2	12.967983	9.910559	1684	2	0.5
1000	2000	2200	3100	1002	585.042	587.048	2.006	Höhenstraße 11	188132	1	03.12.2009	2.006	42.499601	112.350529	360	5	0.5
1000	2000	2200	3100	1002	570.211	572.222	2.011	An der Breite 6 a	188132	1	03.12.2009	2.011	31.860549	58.714257	390	5	0.5
1000	2000	2200	3100	1002	562.414	564.437	2.023	Franz-Krämer-Straße 34	188132	1	03.12.2009	2.023	15.614293	14.394104	1302	2	0.5
1000	2000	2200	3100	1002	586.489	588.52	2.031	Höhenstraße 25	188132	1	03.12.2009	2.031	30.786268	58.865254	412	5	0.5

Figure 4-1: Attribute table of buildings with the calculated values

The roof height for the buildings was assigned according to their total height. Buildings were divided into three groups according to their total height – buildings lower than 4 meters, buildings with height between 4 and 6 meters and buildings which are 6 meters and higher. If the building total height was lower than 4 meters, the roof height was assigned as 0.5 meters. Buildings with the total height between 4 and 6 meters have the roof height 1 meter and buildings with the height 6 and more meters were assigned with the roof height of 2 meters.

The eave height for the buildings was calculated by subtracting a roof height from a total height of a building.

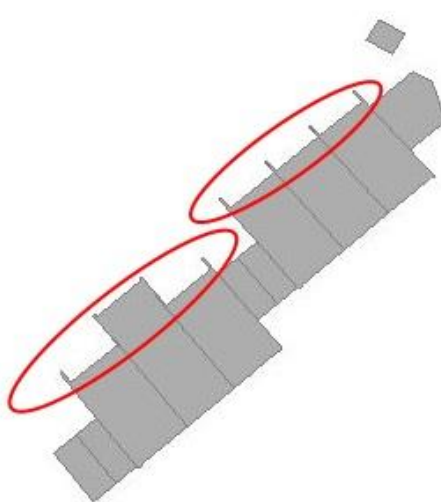
The next step in the data preparation process was the generalization of building footprints. It was done for a better appearance of the buildings. As can be seen in the figures 4-2, 4-3 and 4-4,



polygons of the building footprints contain extra points and the generated buildings do not look well enough.

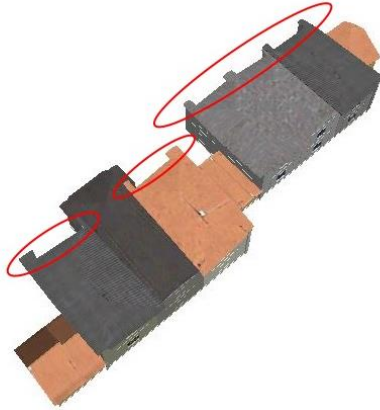


**Figure 4-2: Building footprints in ArcMap before the simplification**



**Figure 4-3: Building footprints before the simplification in CityEngine**





**Figure 4-4: Generated buildings in CityEngine before the simplification of footprints**

As CityEngine is more a design software and not a CAD software, buildings should not have exactly the same shape as initial footprints from geodatabase and a simplification of the shape of footprints can be made.

To avoid the problem that footprints have extra points in polygons, a simplification of buildings in ArcGIS with the tool “Simplify Buildings” from ArcToolbox (Esri 2014a) was used. When using this tool, the simplification tolerance has to be set. After trying different values of the simplification tolerance, the value of 3 meters was chosen as the most appropriate. The result of simplified building footprints in ArcMap can be seen in the figure 4-5.



**Figure 4-5: Building footprints after the simplification in ArcMap**

Figure 4-6 shows the result of the buildings which were generated from the simplified building footprints. As can be seen in the figure 4-6, buildings look better comparing to the buildings generated before the simplification in the figure 4-4 and they do not have extra points in the polygons.





**Figure 4-6: Buildings generated in CityEngine after the simplification of footprints**

After the generalization of the building footprints, the data preparation for vegetation import was done. For the creation of vegetation in CityEngine a SHP file with locations and attributes of the vegetation has to be created. Figure 4-7 shows the generated points for locations of trees in ArcMap. Points were created according to the image base map. The types and the sizes of trees were assigned randomly using Python scripting. For the tree height, there were chosen values between 12 and 17 meters. For tree types “beech”, “pine”, oak”, and “maple” were assigned.



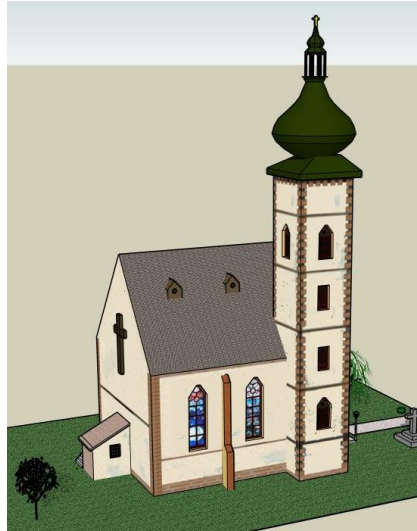
**Figure 4-7: Generated points for the locations of trees**

The last step in the data preparation process was the adaptation of the downloaded 3D model of Bavarian church model to Sankt Peter and Paul Church in Seefeld.

The reason for downloading the ready created model of the Bavarian church from 3D Warehouse (Goesgen 2014) (figure 4-8) and adapting it to the model was the shape of the church which is quite



complex. It would be time consuming to generate it. There should be a special rule file created for such type of building because its shape is completely different from other buildings in this area. The other reason for adding the model of Bavarian church to the project is that it makes the 3D model of Seefeld look more real.



**Figure 4-8: Bavarian church model downloaded from 3D Warehouse (Goesgen 2014)**

There were some color changes made in the model to make it more similar to Sankt Peter and Paul Church located in Seefeld. In the software Trimble SketchUp the location of the church was assigned according to Google Earth (figure 4-9). The location setting for the church is necessary for a successful import into CityEngine.



**Figure 4-9: Bavarian Church model edited for the Seefeld project**



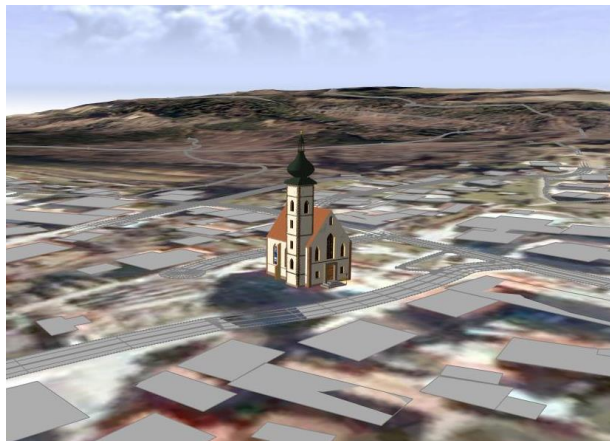
## 4.2 Data import into CityEngine

After the preprocessing, the import of the data into software CityEngine can be started. It starts with a creation of a new scene and setting a coordinate system of it. For this project WGS 1984 Web Mercator was chosen.

There were different types of data imported (table 4-1), for example, the building footprints and the locations of vegetation were imported as shape files, the street network downloaded from OpenStreetMap was imported as osm file and the 3D model of the Bavarian church (Goesgen, 2014) (figure 4-10) was imported as a KMZ file.

**Table 4-1: Overview of the data imported**

Data	Type	Source
Building footprints	SHP	Esri Deutschland GmbH
Street network	OSM	OpenStreetMap
Locations of vegetation	SHP	Assigned according to imagery base map (ArcGIS base map collection)
Vegetation elements	OBJ	Esri 3D Vegetation Library
Imagery base map	JPEG	ArcGIS base map collection
Height map	TIFF	Generated from the DTM25 provided by the Bavarian State Office for Survey and Geoinformation
3D model of the Bavarian Church	KMZ	3D Warehouse
3D model of a car	KMZ	3D Warehouse
3D models of people	KMZ	3D Warehouse



**Figure 4-10: The 3D model of Bavarian church imported into the CityEngine scene**



Terrain of the case study area was generated from the imagery base map and the height map. After the import, all the shapes were aligned to the terrain (figure 4-11).



**Figure 4-11: The building footprints and the street network imported into CityEngine scene**

### 4.3 Steps of procedural modeling using CityEngine

Creation of the 3D model using procedural modeling was done with the software Esri CityEngine 2014. Procedural modeling process in CityEngine software starts with a rule file creation. According to Esri (2014b), “*a rule file is a collection of attributes, functions and rules*”. Rule files can be created as scripts (appendix 8.1) or graphically (figure 4-12).



## 4 Data Processing

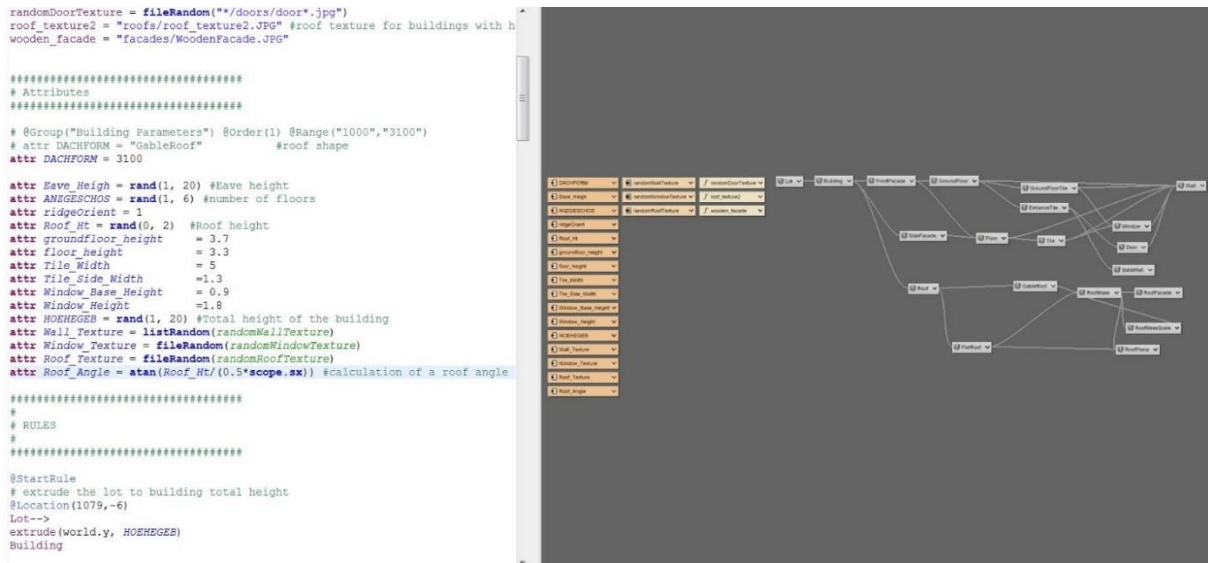


Figure 4-12: Representation of rule files

At the beginning user has to define all the attributes, constants, functions and textures which will be necessary for further modeling process.

Attributes, functions and constants are used to control the operations defined in the file and they work as parameters to the operations used for modeling. Parameter values can be set or changed by user as well in the Inspector window. It gives user more control over modeling process.

When starting to write the rule, the initial shape from which the generation process begins should be defined. It can be created with CityEngine tools or created in other software and imported into CityEngine. The final 3D result is a result of a sequence of transforming operations (for example, extrusions, rotations, etc.) applied on the initial shape (Viinikka 2014).

When the rule file has been created, it is applied to the initial shapes and the 3D content is generated (figure 4-13).



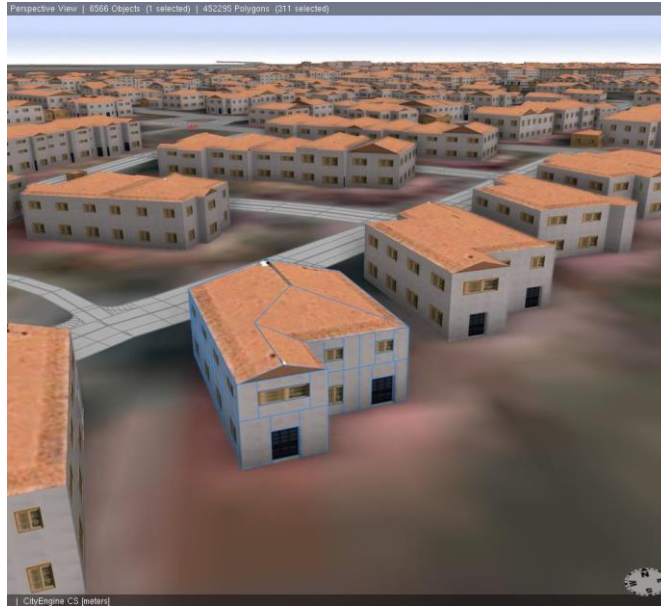
Figure 4-13: Standard CityEngine workflow (Viinikka 2014)



### 4.4 Problems during the modeling process

During the modeling process five problems with the generated buildings occurred. For example, too many doors were created for one building and on the wrong façade, buildings were mismatched with the imported street network, windows and/ or doors were created at the intersections of buildings, and problems occurred with creating roof overhangs. Further these problems and their solutions are described.

1. Doors were created on the façade which does not face a street (figure 4-14).



**Figure 4-14: Doors of the buildings on the wrong façade**

In a 3D city model the main door of the building in the most cases is created on the façade which faces a street but in this project software in some cases created the doors on the façades which face other house or a yard but not a street. The reason for this problem could be different data sources of the building footprints and the street network and the fact that they were imported, not generated in the CityEngine software. To solve this problem, the function “Set the First Edge” was used (figure 4-15). First edge defines which façade will be the Front façade. Usually it is the façade which faces a street.





**Figure 4-15: “Set First Edge” function**

Figure 4-16 shows the result after setting the street façade as the First Edge. In this case it was done manually looking through the whole model. The model consists of 1795 buildings and it took around 5 hours to change the Front façade. As the case study area is a rural area, the entrance of the house is not always from the main road. It can be from a smaller path or a yard as well.



**Figure 4-16: Result after changing “the First Edge”**

### 2. Too many doors for one building (figure 4-17).

In this project, there were quite many buildings which had a complex shape (a shape which is more complex than a rectangular-shaped). For this type of the buildings doors were created on the each façade which faced the street (was defined as the Front façade).





**Figure 4-17: Complex shape buildings with too many doors**

To solve this problem changes in the rule file in splitting the building was made. Buildings have been split into a bottom, a front façade, a roof and a side façade (appendix 8.1, line 53-57). To avoid placing doors on all façades which face front in the complex buildings, the front façade was defined with an index.

After this way of split, doors were created only on one façade (figure 4-18).



**Figure 4-18: Result of complex shape buildings**

3. Buildings were mismatched with the street network (figure 4-19).

In some cases there was a misalignment of the street network, the footprints of buildings and the imagery base map. This misalignment resulted into problem that in some cases the buildings intersected the street network. The reason for this problem could be the different data sources for the model.





**Figure 4-19: Building and street network mismatch**

In this case, to solve the problem of the mismatch of the data layers, the imagery base map was chosen as a reference layer. Streets were moved according to the base map and to avoid crossing with the buildings (figure 4-20). When the osm file with street segments is imported into the CityEngine software, it automatically creates streets with its sidewalks. It can be mismatched with the imagery base map layer and intersect with the building layer.



**Figure 4-20: Result after moving the streets**

4. Windows and/or doors created at the intersections of buildings (figure 4-21).





**Figure 4-21: Windows at the intersections of buildings**

In some cases in the model of the case study area buildings were located next to each other and had intersections.

To avoid windows and door placing at the intersections of buildings, the changes in the rule file were made. There were added code lines for the cases when window (appendix 8.1, line 128-129) or door (appendix 8.1, line 138-139) touches other elements. If a window or a door touches elements of other buildings next to it, then a wall is created instead of windows and doors. Figure 4-22 shows the result when placing the windows at intersections is avoided.



**Figure 4-22: Buildings after avoiding window creation at intersections**

### 5. Creation of the roof overhangs.

Overhangs of the gable roofs were created to make the buildings look more realistic. To create the roof overhangs the operation “`roofGable(angle, overhangX, overhangY, even, index)`” (Esri 2014b) was used. 5 parameters were used for this operation:



- *angle* – the angle of the roof-planes
- *overhangX* – the distance of overhangs which are perpendicular to the ridges (eave overhangs)
- *overhangY* – the distance of overhangs which are in the direction of the ridges (rake overhangs)
- *even* – the parameter which sets the generated gable roof to be even or not.
- *index* – an edge index which is used to control the orientation of the ridge. In this project the index parameter was set to “1” which means that a ridge is oriented along the longest ridge (Esri 2014b)

At the beginning to test the appearance of the roofs generated, test default values were set for the attributes which describe the roof angle and the distances of the overhangs. The attribute roof angle for all the buildings was set to 45 degrees and overhangs were set to 1 meter. As can be seen in the figure 4-23, there was a problem with creation of the roof, it split the façades and it was not created above the eave height.



**Figure 4-23: Roof overhang problem**

To solve this problem, the attribute which describes the roof angle, was defined as a function (appendix 8.1, line 41) which calculates the roof angle for each building according to its base length and the roof height. To avoid the intersection between a roof and façades and to avoid the cuts between the roof planes, t operation for lifting up the textured projection of the roof within RoofPlane tag was added (appendix 8.1, lines 223 and 227). The function of the t operation is used to translate the scope by the vector  $(t_x, t_y, t_z)$ , where parameters  $t_x$ ,  $t_y$  and  $t_z$  describe the amount to translate in each direction (Esri 2014b). As can be seen in the figure 4-24, after these corrections there are no intersections between the roof and the façades.



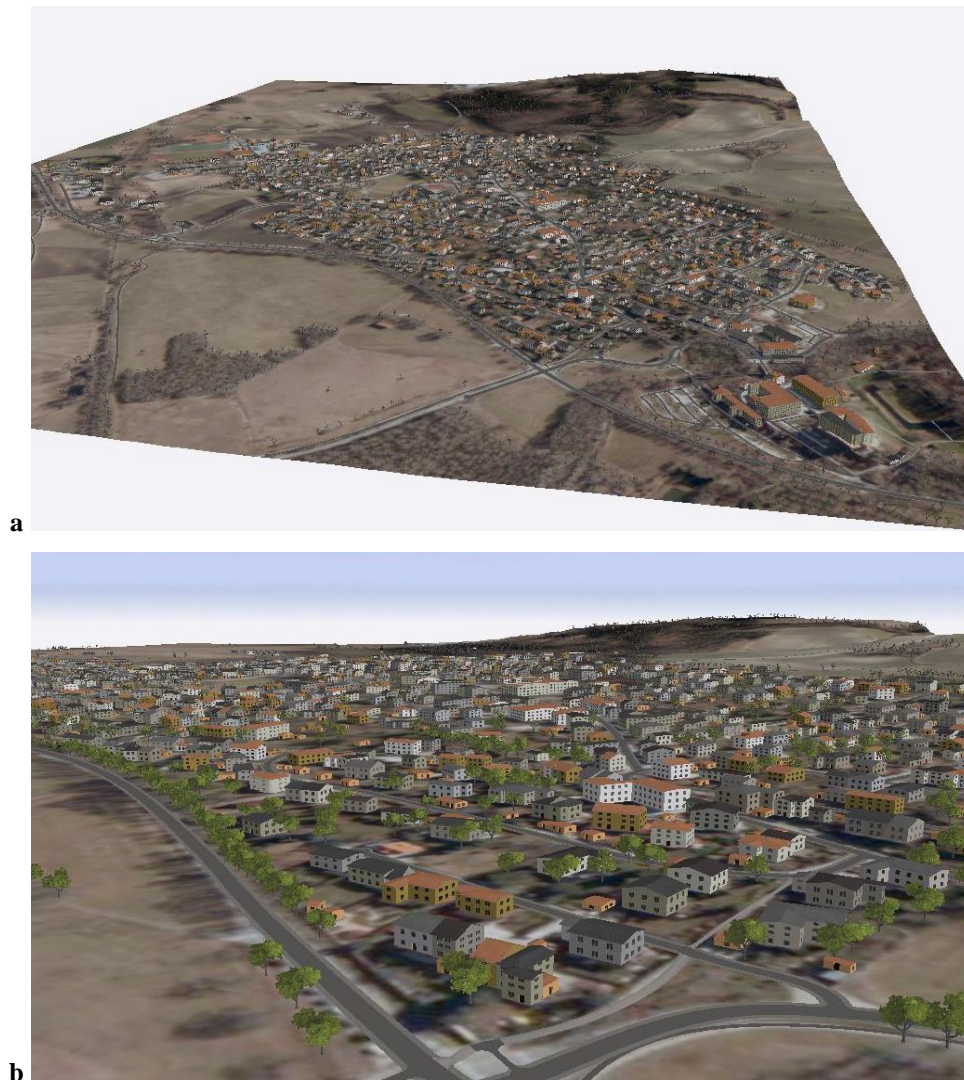


**Figure 4-24: Building with created roof overhangs**



## 5 Results

The final result of this Master's Thesis is a procedurally generated 3D model of Seefeld. The model includes buildings, street network, trees and the relief with the imagery base map (figure 5-1 a and b).



**Figure 5-1: 3D model of the case study area**

The final export to the CityEngine WebScene includes not only the 3D content generated but bookmarks, the current view and lightning settings of the scene as well. These settings should be made before the export step and they improve the visual representation of the scene.

Figure 5-2 a, b, c and d shows the generated 3D content of the case study area. It represents the typical Bavarian rural buildings in Seefeld. It shows façade and window textures which are



## 5 Results

---

common in this area. The content was generated procedurally by using initial shapes and procedural rules.

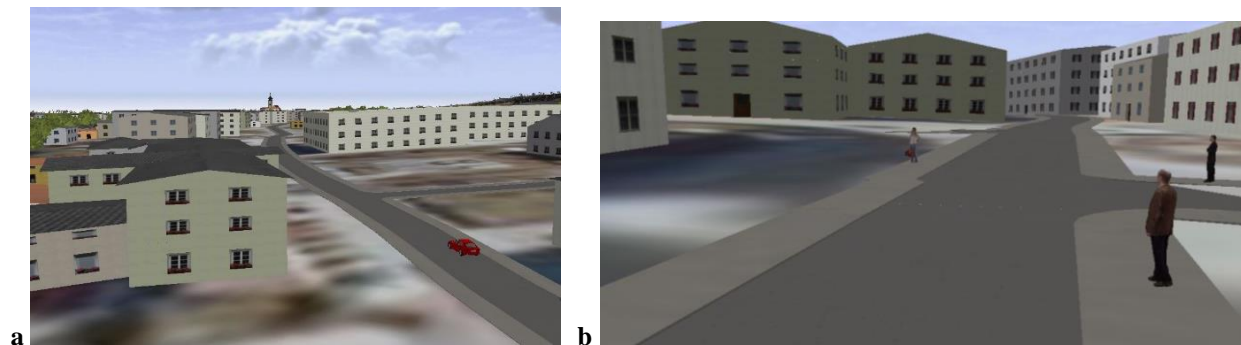






**Figure 5-2: Final results of the 3D model of the case study area**

Additionally, there were added 3 models of people (mrc.rbg 2014), (Gulmez 2014a), (Gulmez 2014b) and a car model (FreshPrince 2014) to make the final result look more realistic (figure 5-3 a and b).



**Figure 5-3: A car model and models of people added to the final result**

The final result of the 3D content generated in the CityEngine can be viewed not only as a WebScene but also in the ArcGIS software because CityEngine allows creating a rule package of the project which can be imported into ArcScene for 3D visualization.



## 6 Summary

### 6.1 Conclusion

Procedural modeling is one of the most appropriate solutions for creating large size 3D city models in a short time and with low expenses because, compared to manual modeling, 3D geometries and textures are constructed using rules and algorithms. It allows the modeler to make changes in the 3D content more easily, and already created rules can be re-used for other projects.

The modeling process consisted of data collection, preparation, import into the software, model creation and export for the model created.

For creation of the 3D model of geotypical Bavarian rural buildings in the case study area of Seefeld, the following data types were collected:

- geodatabase with shape files of the building footprints and attribute table with necessary parameters for modeling
- OSM data for street network
- DTM25 relief data for creation of height map
- imagery base map
- vegetation element models from “Esri 3D Vegetation Library”
- photos of typical textures of building façades and elements
- typical roof textures taken from Google Earth

All these data types were combined to create the 3D content of the case study area.

Data preparation step included selection of the coordinate system for the data, calculation of necessary parameters, simplification of building footprints and creation of locations for vegetation objects.

The modeling process started with importing prepared data into software and creating rule files. After that, the rule files created were applied to the initial shapes, and the 3D content was generated.

As can be seen from the results achieved, procedural modeling in the software CityEngine can be applied for modeling a 3D content of rural areas.

### 6.2 Problem overview

During the modeling process problems occurred. For example, doors were created on the edge which does not face the street, windows and/or doors were created at the intersections of buildings, buildings were mismatched with the street network and problems occurred with creating roof overhangs. For all of these problems solutions were found.



### **6.3 Future work**

The procedurally created 3D model of Seefeld can be used for further analysis and planning purposes. The CGA code created can be applied to generate a realistic 3D model of other similar rural areas.

The model can be improved by creating different Levels of Details (LoD). The content of the model could be improved by adding more infrastructure elements and details of the buildings.



## 7 References

- © OpenStreetMap contributors, n.d. OpenStreetMap. Available at: <http://www.openstreetmap.org/> [Accessed May 27, 2014].
- Adnin, M., 2005. Dusty dirt road. Available at: [http://mayang.com/textures/Nature/images/Soil and Earth/dusty\\_dirt\\_road\\_6190196.JPG](http://mayang.com/textures/Nature/images/Soil and Earth/dusty_dirt_road_6190196.JPG) [Accessed October 22, 2014].
- ArchiExpo, n.d. Wood siding. Available at: [http://img.archiexpo.com/images\\_ae/photo-g/wood-siding-70217-4856255.jpg](http://img.archiexpo.com/images_ae/photo-g/wood-siding-70217-4856255.jpg) [Accessed May 21, 2014].
- BKG, 2012. *Digital Terrain Model Grid Width 25 m*, Available at: [http://www.geodatenzentrum.de/docpdf/dgm25\\_eng.pdf](http://www.geodatenzentrum.de/docpdf/dgm25_eng.pdf).
- Curtis, D.R., 2013. The Emergence of Concentrated Settlements in Medieval Western Europe : Explanatory Frameworks in the Historiography. *Canadian Journal of History*, 48(2), pp.223–251.
- Edvardsson, K.N., 2013. *3d GIS modeling using ESRI's CityEngine*. The University Jaume I.
- Esri, n.d.(a). ArcGIS Resources. 3D urban content creation and design using CityEngine. Available at: <http://resources.arcgis.com/en/communities/city-engine/01w900000008000000.htm> [Accessed August 8, 2014a].
- Esri, 2014a. ArcGIS Resources. ArcGIS Help 10.2, 10.2.1, and 10.2.2. Available at: <http://resources.arcgis.com/en/help/main/10.2/index.html#//007000000000z000000> [Accessed October 26, 2014].
- Esri, 2012. ArcGIS Resources. Developing with Esri CityEngine. Available at: <http://video.arcgis.com/watch/1147/developing-with-esri-cityengine> [Accessed August 4, 2014].
- Esri, n.d.(b). ArcGIS Resources. What is CityEngine? Available at: <http://resources.arcgis.com/en/communities/city-engine/01w900000000m000000.htm> [Accessed October 23, 2014].
- Esri, 2014b. CityEngine Help. Available at: <http://cehelp.esri.com/help/index.jsp> [Accessed August 26, 2014].
- Esri, n.d.(c). Esri CityEngine. Available at: <http://www.esri.com/software/cityengine> [Accessed August 8, 2014].
- Esri, 2014c. Esri Vegetation Library with LumenRT 3D Plants – Plant Loader. Available at: <http://www.arcgis.com/home/item.html?id=e49b1fb0f56e40c19ff6e7ad4e546dad> [Accessed June 16, 2014].



- Esri, 2013. Example Modern Streets 2012. Available at:  
<http://www.arcgis.com/home/item.html?id=50e9b9b6c3ea4617b5b4a7bfd4e7f38c>  
[Accessed October 17, 2014].
- Gemeinde Seefeld, 2012. Gemeinde Seefeld. Available at: <http://www.seefeld.de> [Accessed August 8, 2014].
- Goesgen, 2014. Bavaria Church. *3D Warehouse*. Available at:  
<https://3dwarehouse.sketchup.com/model.html?id=48440627f174e0d011315e527a08c69b>  
[Accessed August 5, 2014].
- Google, 2014. Google Maps. Available at: [www.maps.google.de](http://www.maps.google.de) [Accessed May 13, 2014].
- Govil-Pai, S., 2004. Principles of Computer Graphics. In New York: Springer, pp. 83–130.
- Gulmez, M., 2014a. 3D People. *3D Warehouse*. Available at:  
<https://3dwarehouse.sketchup.com/model.html?id=9521c2b15ae329c0a6c6460b19930644>  
[Accessed October 29, 2014].
- Gulmez, M., 2014b. 3D People. *3D Warehouse*. Available at:  
<https://3dwarehouse.sketchup.com/model.html?id=848098c04c3d72c2a6c6460b19930644>  
[Accessed October 29, 2014].
- Luan, X.-D. et al., 2008. Research and Development of 3D Modeling. *IJCSNS International Journal of Computer Science and Network Security*, 8(1), pp.49–53.
- mrc.rbg, 2014. gym girl 3d. *3D Warehouse*. Available at:  
<https://3dwarehouse.sketchup.com/model.html?id=fdd0793fad4006f81518055461846b23>  
[Accessed October 29, 2014].
- Müller, P. et al., 2006. Procedural Modeling of Buildings. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 25(3), pp.614–623.
- Parish, Y.I.H. & Müller, P., 2001. Procedural Modeling of Cities. In Los Angeles: SIGGRAPH, pp. 301–308.
- Piccoli, C., 2013. CityEngine for Archaeology. In Amsterdam: 3D GIS for mapping the Via Appia.
- Radies, C., 2013a. Procedural Random Generation of Building Models Based Geobasis Data and of the Urban Development with the Software CityEngine. In Bernburg, Germany, pp. 175–184.
- Radies, C., 2013b. *Prozedurale Generierung von Gebäude- modellen aus Geobasisdaten und Daten der Bauleitplanung mit der Software CityEngine*. Technische Universität München.



- Schubiger, S., 2012. Developing with Esri CityEngine. In San Diego: Esri International User Conference.
- Seefeld24.de, 2000. Schloß Seefeld. Available at: <http://www.seefeld24.de/tourismus/galerie.php> [Accessed August 8, 2014].
- Sohail, A., 2013. Creating Smart 3D City Models with Esri CityEngine. Available at: <http://www.youtube.com/watch?v=zScyrwQLf0k> [Accessed June 27, 2014].
- Vehicledonation100, 2014. Sidewalk Texture. Available at: <http://vehicledonation.info/sidewalk-texture/> [Accessed October 22, 2014].
- Viinikka, J., 2014. *Adopting Procedural Information Modeling in Urban Planning*. Aalto University.
- Watson, B. et al., 2008. Procedural Urban Modeling in Practice. *IEEE Computer Graphics and Applications*, 28(3), pp.18–26.
- Wikimedia Foundation, 2010. Nucleated village. Available at: <http://en.academic.ru/dic.nsf/enwiki/5367161> [Accessed August 11, 2014].
- Williamson, T., 2012. *Environment, Society and Landscape in Early Medieval England*, Woodbridge: The Boydell Press.



## **Appendix**

Appendix A: Rule file for buildings

Appendix B: CE project of Seefeld (see Data-CD)



## Appendix A: Rule file for buildings

```

1  /**
2   * File:      buildings_new_rule.cga
3   * Created:   14 Aug 2014 12:44:33 GMT
4   * Author:    iedo
5   */
6
7  version "2014.0"
8  #####
9  # Constants
10 #####
11 const randomWallTexture = fileRandom("/facades/facade*.jpg")
12 const randomWindowTexture = fileRandom("/windows/window*.jpg")
13 const randomRoofTexture = fileRandom("/roofs/roof*.JPG")
14
15 #####
16 # Textures
17 #####
18 randomDoorTexture = fileRandom("/doors/door*.jpg")
19 roof_texture2 = "roofs/roof_texture2.JPG" #roof texture for buildings with
20 height less than 4m
21 wooden_facade = "facades/WoodenFacade.JPG"
22
23 #####
24 # Attributes
25 #####
26 attr DACHFORM = 3100
27 attr Eave_Heigh = rand(1, 20) #Eave height
28 attr ANZGESCHOS = rand(1, 6) #number of floors
29 attr ridgeOrient = 1
30 attr Roof_Ht = rand(0, 2) #Roof height
31 attr groundfloor_height = 3.7
32 attr floor_height = 3.3
33 attr Tile_Width = 5
34 attr Tile_Side_Width = 1.3
35 attr Window_Base_Height = 0.9
36 attr Window_Height = 1.8
37 attr HOEHEGEB = rand(1, 20) #Total height of the building
38 attr Wall_Texture = listRandom(randomWallTexture)
39 attr Window_Texture = fileRandom(randomWindowTexture)
40 attr Roof_Texture = fileRandom(randomRoofTexture)
41 attr Roof_Angle = atan(Roof_Ht/(0.5*scope.sx)) #calculation of roof angle
42
43 #####
44 # RULES
45 #####
46 @StartRule
47 @Location(1079,-6)
48 Lot-->
49 extrude(world.y, HOEHEGEB) # extrude the lot to building total height
50 Building
51
52 @Location(1259,-8)
53 Building--> #building is split into bottom, front facade, roof and side facade
54 comp(f) {bottom : X.
55         |2 : FrontFacade
56         |top : Roof

```



```

57         |all : SideFacade}
58
59 FrontFacade -->
60 case HOEHEGEB < 4 :
61     setupProjection(0, scope.xy, 1.5, 1, 1) # setup 1.5m x 1m texture tiles
62 along scopes xy plane (and distortion in z)
63     setupProjection(2, scope.xy, scope.sx, scope.sy)
64     texture (wooden_facade) # buildings with total height less than 4
65 meters have wooden wall texture
66     split(y){ groundfloor_height : GroundFloor # the front facade is
67 subdivided into one front groundfloor and upper floors
68         | {~floor_height : Floor}* }
69     else:
70     setupProjection(0, scope.xy, 1.5, 1, 1) # setup 1.5m x 1m texture tiles
71 along scopes xy plane (and distortion in z)
72     setupProjection(2, scope.xy, scope.sx, scope.sy)
73     split(y){ groundfloor_height : GroundFloor
74         | {~floor_height : Floor}* }
75
76 @Location(1646,288)
77 SideFacade -->
78 case HOEHEGEB < 4 :
79     setupProjection(0, scope.xy, 1.5, 1, 1) # setup 1.5m x 1m texture tiles
80 along scopes xy plane (and distortion in z)
81     setupProjection(2, scope.xy, scope.sx, scope.sy)
82     texture (wooden_facade) # buildings with total height less than 4
83 meters have wooden wall texture
84     split(y){ groundfloor_height : Floor # a side facade is subdivided
85 into one bottom floor and upper floors.
86         | {~floor_height : Floor}* }
87 else:
88     setupProjection(0, scope.xy, 1.5, 1, 1) # setup 1.5m x 1m texture tiles
89 along scopes xy plane (and distortion in z)
90     setupProjection(2, scope.xy, scope.sx, scope.sy)
91     split(y){ groundfloor_height : Floor
92         | {~floor_height : Floor}* }
93
94 @Location(2000,257)
95 Floor -->
96     split(x){ 1.5 : Wall
97         | { ~Tile_Width : Tile }*
98         | 1.5 : Wall }
99
100 GroundFloor -->
101     split(x){ 1.5 : Wall
102         | { ~Tile_Width : GroundFloorTile }*
103         | ~Tile_Width : EntranceTile
104         | { ~Tile_Width : GroundFloorTile }*
105         | 1.5 : Wall }
106
107 @Location(2226,26)
108 GroundFloorTile -->
109     split(x){ ~Tile_Side_Width : Wall
110         | 2 : split(y){ ~Window_Base_Height: Wall | ~Window_Height:
111 Window | ~1: Wall }
112         | ~Tile_Side_Width : Wall }
113
114 @Location(2164,297)

```



```

115 Tile -->
116     split(x){ ~Tile_Side_Width : Wall
117         | 2 : split(y){ ~Window_Base_Height: Wall | ~Window_Height:
118 Window | ~1: Wall }
119         | ~Tile_Side_Width : Wall }
120
121 @Location(2221,117)
122 EntranceTile -->
123     split(x){ ~1 : SolidWall
124         | 2 : split(y){ 2.5: Door | ~1: SolidWall }
125         | ~1 : SolidWall }
126
127 @Location(2672,164)
128 Window -->
129 case touches () : Wall #avoids placing windows at the intersections of buildings
130 else :
131     s('1','1,0.4)
132     t(0,0,-0.4) # windows and their textures are set back 0.4 meters into the
133 facade
134     texture(Window_Texture)
135     i("builtin:cube")
136
137 @Location(2674,284)
138 Door -->
139 case touches () : Wall #avoids placing doors at the intersections of buildings
140 else:
141     s('1','1,0.1)
142     t(0,0,-0.5) # doors and their textures are set back 0.5 meters into
143 the facade
144     texture(randomDoorTexture)
145     i("builtin:cube")
146
147 @Location(3110,-72)
148 Wall -->
149     case HOEHEGEB < 4 :
150         projectUV(0) projectUV(2)
151         texture (wooden_facade) # buildings with total height less than 4 meters
152 have wooden wall texture
153     else:
154         projectUV(0) projectUV(2)
155         texture (Wall_Texture)
156
157 @Location(2657,408)
158 SolidWall -->
159     case HOEHEGEB < 4 :
160         texture (wooden_facade)
161         s('1','1,0.4)
162         t(0,0,-0.4)
163         i("builtin:cube")
164         projectUV(0) projectUV(2)
165     else:
166         texture (Wall_Texture)
167         s('1','1,0.4)
168         t(0,0,-0.4)
169         i("builtin:cube")
170         projectUV(0) projectUV(2)
171
172 #####Roof Generation####

```



```

173 @Location(1698,495)
174 Roof -->
175     case DACHFORM == 3100 :GableRoof
176     case DACHFORM == 1000 :FlatRoof
177     else :GableRoof
178 #gable roof
179 @Location(2190,484)
180 GableRoof -->
181     case HOEHEGEB < 4 : roofGable(Roof_Angle,0.2,0.2,false,ridgeOrient)
182 #roof slope, eave overhang, rake overhang
183 #Buildings with total height less than 4m, have 0.2m overhangs
184 RoofMass(false)
185 else:
186 roofGable(Roof_Angle,0.8,0.8,false,ridgeOrient) #Buildings which are 4m and
187 higher, have 0.8m overhangs
188 RoofMass(false)
189
190 @Location(2480,487)
191 RoofMass(removeBottomAndTop) -->
192     case removeBottomAndTop:
193         comp(f){ horizontal: NIL | vertical: Wall | all: RoofPlane }
194         RoofMassScale
195     else: # remove only the bottom face
196         comp(f){ bottom: NIL | vertical: Wall | all: RoofPlane }
197
198 # roof volume
199 RoofMassScale -->
200     s('1','Roof_Ht','1')
201     RoofMass(false)
202
203 #flat roof
204 FlatRoof -->
205     case Roof_Ht > 0:
206         RoofPlane offset(-0.4,border) extrude(Roof_Ht) RoofMass(false)
207     else:
208         RoofPlane
209
210 # roof surfaces
211 RoofFacade -->
212     case HOEHEGEB < 4 :
213         setupProjection (0, scope.xy, '1, '1)
214         projectUV (0)
215         texture (roof_texture2)
216 else:setupProjection (0, scope.xy, '1, '1)
217         projectUV (0)
218         texture (Roof_Texture)
219
220 @Location(2672,781)
221 RoofPlane -->
222     case HOEHEGEB < 4 :
223         t(0,0.01,0.01)
224         setupProjection (0, scope.xy, '1, '1)
225         projectUV (0)
226         texture (roof_texture2)
227 else:t(0,0.01,0.01)
228         setupProjection (0, scope.xy, '1, '1)
229         projectUV (0)
230         texture (Roof_Texture)

```