University of Salzburg

Interfaculty Department of Geoinformatics Z_GIS

SEMINAR PAPER

Procedural building modeling. A new approach to estimate heat energy demands of buildings?

SEMINAR

Analysis & Modeling

INSTRUCTORS

Prof. Josef Strobl

STUDENT

Marius Servais (11928363)

Luxembourg, 31.07.2020

Table of Content

1. Introduction ……...… pp. 2-3.
2. Procedural Modeling ……..… pp. 4-8.
3. Estimation of (Net) Heating Energy Demand……..... pp. 9-11.
4. Methodology……..... p. 12.
   4.1. Data acquisition and pre-processing……..... pp. 12-14.
   4.2. Procedural Modeling and Heat Estimation of the Weststadt……. pp. 15-17.
5. Discussion…….. pp. 18-19.
6. Conclusion……. p. 20.
7. References…….. pp. 21-22
8. Appendix……..... pp. 23-28

Table of Figures

Figure 1: The scope of a shape……..……..……...…p.4.

Figure 2: Scenegraph showing the hierarchical structure of CGA shape……...……p.5.

Figure 3: Subdivision split with absolute and relative input values……....…...p.6.

Figure 4: Component split in faces based on object selectors……...….....p.6.

Figure 5: Insert function for integrating complex shapes (sculpture or window) and texture….....p.7.

Figure 6: Stochastic expression for choosing different successor by chance……...…..p.7.

Figure 7: Conditional rule brings control to CGA shape……...…….p.7.

Figure 8: Occlusion rule enables checking for different types of intersection……..….p.8.

Figure 9: Overview of the heating losses and gains of buildings……...p.9.

Figure 10: Net heat energy demand formula (simplified) based on the TABULA approach……....p.10.

Figure 11: Flowdiagram of the pre-processing workflow……..p.13.

Figure 12: 3D Building model of the Weststadt showing a building with two height levels…….....p.13.

Figure 13: Left: building footprint (ALKIS); Center: 'GroundSurface' (CityGML) showing the building structure at the ground level; Right: 'RoofSurface' (CityGML) showing the roof structure of a 3D building model……..p.14.

Figure 14: The 4 major transformation steps of the Weststadt procedural model……..…p.16.

Figure 15: 3D building model of the Weststadt with net heat energy per living space (kWh/a)…...p.17.

Figure 16: (Left) Bottom view on 3D building with two doors, which are adjacent with a neighbouring wall. (Right) Front view on 3D building with four doors……..….p.18.

1. INTRODUCTION

The calculation of heat energy in the building sector is essential for energy and environmental planners. Knowledge about the heating energy demand is needed for planning energy related infrastructure (ex. long-distance heating or decentralized heat stations) and to ensure the energy supply. Additionally the monitoring of heating energy is, as one of the major CO^2 emission contributors, an essential aspect of climate and environmental policies. Studies are monitoring and evaluating the impact of climate measures such as the modification of existing infrastructure, the composition of energy resources or the refurbishment of buildings and heating systems (Kaden & Kolbe 2013).

Heating demand of buildings is often based on estimation due to the retention of consumption values by the energy companies for reasons of privacy regulations or economic interest. Heating is a complex process and depends on several factors: (1) The geometry and the envelope of a building with its transmission heat losses, (2) the building utilities (heating system or boiler) with its energy transformation losses, (3) climate, (4) user behaviour and (5) building usage (Kaden & Kolbe 2013). Approaches to estimate heat energy do not only have to deal with the complex composition of heating, but its design is also influenced by the data availability and the spatial scale of the case study. Studies based on a Building Information Modeling (BIM) technology, mostly limited to a small scale, are covering nearly all parameters (Sola et al. 2018). But a sophisticated collection of all these parameters is impossible to achieve in larger regions or whole countries. Those studies are therefore only focusing on some parameters. TABULA is one approach that has been developed to estimate the heating demand for large regions, whereas the geometry and the user behaviour are based on assumptions (Diefenbach et al. 2015). It uses a reference building typology (representing average geometrical and isolation characteristics of a building type) and national building inventory data consisting of construction year, building type and living space to estimate the heating demand.

A new technique enabling large scale heat energy calculation at the level of an individual building is made possible with laser scanning data. Geometrical information for each building derives from a digital surface model with the help of building footprints and is combined with thematic information from the cadastre (age, building usage, numbers of storeys, etc.). The so called spatio-semantic model, predominantly stored in CityGML, can be used to calculate the heated volume and the surface areas of the building components. In combination with the estimated heat transfer coefficients taken from the construction year one can estimate the heat energy demand. This approach is in comparison to prior ones more accurate in calculating heat energy. However, a major problem of those 3D building models is the low ‘Level of Detail’ (mainly LoD2). The models consist of an outer envelope including walls and a roof. More detailed elements such as windows and doors or the interior of a building are missing (Kaden & Kolbe 2013; Sola et al. 2018; Strzalka et al. 2011). Even though they ‘have a significant influence on the transmission heat loss of a building, and thereby on the estimated heating energy demands’ (Kaden & Kolbe 2016, 164).

On these grounds, a new complementary approach called procedural modeling is tested to generate a more complex building model. Shape grammar based procedural modeling uses a set of rules to describe the architectural structure of a building and is able to create large dynamic spatio-semantic models in an efficient way. In an iterative process a simple shape is converted into a more complex one. The complexity of such models can reach the fourth ‘Level of Detail’ (Müller et al. 2006). In this paper, Weststadt, a quarter located in Bonn (North-Rhine

Westphalia) serves as exemplified case study. On the basis of a digital surface model, a digital terrain model, a building model of LoD2 and a cadastre register a 3D building model of LoD3 is generated, where important geometrical parameters of the building's envelope are derived from and used as input for a heat energy simulation. The focus of this paper lies on the method to generate a complex 3D model with CGA shape, a procedural modeling language, and on the spatio-semantic model as the potential input for a heat energy simulation. The applied heat estimation formula taken from the TABULA approach is therefore simplified in several aspects apart from the geometry factor. Missing input such as the construction years of the building, the structure of the facade or climate data are syntactically generated or based on assumptions. Further, missing parameters like the transmission coefficients of non-residential buildings are left out. Consequently, the energy simulation is only applied to residential buildings of the Weststadt and gives back theoretical net energy values.

The main of objectives the case study are:

- To create a building model as accurate as possible of the Weststadt with the procedural modeling language CGA shape grammar (computer-generated architecture) and the respective software CityEngine.
- To calculate the net heat energy consumption of residential buildings based on geometrical building dimensions (surface area & volume) and building characteristics (construction year & building function).

Furthermore, the study aims for a complete integration of the heat energy formula within the CGA shape.

In the first part of the paper, I will outline shape grammar based procedural modeling, followed by a description of the net heat energy calculation method. In the second part, I will present the case study of Weststadt with the 3D building model and the net heat energy simulation, before discussing in the last chapter the practicability of procedural modeling as a new heat estimation tool.

2. PROCEDURAL MODELING

Procedural modeling is a collection of techniques used in computer graphics to construct a virtual 3D model from a real or imaginary object based on procedures. In contrast to manual modeling where a list of primitives (vertices, lines, surfaces, etc.) constructs a model part-by-part and where the goal is to simply match the visual appearance of an object, the so called procedures (set of rules) are used to describe semantically the structural logic of an element and the relationship between elements. By applying this set of rules or algorithm to an initial input (ex.: geometry), the creation engine will rewrite the initial object in an iterative manner until the final output is produced. Procedures are not describing a single object, but instead are defining a class. The parametrization of the algorithm allows to generate multiple instances of a class where small changes in specifications can lead to a variety of different forms and appearances. Procedural modeling is an efficient approach to generate large and complex scenes in an automated manner and in reducing storage (Chen 2008; Heagler et al. 2009; Hidalgo et al. 2008).

Procedural modeling supports creating many objects of different types such as plants, buildings, streets, clouds, fire, terrain models and even crowds of people. As a result, procedural modeling incorporates a broad spectrum of modeling techniques: Fractal, particle, physically-based, image-based or grammar-based model systems. Historically, most techniques originate from a grammar based technique called L-systems introduced by Lindenmayer to model cellular interactions of plants. Over time, L-systems were modified and extended with arithmetic, boolean, stochastic expressions as well as context-sensitive rules to support interaction among objects (Chen 2008; Hidalgo et al. 2008). In contrast to the growth modeling of plants with L-systems, the underlying model technique of this paper is based on a technique describing buildings and other architecture types. The so called 'shape grammar-based model' is designed to transform simple shapes into more complex ones, where each transformation step increases the details of the shape. For example, a simple cube is converted to a building containing a roof, windows and doors. This concept of shape replacement was introduced first by Stiny in the 1970s and extended in the early 2000s by Müller et al. (2006) to make architectural modeling more practicable. Müller et al. introduced a shape grammar called 'CGA shape' (Computer Graphics Architecture) capable of integrating attributes of shapes as parameters that allow a greater variety of modeling (Grêt-Regamey et al. 2013; Krecklau et al. 2010; Müller et al. 2006).

The foundation of 'CGA shape' is built upon shapes and a production process. Shapes are geometries containing attributes (like color or the year of construction) and are identifiable by a symbol. Geometries, as shown in figure 1, are described by their position *P*, a coordinate system, their orthogonal *X, Y, Z* vectors and a size vector *S*. The vectors for orientations and the size are defining a bounding box or respectively the scope of a shape. The symbol gives the geometry an identifier in string format. It can be a simple letter or a term like 'wall'. Symbols are either terminal or non-terminal referring to the state of the shape. Terminal symbols have received their final transformation and can't be rewritten anymore. The starting shape or symbol that is used to initiate the transformation process is called axiom (non-terminal symbol). The

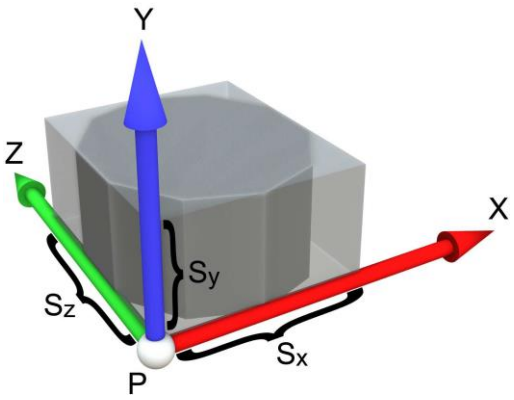


Figure 1: The scope of a shape (Müller et al. 2006).

production process describes an iterative and sequential process where shapes are transformed to more complex shapes on the basis of different rules. In the first iteration a production rule is applied to the initial set of shapes (axioms) and replaces them with one or more successor shapes. The initial shapes are not deleted but marked as terminal. In the second iteration the successor shapes of the first iteration are now the predecessor shapes and are being rewritten with another rule. The production process is repeated until all non-terminal shapes are replaced and set to terminal. During this production process a hierarchical structure of parents and instances is built, called scenegraph, describing the whole evolution of the initial shape (Müller et al. 2006; see figure 2).

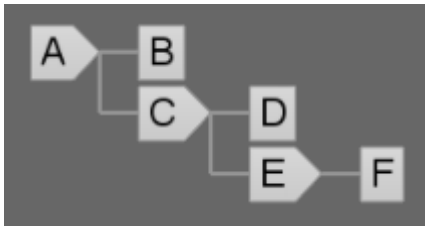


Figure 2: Scenegraph showing the hierarchical structure of CGA shape (ArcGIS CityEngine 2020).

A production rule is defined in the following form:

Rule:

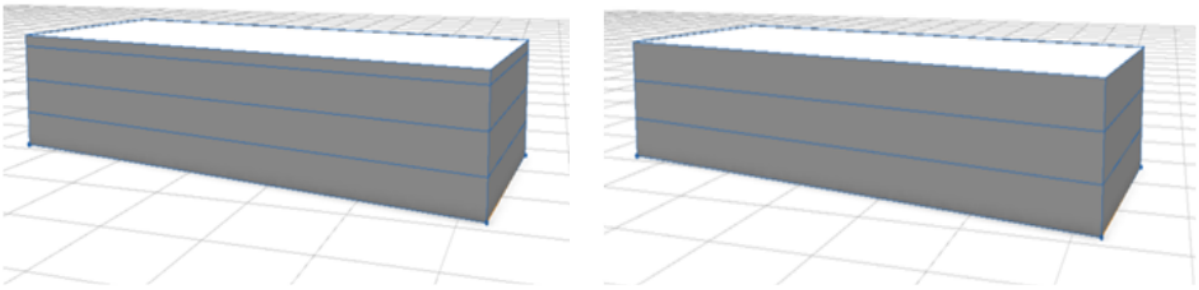
id: predecessor: condition → successor: rules/operations

A rule consists of an *id* that refers to an identifier of a rule and the already known structure of *predecessor* and *successor*. A predecessor can be conditioned by a logical expression before being executed. For example, this rule:

Building(height): height > 9 → color(red) Red Building

where a shape with the symbol 'Building' must be greater than nine to create a successor shape in red color (Müller et al. 2006).

On the successor side we locate the rewriting rules or operations for the predecessor. The main rules/operations incorporated in CGA shape are boolean, stochastic and context-sensitive rules as well as shape operations. Further, CGA shape allows queries for geometric information as the surface area, the volume or the orientation of a shape. The basic shape operations are about relocating (translating), rotating and resizing objects. Also, 2D objects can be extruded, meaning that a surface receives a height information and becomes a volumetric shape or mass model (ArcGIS CityEngine 2020). The two most important rules within shape operations are subdivision split and component split. The subdivision split is able to split an object along one or multiple axis. An example, shown in figure 3, splits a cube of height 10 vertically into three floors. The dimension of the single elements, in this case the height of the floors, can be in absolute or relative values. On the left side we see an absolute split resulting in three floors and a ledge. On the right there is a relative split creating 3 floors of equal height. Further it is possible to achieve the same outcome without precising the number of floors. The repeat function divides a length of an object in as many elements as there is space by just indicating an absolute or relative length of one single element (see figure 3).



Left: Floor height as absolute value
Without repetition:

```
Cube -->
split(y) {3:Floor |3:Floor|3:Floor}
```

Or with repetition:

```
Cube --> split(y) {{3:Floor}*}
```

Right: Floor height as relative value
Without repetition:

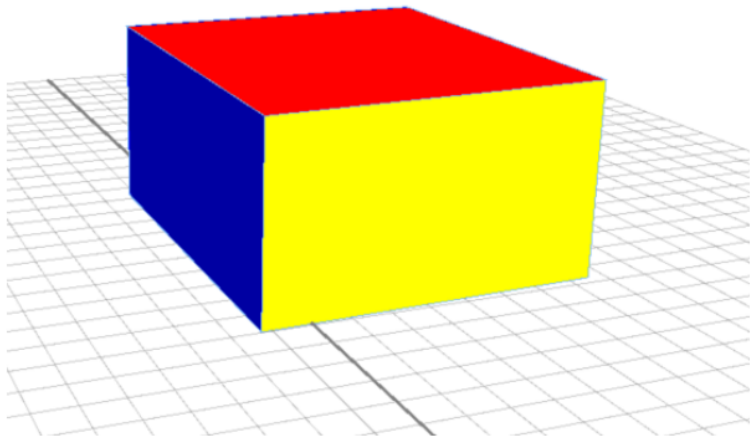
```
Cube -->
split(y) {~1:Floor|~1:Floor|~1:Floor}
```

Or with repetition:

```
Cube --> split(y) {{~3:Floor}*}
```

Figure 3: Subdivision split with absolute and relative input values.

The component split disassembles an object into its components. This split function reduces a shape into shapes of lesser dimensions. A three-dimensional shape can be split into its faces, edges and vertices. So called 'selectors' help to determine the components by its orientation. This can be in relation to the object itself (front, back, left, right, side, top, bottom) or to the world coordinate system (south, north, west, east, up, down) or to a street network (street.front, street.back, street.left, street.right, street.side) (ArcGIS CityEngine 2020). Figure 4 shows an example, where a cube is spilt into its face components based on object selectors.



```
Cube -->
comp(f){top: color (red)Roof | front: color(yellow)Frontfacade
| side: color(blue)Sides| bottom: Groundfloor}
```

Figure 4: Component split in faces based on object selectors.

Another important function in order to achieve complex shapes is to insert predefined and sophisticated geometry meshes like windows, plants or sculptures. Additionally, it is possible to add texture to a shape (ArcGIS CityEngine 2020; see figure 5).

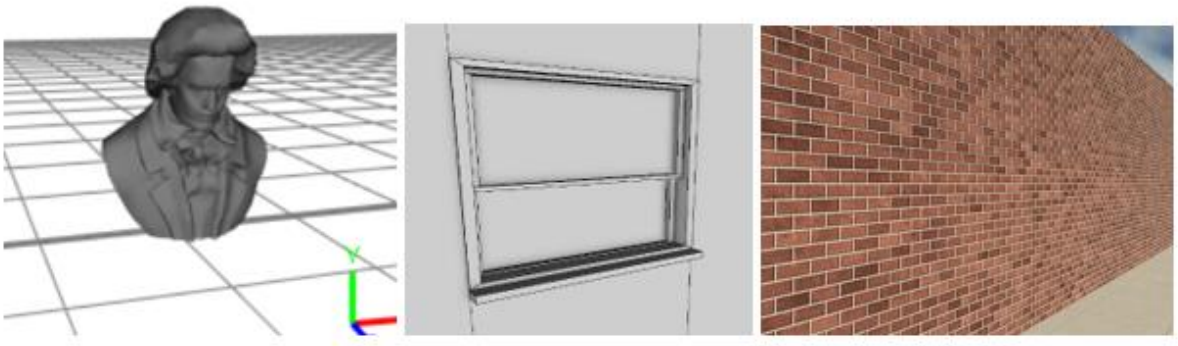


Figure 5: Insert function for integrating complex shapes (sculpture or window) and texture (ArcGIS CityEngine 2020).

The integration of boolean, stochastic, conditional expressions brings randomness and control to CGA shape. A randomness example can be seen in figure 6. A stochastic rule determines probabilities for the choice between different successor types. Figure 7 illustrates shape control via arithmetic and conditional operators (Müller et al. 2006).

```
Lot -->
        30%  : Lot("residential")
        20%  : Lot("retail")
        else : Lot("industrial")
```

Figure 6: Stochastic expression for choosing different successor by chance (ArcGIS CityEngine 2020).

```
Footprint(type) -->
      case type == "residential" : extrude(10)
      case geometry.area/2 < 200 : extrude(30)
      else : NIL
```

Figure 7: Conditional rule brings control to CGA shape (ArcGIS CityEngine 2020).

A last essential rule is the context-sensitive occlusion rule. It tests for intersection between shapes and enables therefore interrelationships between objects and the environment. The rule tests for an inside intersection, an overlap or a touching of entities (ArcGIS CityEngine 2020; see figure 8).

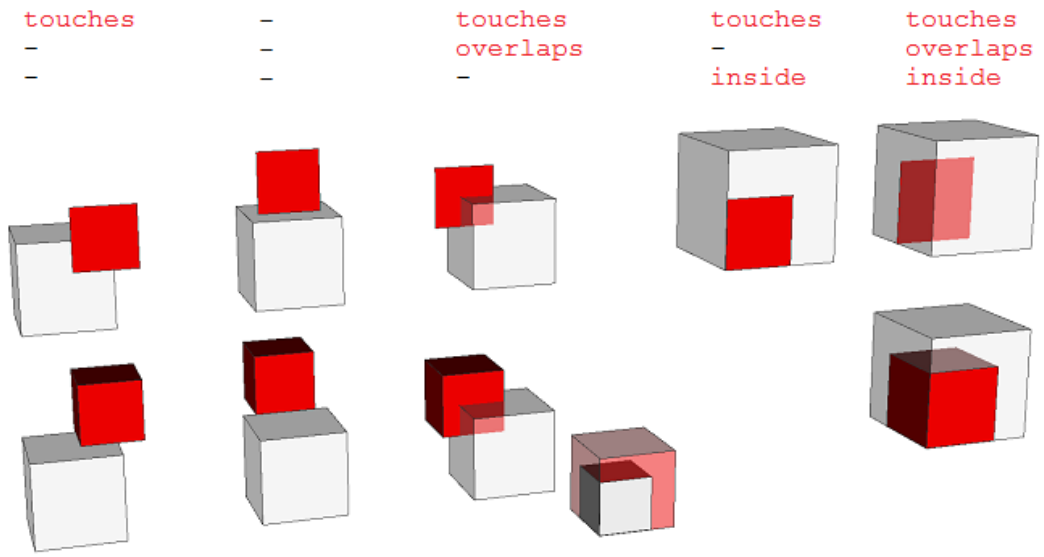


Figure 8: Occlusion rule enables checking for different types of intersection (touches, overlap or inside) (ArcGIS CityEngine 2020).

3. ESTIMATION OF (NET) HEATING ENERGY DEMAND

Estimating heating energy demands of buildings may look trivial at first sight. Heat energy demand is the subtraction of heat gains from heat losses. At second sight though, it becomes clear that this isn't the case. Heat losses are due to heat flows transcending the building's fabric, ventilation by opening doors and windows and energy conversion losses of the heating system or boiler (see figure 9). The heat loss through the hull of a building depends on the geometry of a building, the surface areas of the different components (floor, wall, roof, window & door) and the transmission coefficient of these components. The transmission coefficient (U-value) describes the heat conductibility and can be estimated on the base of a building's age. Buildings with a good isolation are having low U-values and are energy efficient. The geometry is relevant insofar as the ratio between outer hull surface and the volume are affecting the heat loss. Compact buildings close to a cube are more efficient than longish buildings. Further, the impacts of adjacent buildings must be considered. Shared surfaces between buildings are not contributing to the transcending heat flows to the outer air and must be neglected. Sources for heat gains apart of the heating system come from solar radiation penetrating the windows and from the heat emissions of residents, electronical devices and hot water use. The share of these internal gains is small. The amount of energy needed to heat a building is not only defined by the geometrical and technical characteristics of a building, but consists also of the building usage, the heating duration, the climatic conditions (soft or hard winter) and the personal preference of the room temperature (Kaden & Kolbe 2013; OpenLearn 2020; Servais 2019).

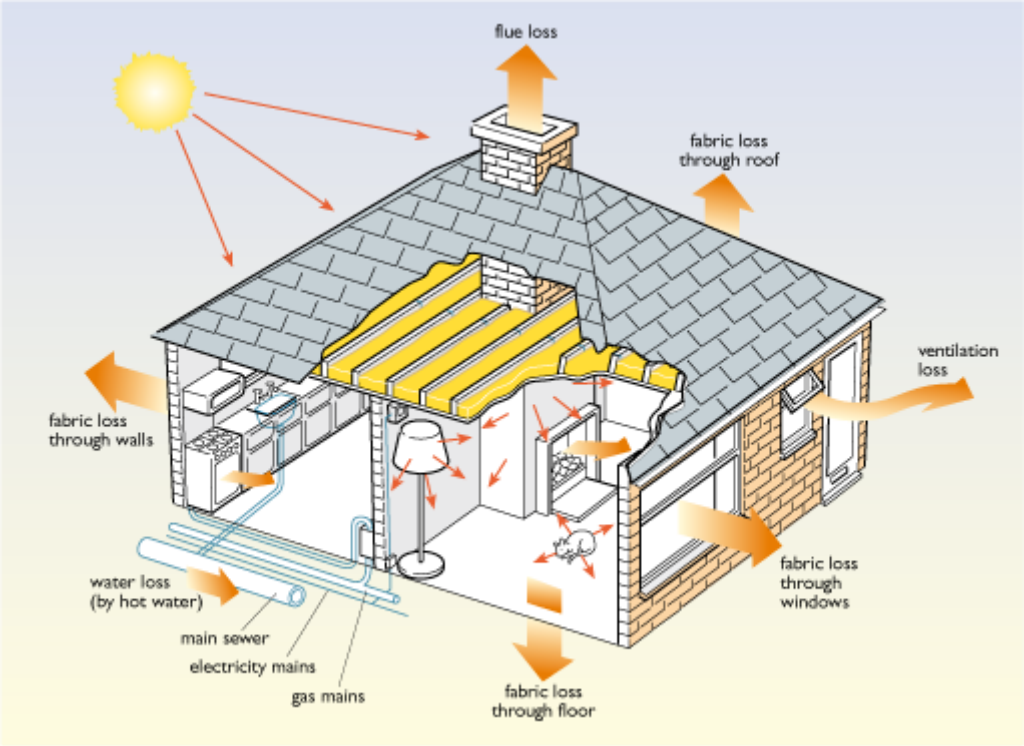


Figure 9: Overview of the heating losses and gains of buildings (OpenLearn 2020).

The following calculation method for heat energy demand of buildings is a simplified method of the TABULA [1]approach and is limited to the calculation of net energy (TABULA 2013). The output of the formula describes the energy needed to heat up a building to a certain room temperature without taking energy conversion losses into account. The method emphazises the

[1] The TABULA calculation formula of heat energy demand of buildings is based on EN ISO 13790/EN 15316.

building specific losses (geometry, envelope) and gains. Transmission coefficients, user preferences (ventilation rate, room temperature), physical parameters (heat capacity of air, solar radiation including material penalties) or other correction factors are assumed from the TABULA approach (see TABULA 2013).

Net Heat Energy Formula (Simplified):

Heat transfer coefficient by transmission (H_{tr} (W/K)):
U-Value * **Area of Roof** = H_{tr} Roof (W/K)
U-Value * **Area of Wall** = H_{tr} Wall (W/K)
0.5 * U-Value * **Area of Floor** = H_{tr} Floor (W/K)
U-Value * **Area of Window** = H_{tr} Window (W/K)
U-Value * **Area of Door** = H_{tr} Door (W/K)
U-Value * **Area of Envelope** = H_{tr} Bridge (W/K)

$\sum$ H_{tr} Building Elements

Heat transfer coefficient by ventilation (H_{ve} (W/K)):
Heat Capacity Air * Air Change Rate * **Building Volume** = H_{ve} (W/K)

Total heat transfer (Qht (kWh/a)):
(Internal Temperature - External Temperature) * Heating Days * (H_{tr} + H_{ve}) = Q_{ht} (kWh/a)

Solar heat load during heating season (Q_{sol} (kWh/a):
Reduction Factors * Solar Radiation * **Window Area North** = Q_{sol} North (kWh/a)
Reduction Factors * Solar Radiation * **Window Area East** = Q_{sol} East (kWh/a)
Reduction Factors * Solar Radiation * **Window Area South** = Q_{sol} South (kWh/a)
Reduction Factors * Solar Radiation * **Window Area West** = Q_{sol} West (kWh/a)

$\sum$ Q_{sol} Window Elements

Internal heat sources (Q_{int} (kWh/a)):
Internal Heating Factor * Heating Days * **Living Space** = Q_{int} (kWh/a)

ENERGY NEED (Net Energy) for Heating (Q_H (kWh/a)):
Q_{ht} - Gain Factor *(Q_{sol} + Q_{int}) = Q_H (kWh/a)

NET ENERGY per AREA (Q_{Ha} (kWh/a)):
Q_H / Living Space = Q_{Ha} (kWh/a)

Figure 10: Net heat energy demand formula (simplified) based on the TABULA approach (TABULA 2013).

The first section of the formula defines the heat transfer coefficient by transmission, which is the sum of all hull components' surface areas multiplied with their transmission coefficient. For the floor element an additional reduction factor of 0.5 is implemented, because the transmission losses through the soil are significantly less than through air. Further, a bridge transmission loss is integrated describing the transition heat losses between components and can be calculated on the basis of the whole envelope. The second section specifies the heat transfer coefficient by ventilation, which is a multiplication of the ventilation rate with the building's volume. Both coefficients can be defined as space heating energy flow rate in watts. For calculating the total heat loss, the duration and the intensity of the heating period needs to be determined. The according degree-day method adds up the daily average temperature difference between the

interior and the outside air when the average of the outside temperature is below the heating threshold temperature to an annual value (Kd/a). The fourth and fifth sections specify the solar and internal heat gains. The solar heating requires the insolation and the surface area of the window by orientation. The gain depends on the orientation. Internal gains are defined by the heating period, the living space and a predefined heating factor. In a final step, the net heating energy demand can be calculated by subtracting the gains from the losses. The energy demand is mostly referred to as energy per reference area (living space) in kWh/a (TABULA 2013).

4. METHODOLOGY

The practicability of procedural modeling as a new heat energy estimation approach is tested on the basis of CGA shape and the underlying CityEngine software. The focus lies on the creation of a complex and high in detail building model (LoD 3), where important geometrical parameters of the building's envelope are derived from and used as input for a heat energy simulation. Based on a good availability of the required data for creating such a 3D scene, Weststadt, a predominantly residential quarter of the city of Bonn (North-Rhine Westphalia), was chosen to be the area for the case study. Height information in form of a digital surface and terrain model, building footprints as well as building information (building function & roof shapes) are freely accessible and can be used to reconstruct the quarter as accurate as possible. However, information about the structure of facade's element is not available and it is thus based on a very simple structure describing the size and the position of windows and doors. Also, the year of construction is missing and is therefore randomly generated for each building. The applied heat estimation formula is simplified in several aspects apart from the geometry factor (see chapter 3). Several correction parameters or energy conversion losses through heating systems are not being considered. The U-values are derived from the TABULA approach and are classified in construction periods (see appendix 1). Assumptions about the duration and intensity of the heating period are made and simulate an average winter. U-values and the ventilation rate are reference values from residential buildings. Consequently, the energy simulation is only applied to residential buildings of the Weststadt and gives net energy values per area back. Those values are not representing real estimations but should rather be viewed as notional energy values!

4.1 DATA ACQUISITION AND PRE-PROCESSING

The first step requires a pre-processing of the data before modeling the Weststadt as a scene. All required data is acquired via the OpenGeoData portal of the state of North-Rhine Westphalia. The datasets consist of:

- a building inventory dataset of Bonn including building footprints and an attribute describing the building function, which can later be used for classifying the buildings in residential and non-residential[2]
- a laser scanning dataset as digital surface model in tiles[3]
- a digital terrain model in a .xyz format and in tiles[4]
- a 3D model of buildings in CityGML format and in tiles[5]
- an administrative boundary shape of the Weststadt[6]

[2] https://www.opengeodata.nrw.de/produkte/geobasis/lk/bda_oe_xml/
[3] https://www.opengeodata.nrw.de/produkte/geobasis/hm/dgm1_xyz/dgm1_xyz/
[4] https://www.opengeodata.nrw.de/produkte/geobasis/hm/dgm1_xyz/dgm1_xyz/
[5] https://www.opengeodata.nrw.de/produkte/geobasis/3dg/lod2_gml/
[6] https://www.opengeodata.nrw.de/produkte/geobasis/lm/basis-dlm/

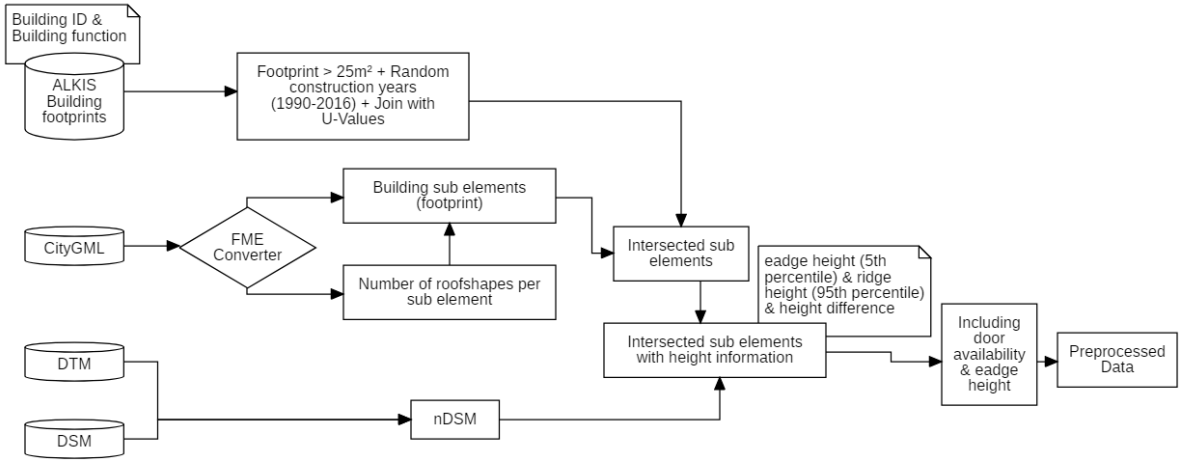


Figure 11: Flowdiagram of the pre-processing workflow.

The workflow of the pre-processing is described in figure 11 and is realized with GIS tools (ArcGIS Pro, QGIS & R-Studio). After merging all tiles together, we intersect all datasets with the boundary shape of the Weststadt. In a next step, a subset of the building inventory dataset (ALKIS) is defined with an area of the polygons greater than 25m² for getting rid of unheated shapes such as balconies or garages. Also, random construction years between 1900 and 2016 are assigned to the buildings and are joined with the respective U-values for each building element (floor, wall, roof, window, door & bridge) derived from the TABULA approach[7] (see appendix 1). Further, the laser scanning data and digital terrain model in .xyz format are processed to a digital surface respectively digital terrain model as raster with a one meter resolution. In order to make building height information extractable, the terrain model is subtracted from the surface model resulting in a normalized digital surface model (nDSM). By overlaying this newly created elevation model with the building footprints, I detected for some buildings different height levels of the roofs within a polygon. This difference in height was also reproduced in the 3D model as we can see in figure 12. Furthermore, the 3D model allowed me to identify different types of roofs (flat roof, gable roof, hip roof, pyramid roof or more complex roof structures). For producing a spatio-semantic model as accurate as possible, I decided to extract the subdivision of buildings on the basis of height differences and the number of roof shapes per sub element with the help of a FME Converter. The number of roof shapes is used later in the procedural modeling to define the roof type[8]. The FME workspace

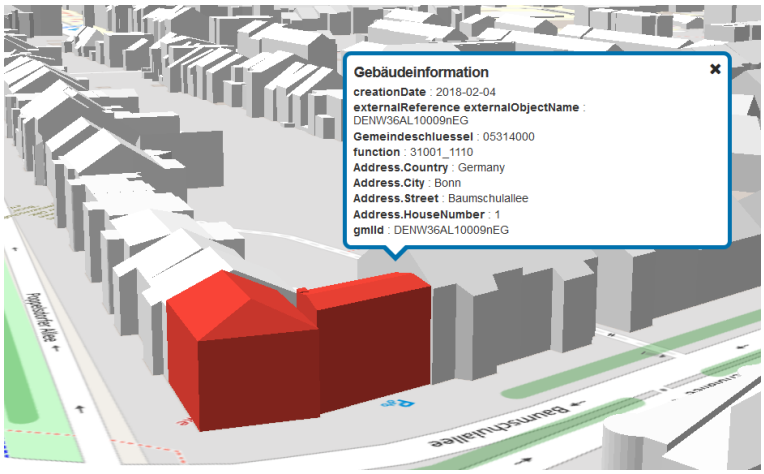


Figure 12: 3D Building model of the Weststadt showing a building with two height levels.

[7] The U-values for each component are average values of the building types SFH, TH, MFH, AB.
[8] The FME Converter allows to extract the roof type information directly from the 3D model, but due to my limited knowledge with the Converter this wasn't possible.

template ‘CityGML to Esri File Geodatabase’ processes 3D models into 2D layers[9]. The 2D output layers ‘RoofSurface’ and ‘GroundSurface’ shown in figure 13 represent the structure of the buildings on the ground and the roof level. After counting the roof shapes per building sub element, the ‘GroundSurface’ Layer is intersected with the ALKIS layer to eliminate the unheated polygons and to import all attributes (building id, building function, construction year, U-values) in the new basis layer. A next step is to extract the height information from the nDSM. To avoide outliers (ex.: chimney) the eave is defined by the 5th percentile and the roof ridge by the 95th percentile. Further, a height difference between eave and ridge is calculated that is needed for the roof height. A problem that occurs with the building’s subdivision as basis layer in perspective to the later procedural modeling is that not all subdivisions possess a door. To overcome this situation, another attribute is defined describing a door availability. With the help of a for loop, only one sub element of each house receives a door. Finally, a last cleaning of the data is done by eliminating all buildings smaller than 3 meters.

Figure 13: Left: building footprint (ALKIS); Center: ‘GroundSurface’ (CityGML) showing the building structure at the ground level; Right: ‘RoofSurface’ (CityGML) showing the roof structure of a 3D building model.

[9] https://hub.safe.com/publishers/vcs/templates/citygml-to-esri-file-geodatabase

4.2 PROCEDURAL MODELING AND HEAT ESTIMATION OF THE WESTSTADT

After the dataset is prepared with all required information, the building's subdivision layer is imported as initial shape to the CityEngine software. The workflow of the procedural modeling with the set of rules can be seen in appendix 2 & 3. The procedural modeling design differentiates between residential buildings, where we can simulate the heat energy, and non-residential buildings, where energy simulation isn't possible. Both types of buildings will be modelled, but in different 'Level of Details'. Due to lack of information on the facade structure, I decided that residential buildings' facade will be generated by a simple common logic. Non-residential buildings with a very high variation in facade structure (ex.: warehouse vs. church) will only be LoD2.

The first rule of the procedures is thus a condition rule, where this partition in a simple non-residential shape and in a complex residential shape is done on the basis of the building's function[10]. Buildings with mixed function are classified as residential. Both types of shapes are extruded by the height of the eave to a basic cube and subsequently split into their components (bottom, front, side, top; see figure 14 a). In the next step, the flat surface of the roof is updated with predefined roof shapes. A further condition rule based on the number of roof elements is used to generate different forms of roof shapes[11] (see figure 14 b):

```
Roof -->
      case roofpoly <2  : extrude(0)flat_roof
      case roofpoly ==2 : roofGable(roof_H)gable_roof
      case roofpoly > 2: 80%: roofHip(byHeight, roof_H)
            else: roofPyramid(byHeight, roof_H)hip_pyr_roof
      else: NIL
```

(roofpoly = number of roof elements; roof_H = roof height)

As illustrated in the code, I used a stochastic rule for increasing the variety of roof shapes for the roofs having more than two roof elements. The more traditional hip roof is modelled by a chance of 80. The simpler pyramid form appears only in 20 percent of cases. For the simple buildings, the shape modeling ends at this stage. The front side ('Frontfacade') and the other sides ('Sidefacade') of the complex shape additionally receive a facade structure. A vertical split divides the facade into a ground floor of 4 meters height and in repetitive floors of 3 meters height (relative split; see figure 14 c). For each floor level another horizontal and vertical split defines the space of the windows. Furthermore, integrated occlusion rules are testing for an intersection ('touche') and in the absence windows are generated. In the case of the first storey of a front facade a condition is assessing if a sub element has a positive door availability. If this is the case, a door is generated (see figure 14 d). At each stage, where a relevant shape element for the energy simulation is processed a query is asking for the surface area or volume of an object. Shared surfaces between buildings or sub elements of buildings are by the means of another occlusion rule excluded from the information extraction. Notably, queries are not only asking for the windows area, but also differentiate between the orientation of the shape:

[10]. ALKIS uses a coding for defining the building function. In our study all buildings <1200 are classified as 'residential'.

[11] The code used in the case study is slightly simplified. A correction of the roof height is left out. See annex one.

```
Window -->
      case touches(inter):    color(wallColor)

      else:
      color(wallColor)
      s('1,'1,0.1)
      t(0,0,-0.1)
      report("window_area.south", geometry.area(world.south))
      report("window_area.north",  geometry.area(world.north))
      report("window_area.east", geometry.area(world.east))
      report("window_area.west",  geometry.area(world.west))
      i(window_asset)
```

(s = scale; t = translate; i = insert)

Figure 14: The 4 major transformation steps of the Weststadt procedural model. A) Simple cubes, B) Building with roof shapes, C) Building with floor structure & D) Building with facade structure.

The existing hierarchical structure of the procedural model allows a top down passing of the energy relevant parameters from a parent to its instance(s). A horizontal sharing of this information with other branches of the hierarchy is impossible. Therefore, a direct calculation of the heat energy can't be realized within CGA shape. The building model is exported as an ESRI FileGDB into ArcGIS Pro. The surface and volume information are imported in a R-script, where the heat energy is simulated on the basis of 180 heating days, an inner temperature of 20°C and an external temperature of 8°C (see appendix 4). In order to get the final energy values per area (kWh/a), the net heat is aggregated to the level of buildings and divided by the living space. The last step entails connecting the results of the R-script with the model in ArcGIS Pro and to visualize the buildings based on their heat energy demand (see figure 15).

Figure 15: 3D building model of the Weststadt with net heat energy per living space (kWh/a).

5. DISCUSSION

The case study demonstrated that both objectives, the creation of an accurate urban model as well as the net heat energy simulation, were realized. However, a direct integration of the energy formula within CGA failed.

The procedural modeling technique is capable of reconstructing Weststadt as a scene in LoD3 from subdivided building footprints and building information. The absence of facade information made the creation of a syntactical structure necessary. Although the facade is based on assumptions, this improvement in 'Level of Detail' is important for estimating precise heat energy values. CGA shape is not only capable of generating large models and an enormous variety of shapes based on rules, but also allows to do it on the fly and in a very efficient way (fast & small storage footprint). Modeling on the fly describes a state where the model is immediately updated, if a parameter has changed.

Shape grammar based procedural modeling also has its downside. Rules are limiting the extent of freedom to design a building. As figure 16 shows, the positioning and the number of some doors turn out to be problematic. The position of the door depends on the orientation of the front facade, which is selected randomly. The front facade and respectively the door can therefore be adjacent to surrounding buildings. A component split rule with an occlusion selector doesn't exist as of today, to overcome this situation. The existing street selectors (street.front | street.back | street.left | street.right | street.side) would be a good alternative, but are unfortunately unusable. Street selectors rely on an integrated street network that is adjacent with the initial shapes. Building footprints don't fulfil this condition due to an offset to the street. A further problem is the number of doors per front facade. The rule presumes one door per front, but in some cases we get two or more. This strange behaviour is inexplicable (This error couldn't be repeated with other initial shape layers).

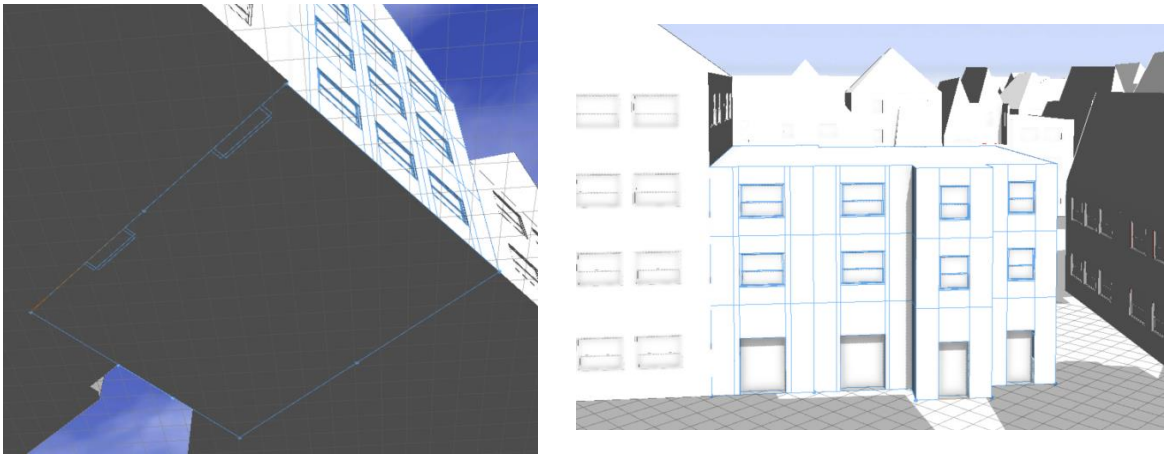


Figure 16: (Left) Bottom view on 3D building with two doors, which are adjacent with a neighbouring wall. (Right) Front view on 3D building with four doors.

The reconstruction of the Weststadt quarter on the basis of subdivided building footprints was inspired by the structure of a CityGML model and was implemented successfully. In opposition to regular building footprints this approach simplifies the initial shape and is able to create buildings with internal height differences in an efficient way. However, a holistic examination of the building is unfeasible, due to the creation of independent sub shapes, as long as an aggregation function doesn't exist.

The created spatio-semantic model with a high level of detail is the basis for a precise heat energy simulation. A necessary precondition for being able to extract energy relevant information such as volume and surface areas of the building components is the semantic structure. It represents the coupling link between CGA and the simulation method. Further, the integrated intersection and orientation capabilities within CGA shape are important to identify adjacent and non-adjacent surfaces as well as the direction of elements. Those abilities are key to determine relevant heat transmitting surfaces and to define heat gains of windows.

A complete integration of the heat energy formula within CGA is, however, not (yet) possible (or at least not possible within one .cga file). The hierarchical tree structure and the one-directional generating process from simple to complex shapes inhibit a vertical and bottom up sharing of attributes. For example, all surface areas besides the roof surface area aren't retrievable at the level of the roof shape. On these grounds, an energy calculation at the level of any shape is impossible to realize. An integration of a global variable could overcome this obstacle, but would at the same time constitute the risk of an endless loop in cases where the variable is used for generating shapes. A further disadvantage of CGA is that the formula must be integrated within the geometry construction of one specific shape. An extension of CGA with a 'non-shape-modeling' code section may be useful for a future integration with energy simulations or other non-shape modeling methods.

The approach in this case study was to outsource the heat energy simulation and the colouring of the buildings to R-Studio and ArcGIS Pro. By doing so, we lose the dynamic 'on-the-fly' modeling capacity of CGA shape. An alternative, not yet tested and maybe able to keep the dynamic model, would be to use the integrated python console in CityEngine. In the first section of the python code a building model is generated with CGA shape and energy relevant parameters are extracted, before calculating the energy simulation of the building's sub elements and aggregated back to the level of entire buildings in the second section. In the last section those heat energy values are imported back into the building model for the colouring of the envelope. Dynamic heat modeling would extend the spectrum of use cases. For example, the impact of refurbish measurement or building extension on the total heat energy or on the CO^2 emissions could be seen immediately. Or, urban planners could create a huge variety of quarter concepts by manipulating the input parameters and then take their final decisions based on the most efficient energetic concept.

In general, a further development of CGA shape is desirable. The integration of global variables, a non-geometrical code section, an aggregation functionality, a better indexing or the extension of rules (component split) would bring more control back to the users without undermining the concept of rule based modeling. Shape grammar based modeling as a technique shows enormous potential for reconstructing whole cities as accurate as possible. A new technique called 'inverse procedural modeling' may be a solution for integrating real facade structures in CGA in the future. This technique analyses pictures of house facades and translates them into a set of rules (see Mathias et al. 2011). Further, the creation of the inner of a building (LoD4) is already possible and can further improve the heat energy simulation. Nonetheless, procedural modeling is not a solution for every type of building. Complex and irregular architectural structures remain a major challenge.

6. CONCLUSION

Procedural modeling as a technique to generate complex building models is powerful in many ways. The iterative process from a simple mass to a complex one enables individual building modeling in an automated way without modeling manually. The foundation of procedural modeling relies on rules describing the structure of an architectural element and on parametrization allowing a huge variety of shapes with one rule. Simple and complex shapes with a repetitive structure can therefore be described with a few lines of code. This allows to generate large models on the fly and in an efficient way (small storage footprint & fast). The reconstruction of the real world as a spatio-semantic model depends on the available data. With a good data availability, like in our case study, a high level of detail is possible (LoD3). Missing information like from the facade can be replaced easily with a simple (or complex) structure. Furthermore, shape grammar is based on procedural modeling as a good technique for generating models that can be used for heat energy calculation. The semantic oriented modeling gives the shape a symbol that is needed for extracting energy relevant information. The integration of occlusion rules (intersection) and orientation in CGA shape are further key elements for determining the right surfaces as energy input. A direct implementation of the heat energy formula within CGA shape can't be realized and comes with the penalty of losing the 'on the fly' capability. The hierarchical structure of CGA shape does not allow for the sharing of attributes back to the parents or horizontally to other shapes which would be required for an integration. We could identify a need for further development. Not only for making an integration of energy simulation possible within CGA shape, but also to bring a greater control to the user. An important extension would be an integration of occlusion rules within the component split operation.

7. REFERENCES

ArcGIS CityEngine (2020). CGA reference. (https://doc.arcgis.com/en/cityengine/latest/cga/cga-context-queries.htm).

Chen, J.X. (2008²). Modeling and Rendering. In Chen, J.X. (Eds.) *Guide to Graphics Software Tools* (pp. 340-344). London, UK: Springer.

Diefenbach, N., Loga, T. & Stein, B. (2015) Szenarienanalysen und Monitoringkonzepte im Hinblick auf die langfristigen Klimaschutzziele im deutschen Wohngebäudebestand. (https://www.iwu.de/fileadmin/publikationen/gebaeudebestand/episcope/2015_IWU_DiefenbachEtAl_Szenarienanalysen-und-Monitoringkonzepte.pdf).

Grêt-Regamey, A., Celio, E., Klein, T.M. & Hayek, U.W. (2013). Understanding ecosystem services trade-offs with interactive procedural modeling for sustainable urban planning. *Landscape and Urban Planning*, 109, 107-116.

Haegler, S., Müller, P. & Van Gool, L. (2009). Procedural modeling for digital cultural heritage. *EURASIP Journal on Image and Video Processing*, 2009, 1-8.

Hidalgo, J.L., Camahort, E., Abad, F. & Vicent, M.J. (2008). Procedural graphics model and behavior generation. *Computational Science – ICCS*, 2008, 106–115.

Kaden, R. & Kolbe, T.H. (2013). City-wide total energy demand estimation of buildings using semantic 3D city models and statistical data. *ISPRS Ann Photogramm Remote Sens Spat Inf Sci*, 1(1), 163-171.

Krecklau, L., Pavic, D. & Kobbelt, L. (2010). Generalized use of non-terminal symbols for procedural modeling. *Computer Graphics forum*, 29 (8), 2291-2303.

Mathias, M., Martinovic, A., Weissenberg, J. & Van Gool, L. (2011). Procedural 3D building reconstruction using shape grammer and detectors. *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 304-311.

Müller, P., Wonka, P., Haegler, S., Ulmer, A. & Van Gool, L. (2006). Procedural modeling of buildings. *ACM Trans. Graph.*, 25, 614-623.

OpenLearn (2020). Energy in buildings. Heating a house. (https://www.open.edu/openlearn/nature-environment/energy-buildings/content-section-2.1).

Servais, M. (2019). Das energetische Einsparpotenzial von Wärme durch Sanierungsmaßnahmen im Wohngebäudebestand in der Region “Nordlippe” (Barntrup, Dörentrup & Extertal). Eine Simulation des Energiebedarfs bis 2030 zur Überprüfung der regionalen Klimaschutzziele (Bachelorarbeit).

Sola, A., Corchero, C., Salom, J. & Sanmarti, M. (2018). Simulation tools to build urban-scale energy models: A review. *Energies*, 11 (12), 1-24.

Strzalka, A., Bogdahn, J., Coors, V. & Eicker, U. (2011) 3D City modeling for urban scale heating energy demand forecasting. *HVAC&R Research*, 17 (4), 526-539.

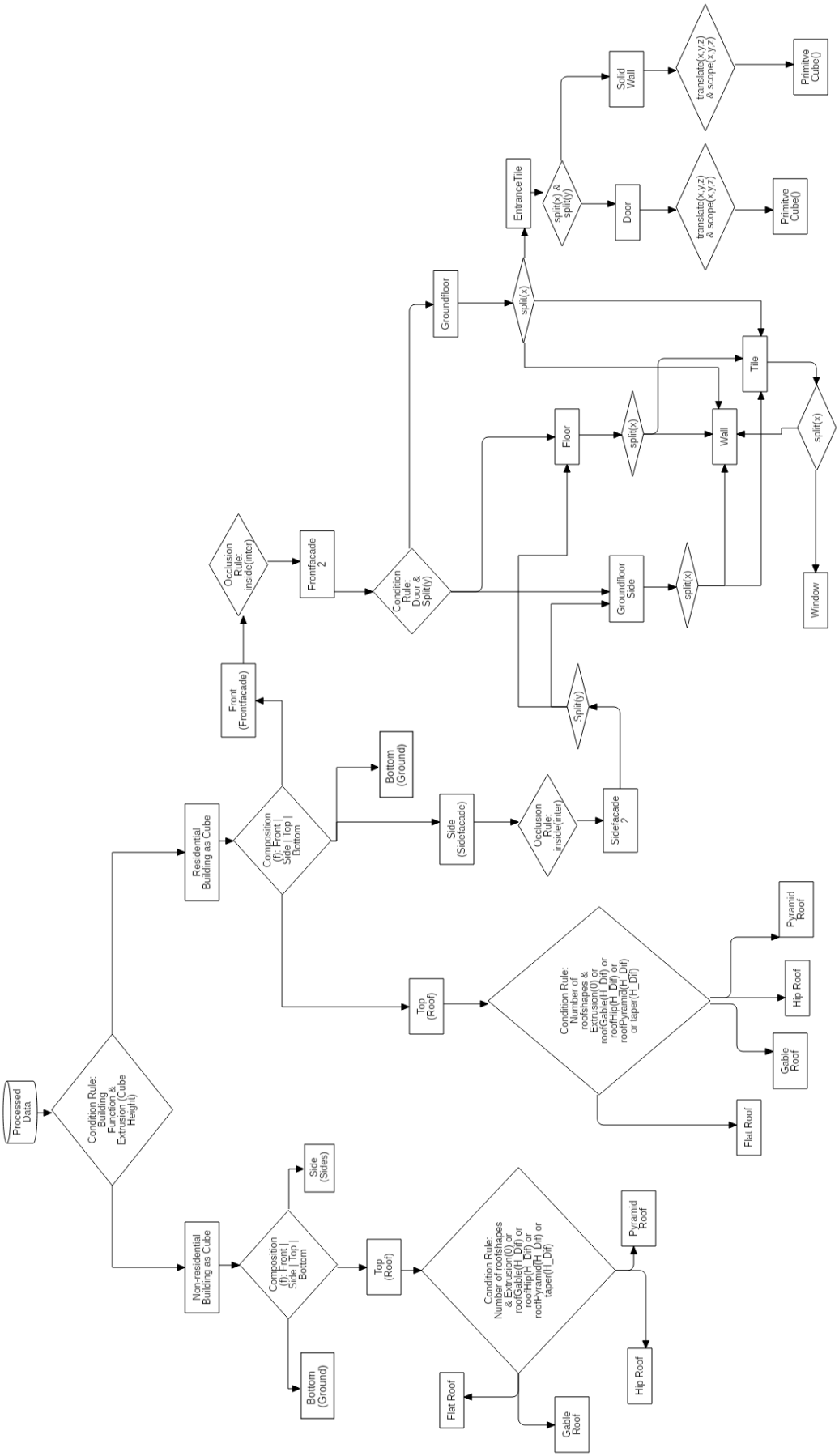
TABULA (2013). TABULA calculation method – energy use for heating and domestic hot water – Reference Calculation and adaptation to the typical level of measured consumption. (https://www.episcope.eu/fileadmin/tabula/public/docs/report/TABULA_CommonCalculationMethod.pdf).

8. APPENDIX

Appendix 1: U-values for the respective building elements and the construction period

Begin of period	End of period	Roof	Wall	Floor	Window	Door	Bridge
1860	1918	1.3	1.7	1.2	2.8	3	0.1
1919	1948	1.4	1.7	1	2.8	3	0.1
1949	1957	1.4	1.4	1.01	2.8	3	0.1
1958	1968	0.8	1.2	1.6	2.8	3	0.1
1969	1978	0.5	1	1	2.8	3	0.1
1979	1983	0.5	0.8	0.8	4.3	3	0.1
1984	1994	0.4	0.5	0.6	3.2	3	0.1
1995	2001	0.35	0.3	0.45	1.9	2	0.1
2002	2009	0.25	0.3	0.3	1.4	2	0.05
2010	2015	0.2	0.28	0.35	1.3	1.8	0.05

Appendix 2: Workflow of the Procedural Building Model



Appendix 3: CGA shape code of Procedural Building Model

```
/**

 * File:    energy_cons_preprocessing.cga
 * Created: 16 Apr 2020 13:42:49 GMT
 * Author:  marius
 */

version "2019.1"

/* Attributes ************************************/

#Buildings attributes are initialized (later: updated)
@Group("Building")
attr buildfunc = 0      #Function of building <1100 only resendital; <1200
mixed function
attr eave_H = 0   #Height     of eave
attr ridge_H = 0  #Height of ridge
attr H_dif = 0          #Height difference between eave and ridge
attr door = 0           #Door availabilty
attr roofpoly = 0       #Number of roof polygons
attr groundfloor_height = 4         #Height of ground floor
attr floor_height             = 3         #Height of floor(s)

@Color
attr wallColor = "#fefefe"    #Grey
attr red = "#ff0000"

/* Assets ***********************************/
// Geometries
window_asset = "facades/window.obj"

/* Functions ******************************/
#Function
getCorrectedHeight (H_dif) =  #For avoiding unrealistic roofshapes (higher
than 6 meters)
case H_dif > 6: 6                   #I limit the height to 6m
else: H_dif


/* Initial Shape starting rule *************/

@StartRule
#Partition in residential ('Building') and non residential buildings
('simpleBuilding')
Lot -->
      case buildfunc <1200 : extrude(eave_H)Building #<1100 only
resendital; <1200 mixed function
      else : extrude(eave_H)simpleBuilding

Building -->
      #Decomposing building in its faces
      comp(f)  {front : Frontfacade | side : Sidefacade | top: Roof
|bottom: Ground}
      #Extracting volume of the building
      report("cube_volume", geometry.volume)

simpleBuilding -->
      #Decomposing building in its faces
      comp(f)  {side : simplefacade | top: Roof |bottom: simpleGround}
```

```
 Roof -->
      #Defining roof shapes based on the number of roof polygons
      case roofpoly <2  : extrude(0)flatroof
      case roofpoly ==2 : roofGable(byHeight,
getCorrectedHeight(H_dif))gableroof
      case roofpoly > 2: 80%: roofHip(byHeight, getCorrectedHeight(H_dif))
      else: roofPyramid(byHeight, getCorrectedHeight(H_dif))hip_pyr_roof
      else: NIL

flatroof -->
      #Extracting surface area of the flat roof
      report("roof_area", geometry.area-geometry.area(bottom))

gableroof-->
      #Extracting surface area of the gable roof
      report("roof_area", geometry.area-geometry.area(bottom))

hip_pyr_roof-->
      #Extracting surface area of the hip or pyramid roof
      report("roof_area", geometry.area-geometry.area(bottom))

Ground -->
      #Extracting surface area of the ground floor
      report("floor_area", geometry.area)

Frontfacade -->
      #Extracting surface area of non-intersecting front facade
      case inside (inter): color(red)
            else: report("wall_area", geometry.area)Frontfacade_1

Frontfacade_1 -->
      #Door availabilty is checked & Vertical split of the front facade
      #Front facade with door, --> 'Groundfloor'
      #Front facade without door, --> 'Groundfloor_side'
      case door == 1:   split(y){4:Groundfloor|{~3: Floor}*}
      case door == 0: split(y){4:Groundfloor_side|{~3: Floor}*}
      else: NIL

Sidefacade -->
      #Extracting surface area of non-intersecting side facade
      case inside (inter): color(red)
      else: report("wall_area", geometry.area)Sidefacade_1

Sidefacade_1 -->
      #Vertical split of the side facade
      split(y){4:Groundfloor_side|{~3:Floor}*}

Floor -->
      #Horizontal split of the floor
      split(x){1:Wall|{~3: Tile}*|1:Wall}

Groundfloor -->
      #Horizontal split of the ground floor including entrance shape
      split(x){1:Wall|{~3:Tile}|~3:EntranceTile|1:Wall}

Groundfloor_side -->
      #Horizontal split of the ground floor without entrance shape
      split(x){1:Wall|{~3: Tile}*|1:Wall}

Tile -->
      #Vertical and horizontal split for defining the position of the
window
```

```
      split(x){~1:Wall
                  |2:split(y){~1:Wall|1.5:Window|~1:Wall}
                  |~1:Wall}

EntranceTile -->
      #Vertical and horizontal split for defining the position of the door
      split(x){~1:SolidWall
                  |2:split(y){2.5:Door|~2:SolidWall}
                  |~1:SolidWall}

Window -->
      #Integration of window in the facade when no intersection occurs
      #Extraction of the window surface area
      #Insertion of predefined window shape
      case touches(inter):    color(red)
      else:
      color(wallColor)
      s('1,'1,0.1)
      t(0,0,-0.1)
      report("window_area.south", geometry.area(world.south))
      report("window_area.north",  geometry.area(world.north))
      report("window_area.east", geometry.area(world.east))
      report("window_area.west",  geometry.area(world.west))
      i(window_asset)

Door -->
      #Creation of door as mass model object (3D) due to visualization
reasons
      #Extraction of door surface area
      s('1,'1,0.1)
      t(0,0,-0.5)
      report("door_area", geometry.area)
      primitiveCube()

Wall -->
      color(wallColor)

SolidWall -->
      #Creation of wall as 3D object due to visualization reasons
      color(wallColor)
      s('1,'1,0.4)
      t(0,0,-0.4)
      primitiveCube()
```

Appendix 4: Complete NET HEAT ENERGY FORMULA with values:

Heat transfer coefficient by transmission (H_{tr} (W/K)):
U-Value * **Area of Roof** = H_{tr} Roof (W/K)
U-Value * **Area of Wall** = H_{tr} Wall (W/K)
0.5* U-Value * **Area of Floor** = H_{tr} Floor (W/K)
U-Value * **Area of Window** = H_{tr} Window (W/K)
U-Value * **Area of Door** = H_{tr} Door (W/K)
U-Value * **Area of Envelope** = H_{tr} Bridge (W/K)

$\sum$ Htr Building Elements

Heat transfer coefficient by ventilation (H_{ve} (W/K)):
Heat Capacity Air (0.34) * Air Change Rate (0.6) * **Building Volume** = H_{ve} (W/K)

Total heat transfer (Q_{ht} (kWh/a)):
(Internal Temp. (20) - External Temp. (8) * Heating Days (180) * Conversion Factor (0.024) (H_{tr} + H_{ve}) = Q_{ht} (kWh/a)

Solar heat load during heating season (Q_{sol} (kWh/a):
Red. Factors (0.23) * Solar Radiation * **Window Area North** (160**)** = Q_{sol} North (kWh/a)
Red. Factors (0.23) * Solar Radiation * **Window Area East** (271) = Q_{sol} East (kWh/a)
Red. Factors (0.23) * Solar Radiation * **Window Area South** (392)= Q_{sol} South (kWh/a)
Red. Factors (0.23) * Solar Radiation * **Window Area West** (271) = Q_{sol} West (kWh/a)

$\sum$ Q_{sol} Window Elements

Internal heat sources (Q_{int} (kWh/a)):
Internal Heating Factor (0.072*) * Heating Days (180) * **Living Space** = Q_{int} (kWh/a)

ENERGY NEED (Net Energy) for Heating (Q_H (kWh/a)):
Q_{ht} - Gain Factor (0.9) * (Q_{sol} + Q_{int}) = Q_H (kWh/a)

NET ENERGY per AREA (Q_{Ha} (kWh/a)):
QH / Living Space = Q_{Ha} (kWh/a)