Hochschule Neubrandenburg
University of Applied Sciences

Hochschule Neubrandenburg

Bachelor-Studiengang Geoinformatik

Parametergestützte Erstellung von Gebäudetypen als 3D Gebäudemodelle mit novaFACTORY

Bachelorarbeit

vorgelegt von: Max Schultze

Zum Erlangen des akademischen Grades

„Bachelor of Engineering“ (B.Eng.)

Erstprüfer: Prof. Dr. L. Vetter

Zweitprüfer: Dipl.-Ing. J. Opitz

Dipl.-Inform. J. Schäfer

Eingereicht am: 10.02.2020

URN: urn:nbn:de:gbv:519-thesis 2019-0496-0

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

In diesem Abschnitt möchte ich denjenigen danken, die mich bei der Anfertigung dieser Bachelorarbeit tatkräftig unterstützt und mir die Arbeit ermöglicht haben.

Ein besonderer Dank geht an die Firma M.O.S.S. Computer Grafik Systeme, die mir ermöglicht hat, nach meinem Praktikum an meiner Bachelorarbeit zu arbeiten. Herr Opitz und Herr Willkomm haben mich bei Themenfindung sowie Planung und Kontrolle jederzeit unterstützt. Ihre konstruktive Kritik an den Texten hat mir geholfen diese zu verbessern. Außerdem bin ich dankbar, dass Herr Opitz als Vertreter der Firma M.O.S.S. einer meiner Zweitbetreuer ist. Ein weiterer Dank geht an Herr Hudra der mich in die Entwicklung von novaFACTORY einwies und bei Fragen jederzeit half.

Ein besonderes Dankeschön geht an meine Betreuer aus der Hochschule Neubrandenburg, Herrn Prof. Dr. Vetter und Herrn Schäfer. Herr Prof. Dr. Vetter übernahm die Arbeit als Erstbetreuer. Außerdem waren seine Anmerkungen zu Formaten und Rechtschreibung sehr hilfreich. Herr Schäfer unterstützte und hinterfragte viel, was dazu führte, dass die Arbeit strukturierter und verständlicher wurde.

Zum Abschluss möchte ich meinen Eltern dafür danken, dass sie mich jeder Zeit unterstützt und meine Gedanken geordnet haben. Gerade meine Mutter leistete einen großen Beitrag beim Korrektur lesen.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dresden, den 10.02.2020

_____

Unterschrift

Zusammenfassung

In dieser Arbeit wurde eine Methode entwickelt, um 3D Gebäudemodelle anhand von Typen zu klassifizieren und mit bestimmten parametrisierten Objekten zu erweitern. Es wurde dazu die Software novaFACTORY von der M.O.S.S. Computer Grafik Systeme genutzt. Bei der Entwicklung des Teilprogramms wurde CityGML 2.0 als Standardformat für die Gebäudemodellierung verwendet. Außerdem wurden Konzepte vorgestellt, wie man Gebäudemodelle anhand von Parametern detailgetreuer abbilden kann. Das Ergebnis ist eine Erweiterung von LoD2 Gebäudemodellen auf LoD3, um diese für komplexere Analysen nutzbar zu machen.

1. Motivation

1.1 3D Gebäude- und Stadtmodelle

In der Architektur gibt es schon lange realitätsnahe, analoge Gebäudemodelle. Diese dienen bis heute der Veranschaulichung von Gebäuden oder zur Planung städtebaulicher Veränderungen. Um diese reale Darstellung von Gebäuden zu unterstützen und dem heutigen Stand der Technik anzupassen, werden die Modelle zunehmend auch digital erzeugt. Damit ist es möglich, immer mehr Modelle schneller und veränderbarer zu erstellen, als dies mit einem analogen Modell möglich wäre [1].

Einzelne detaillierte, digitale Modelle werden schon länger mit BIM (**B**uilding **I**nformation **M**odeling) modelliert. Mit dieser Methode werden, Bauwerke anhand digitaler Gebäudemodelle so realitätsnah wie möglich abgebildet. Dabei wird nicht nur eine Version des Gebäudes im Modell gezeigt, sondern der gesamte Lebenszyklus mit all seinen relevanten Informationen wie z.B. Bauphasen und Leitungssysteme im Gebäude. Außerdem können alle Projektbeteiligten auf das Bauwerk einwirken und es verändern. Dies ermöglicht das vernetzte Management des Bauvorhabens und verbessert die Kommunikation im Bauprozess. Es ist daher eine Methode um große Bauprojekte, wie z.B. Kaufhäuser umzusetzen. Durch die erforderliche Detailtiefe ist jedoch ein extrem hoher Arbeitszeitaufwand zur Erstellung des Modells nötig. Durch diesen Zeitaufwand ist die Methode nur für besondere Gebäude wie z.B. Kirchen, Rathäuser geeignet, aber nicht für die Modellierung ganzer Städte [2].

Um Städte in 3D Stadtmodellen visualisieren zu können, muss eine automatische Erstellung der Gebäude erfolgen. Dazu können die Grundrisse von Gebäuden, die den Ämtern im ALKIS (**A**mtliches **L**iegenschafts**k**ataster **I**nformations**s**ystem) in digitaler Form vorliegen, genutzt werden. Diese Daten werden mit 3D-Informationen aus Sachdaten oder Punktwolken verknüpft und zur automatisierten Generierung von 3D Stadtmodellen verwendet. Damit ist eine generalisierte Veranschaulichung der Stadt zwar möglich, aber nur in den Detailgraden LoD1 oder LoD2 (**L**evel **o**f **D**etail siehe 2.4.1.3). Um genauere Modelle erstellen zu können, werden neue Ideen und Techniken benötigt [3].

Denn je genauer ein Modell ist, desto besser kann es für weitere Analysen genutzt werden. Mit den zurzeit herkömmlichen Stadtmodellen in LoD2 lassen sich Analysen zur

Lärmbelastung, Tourismus, Marketing oder Schattenlagen in Städten durchführen. Aber es gibt noch weitere Anwendungsfelder, wie Beleuchtungssimulationen, Starkregenereignis- und Hochwasseranalysen oder Trainingssimulationen für Feuerwehr und Polizei. Die dafür notwendigen realen Analyseergebnisse können erreicht werden, durch die Verwendung von LoD3 oder LoD4 Modellen, die genauer und detaillierter sind. Doch zurzeit gibt es noch keine flächendeckenden, detaillierten Modelle von Gebäuden, um diese genauen Analysen durchführen zu können [4].

1.2 Zielstellung der Arbeit

In dieser Arbeit geht es darum, eine Methode zu entwickeln und zu testen, mit der es möglich ist mit so wenig Parametern wie möglich, regelbasiert ein detailliertes 3D-Gebäudemodell zu erstellen, welches so genau wie nötig ist. Außerdem soll die Methode ein automatischer Prozess sein und durch die Parameter ein möglichst flexibler Einsatz gewährleistet werden.

Das Ergebnis soll für Nutzer von Stadtmodellen basierend auf CityGML 2.0 eine neue Möglichkeit bieten, 3D Stadtmodelle detaillierter darzustellen und für komplexe Analysen zu nutzen.

2. Analyse

2.1 Die Software novaFACTORY

Um 3D Gebäudemodelle erstellen zu können, benötigt man eine Software. Eine Möglichkeit ist die von der Firma M.O.S.S. entwickelte novaFACTORY. Sie ist eine umfassende Softwarelösung für die Verwaltung, Prozessierung und Bereitstellung von geotopographischen Daten. Ob Orthofotos, topographische Karten, Gelände-, Landschafts- oder 3D Gebäudemodelle novaFACTORY steigert die Effizienz der Arbeitsprozesse an jedem Arbeitsplatz mittels:

- automatisierter Erstellung von Datenprodukten (z.B. 3D Gebäuden)
- qualifizierter Bereitstellung von Daten (Import)
- zentrale Datenhaltung
- flexibler Verteilung von Daten (Export, Services) und
- performanter Visualisierung.

Aufgrund der modularen Architektur passt sich die Software ständig neuen Herausforderungen der Geodatenwelt an und ermöglicht dem Kunden viel Flexibilität.

Für die automatische 3D Produktion gibt es unter novaFACTORY verschiedene Produktionsmodule, wie 3D GDI (**G**eo**d**aten**i**nfrastruktur), 3D Solar, 3D Pro und weitere. Um Gebäudemodelle zu erstellen ist das Produktionsmodul novaFACTORY 3D Pro entwickelt worden (Abbildung 1). Dieses Modul kann mit Hilfe von weiteren Daten, wie Gebäudegrundrissen, Oberflächenpunkten aus Luftbildern und einem Geländemodell eine ganze Stadt in LoD2 produzieren. Dabei übernimmt novaFACTORY in einem automatischen Workflow alles vom Eingang (Import) der Daten bis hin zur Visualisierung und Prüfung der Ergebnismodelle. Darüber hinaus ist eine weitere Bearbeitung durch Texturieren, Erweitern oder Analysieren möglich. Am Ende der Produktion können die Gebäudemodelle in ausgewählte Formate wie z.B. CityGML exportiert werden [5].

Abbildung 1: Gebäudemodellbeispiel [5]

2.2 Methoden zur Erstellung von 3D Gebäudemodellen

2.2.1 Analoge Modelle und CAD

Heutzutage werden unterschiedlichste Modelle, wie Bauwerke, Flugzeuge, Züge oder kleine Szenen aus dem Leben, erstellt. Dadurch werden Modelle nicht nur im geschäftlichen Rahmen genutzt, sondern werden auch zu privaten Zwecken verwendet.

Die ersten Modelle wurden schon in der Antike und dem Mittelalter konstruiert. Darunter sind heute noch erhaltene Modelle von Bauwerken, wie Kirchen oder Prunkbauten. Laut [6 S. 11] wurden bereits im 14 Jahrhundert Modelle zur Baubegleitung gefertigt. Diese waren maßstäblich verkleinert und zum Teil mit originalen Materialen gebaut. Das half bei der Vermittlung von geplanten Bauausführungen gegenüber dem Auftraggeber und testete die Funktion der verwendeten Materialien. Die Grundfunktion von Gebäudemodellen, den Entwurf eines zukünftigen oder schon existierenden Bauwerks darzustellen, bleibt jedoch zu jeder Zeit gleich. Über die Jahrhunderte zeigten Modelle ihre Berechtigung, da sie für die Formfindung und für den Entscheidungsprozesses in Architektur- und Bauwettbewerben herangezogen wurden [7].

Mit der Zeit wurden jedoch die realen Gebäudemodelle immer mehr von digital erzeugten Gebäudemodellen mittels CAD (**C**omputer-**A**ided **D**esign) abgelöst (Abbildung 2). Freie Modellierungssoftware wie Blender ermöglichen jedem den Einstieg in die Welt der Modelle, ohne handwerklich begabt sein zu müssen, da diese digitalen Modelle mit Hilfe eines 3D Druckers in ein reales Modell umgesetzt werden können.

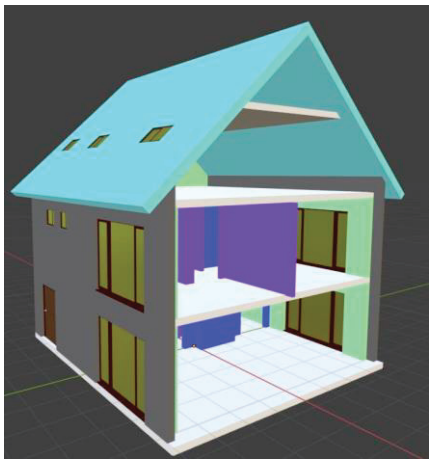


Abbildung 2: CAD Beispielmodell [23]

2.2.2 Digitale Modelle aus Punktwolken

Eine weitere digitale Methode, um 3D Modelle zu erstellen, sind Modelle, die aus Punktwolken abgeleitet wurden. Punktwolken beschreiben eine Menge von Punkten eines Vektorraumes, wobei der Begriff „Wolke“ für die unorganisierte Struktur der Menge ohne scharfe Grenzen steht. Die enthaltenden Punkte einer Wolke sind durch ihre Vektorkoordinaten erfasst. Zudem können zusätzliche Informationen (wie Farbe oder eine Messgröße) zu jedem Punkt zugeordnet werden. Ist die Punktwolke

georeferenziert, dann enthält sie Punkte in einem erdbezogenen Koordinatensystem [8]. Punkwolken können mit verschieden Methoden erstellt werden. Eine Möglichkeit ist, anhand von Bildern mit Referenzmarken und einer Software, Punktwolken errechnen zu lassen [9].

Ein anderes Beispiel für die Erzeugung von Punktwolken zur Generierung von Modellen ist LIDAR (**li**ght **d**etektion **a**nd **r**anging), welches vom Boden oder der Luft mittels Laserscanner ausgeführt werden kann (Abbildung 3) [10]. Mithilfe einer Software ist es dann möglich diese Daten zu bearbeiten. novaFACTORY ist eine solche Software und ermöglicht das Verwalten und Verändern von LIDAR Daten. Es gibt außerdem noch Produktionsmodule (novaFACTORY 3D Pro) die anhand von Workflows, großflächig und automatisiert 3D-Gebäude- und Bauwerksmodelle erzeugen. Allerdings haben diese Modelle nur einen Detailgrad von LoD2 [5].

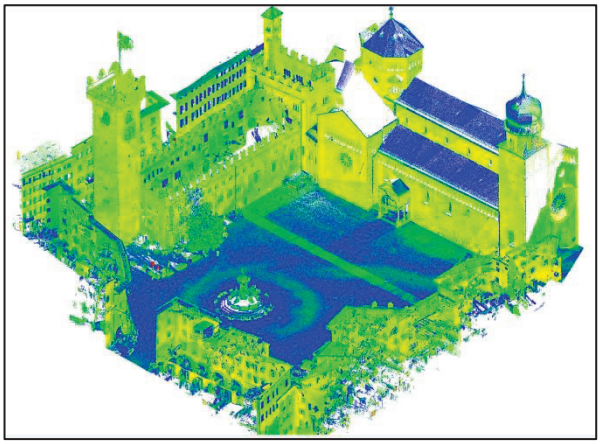


Abbildung 3: LIDAR Punktwolke [19]

2.2.3 Parameterbasierende Modelle

Ein anderer Ansatz für das Vorantreiben der Entwicklung der automatisierten Erstellung von 3D Gebäudemodellen ist die parametrisierte Beschreibung von Objekten. Damit ist gemeint, dass anhand von verschiedenen Parametern wie beispielsweise Länge, Höhe und Breite ein Gebäudemodell erzeugt werden kann. Die Parameter können dabei unterschiedliche Werte annehmen, wodurch Objekte flexibel und hinreichend genau abgebildet werden. Die Anzahl der Parameter sollte aber aus Performancegründen handhabbar bleiben, denn je mehr Parameter vorhanden sind, desto mehr Berechnungen müssen getätigt werden, um die Objekte zu erzeugen. Außerdem müssen diese Parameter durch den Anwender definiert werden. Der Fokus der parametrisierten Methode liegt auf der automatischen Verarbeitung und Erstellung der Modelle, wodurch effizientere Arbeitsabläufe und eine höhere Variabilität in der Generierung der Modelle möglich werden. Bei der Verwendung von standardisierten Gebäudetypen (wie Einfamilienhäusern oder Reihenhäusern) bietet diese Methode den Vorteil, dass in kurzer Zeit viele unterschiedliche Gebäudemodelle erstellt werden können und so ein realitätsnäheres Abbild entsteht.

2.3 Gebäudetypen

Um den hier beschriebenen Ansatz der parameterbasierenden Modelle umsetzen zu können, werden bestimmte Gebäudetypen benötigt. Bei den Gebäudetypen wurde sich für diese Arbeit an einer Publikation mit dem Titel „Wohngebäudetypologie, Beispielhafte Maßnahmen zur Verbesserung der Energieeffizienz von typischen Wohngebäuden“ orientiert [11].

Tabelle 1: Gebäudetypen

Name	Einfamilienhaus (EFH)	Reihenhaus (RH)	großes Mehrfamilienhaus (GMH)
Bauzeitraum	1958-1968	1919-1948	1969-1978
Geschosse	1	2	8
Dachform	Satteldach	Walmdach	Flachdach
Grundfläche in[m²]	~ 110	~100	~ 375
Grundmaße	10*11	10*10	15*25
Fensterzahl	2	3	6
Traufhöhe	3	6	24
Firsthöhe	7	8	24

Die gewählten Typen und Jahre richten sich nach der Tabelle 4 in „Wohngebäudetypologie, Beispielhafte Maßnahmen zur Verbesserung der Energieeffizienz von typischen Wohngebäuden“ [11 S. 18]. Diese beschäftigt sich mit der Häufigkeit des deutschen Wohngebäudebestandes bis 2009. Drei der vorgestellten Basis-Typen EFH, RH, GMH (siehe Tabelle 1) haben zu den gewählten Zeiträumen immer die höchste Anzahl an gebauten Gebäuden dieses Typs. Dazu haben alle Gebäudetypen andere typische Dächer (vgl. Tabelle 3 [11]). Die Grundflächen, sowie die Anzahl der Fenster und Geschosse wurden dem Anhang D1 von der oben genannten Publikation entnommen und die Grundmaße, sowie die verschiedenen Höhen daraus abgeleitet.

2.4 Weitere Erstellungsvoraussetzung

2.4.1 Dateiformate

2.4.1.1 XML

Um Informationen jeglicher Art maschinell verarbeiten zu können, werden einfache und strukturierte Formate benötigt. Eine Grundsprache dafür wurde im Jahre 1998 mit der e**X**tensible **M**arkup **L**anguage (XML) veröffentlicht. Sie wurde 1996 abgeleitet aus der **S**tandard **G**eneralized **M**arkup **L**anguage (SGML) die von mehreren Gruppen unter der Organisation des **W**orld **W**ide **W**eb **C**onsortiums (W3C) erstellt wurden. Die aktuelle 5th Version wurde 2008 veröffentlicht. Die Sprache wird aber noch weiter entwickelt mit den gleichen Zielen von damals, wie z.B. einfache Nutzung übers Internet, einfaches Erstellen von XML Dokumenten und weite Unterstützung von verschiedenen Programmen. Dadurch ist XML sehr flexibel gestaltet und erweiterbar.

Workflow XML Preview

```
<workflow-app name="WkflowEmail2"
    xmlns="uri:oozie:workflow:0.5">
    <start to="email"/>
    <action name="email">
        <email
            xmlns="uri:oozie:email-action:0.2">
            <to>bandalora@hortonworks.com</to>
            <subject>TEST</subject>
            <body>testing the workflow email node.</body>
        </email>
        <ok to="end"/>
        <error to="kill"/>
    </action>
    <kill name="kill">
        <message>${wf:errorMessage(wf:lastErrorNode())}</message>
```

Abbildung 4: XML Codebeispiel [22]

In den letzten Jahrzehnten haben sich deshalb für verschiedene Themenbereiche neue Formate gebildet die auf XML basieren, wie z.B. GML (siehe GML) oder **Math**ematical **M**arkup **L**anguage (MathML) [12]. Die Abbildung 4 zeigt eine Variante vom XML Code, dieser sieht je nach Bearbeitungsprogramm anders aus.

2.4.1.2 GML

Das Format **G**eography **M**arkup **L**anguage (GML) ist eine Erweiterung von XML und wird für die Verarbeitung von Geodaten verwendet. Es wurde und wird entwickelt von der **O**pen **G**eospatial **C**onsortium (OGC) und beschäftigt sich mit dem Austausch von raumbezogenen Objekten. Dieses Format erlaubt bei der Übermittlung von Objekten, das Anhängen von Attributen, Relationen und Geometrien unter Einbeziehung von z.B. Sensordaten [13]. Dafür ist die Sprache in zwei Schemata unterteilt. In dem einen stehen die Beschreibungen der Daten und in dem anderen sind die Daten selbst enthalten. In erster Linie wird die Sprache zum Darstellen von Punkten, Polygonen und Linien verwendet. Die Abbildung 5 zeigt einen Auszug aus einer GML Datei, welche abhängig vom Editor unterschiedlich dargestellt wird. Es ist gut zusehen, dass in dem Auszug ein Polygon, ein Punkt und eine Linie mittels Koordinaten beschrieben werden. Dazu dient die Struktur der Klammern am Anfang „<Point>“ und zum Ende „</Point>“ eines Objektes. Das hat zur Folge, dass eine klar lesbare Struktur entsteht, um komplexere Geometrien wie Gebäude zu erstellen.

```
<gml:Polygon>
    <gml:outerBoundaryIs>
        <gml:LinearRing>
            <gml:coordinates>0,0 100,0 100,100 0,100 0,0</gml:coordinates>
        </gml:LinearRing>
    </gml:outerBoundaryIs>
</gml:Polygon>
<gml:Point>
    <gml:coordinates>100,200</gml:coordinates>
</gml:Point>
<gml:LineString>
    <gml:coordinates>100,200 150,300</gml:coordinates>
</gml:LineString>
```

Abbildung 5: GML Codebeispiel [20]

2.4.1.3 CityGML

Eine Anwendung von GML ist CityGML (**City** **G**eography **M**arkup **L**anguage), welches seit 2008 ein Standard der OGC ist. Seit 2002 wird sie von der SIG3D (**S**pecial **I**nterest **G**roup 3D) entwickelt und bis heute unter der Führung des GDI-DE (**Geo**daten**i**nfrastruktur **De**utschlands) weitergeführt. Sie dient zur Speicherung, Modellierung und zum Austausch von virtuellen 3D Stadtmodellen. Dabei werden insbesondere Gebäude, Wasser und Verkehrsflächen dargestellt. Aber auch Vegetation, Landnutzung und Städteplanung kann damit realisiert werden. Die Objekte können dabei verschiedene Detailgrade (LoD) annehmen. Dazu werden Geometrie, Semantik und Topologie der Objekte beschrieben. Anwendung findet dieses Format beim Austausch unter Ämtern oder bei Szenarien wie 3D Stadtmodellen, Planungen und Simulationen von Gefährdungen.

Die Abbildung 6 zeigt einen Auszug eines Codebeispiels aus einer CityGML Datei. Die erste Zeile fängt mit einem Building (Gebäude) an und fügt dann Elemente wie Punkte und Polygone für Wände hinzu. Diese werden mit der gleichen Struktur wie bei GML beschrieben.

```
<bldg:Building gml:id="GMLID_BUI300347_1222_13745">
  <bldg:lod2Solid>
    <gml:Solid>
      <gml:exterior>
        <gml:CompositeSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="PolyID27581_1456_753854_82510">
              <gml:exterior>
                <gml:LinearRing gml:id="PolyID27581_1456_753854_82510_0">
                  <gml:pos>10.0 0.0 6.5</gml:pos>
                  <gml:pos>10.0 4.0 9.0</gml:pos>
                  <gml:pos>0.0 4.0 9.0</gml:pos>
                  <gml:pos>0.0 0.0 6.5</gml:pos>
                  <gml:pos>10.0 0.0 6.5</gml:pos>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
          <gml:surfaceMember>
            <gml:Polygon gml:id="PolyID27582_941_597987_91531">
              <gml:exterior>
                <gml:LinearRing gml:id="PolyID27582_941_597987_91531_0">
                  <gml:pos>10.0 1.0 0.0</gml:pos>
                  <gml:pos>10.0 8.0 0.0</gml:pos>
                  <gml:pos>10.0 8.0 3.0</gml:pos>
                  <gml:pos>10.0 1.0 3.0</gml:pos>
                  <gml:pos>10.0 1.0 0.0</gml:pos>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
```

Abbildung 6: CityGML Codebeispiel [21]

Ein Problem, welches bei der Gebäudevisualisierung auftritt, sind die durch das Format vordefinierten Einschränkungen. Zum Beispiel kann die Wandstärke nicht dargestellt werden, was zu Ungenauigkeiten zwischen Raum- und Gebäudegrößen führt. Daher müssen sich für Sonder- und Spezialfälle Möglichkeiten und Kompromisse in der Modelldarstellung überlegt werden. Diese müssen praktikabel sein und dürfen die Funktion des automatischen, parameterbasierten Arbeitsablaufes nicht stören.

Die Modelle können in verschiedenen Detailgraden erstellt werden. Das **L**evel **o**f **D**etail (LoD) gibt für jedes Modell einen Detailgrad an. Dabei wird beschrieben wie detailliert ein Modell ist oder sein sollte. Beim CityGML werden fünf verschiedene Stufen unterschieden. Diese LoD Stufen gehen von einfachen Modellen bis hin zu fertigen Architekturmodellen mit Innenraumgestaltung [1].

LoD0: „Fußabdruck“ – Beschreibt die Grundrissfläche eines Gebäudes aus der Vogelperspektive in 2D. Mit dieser Methode können Lageanalysen ohne großen Rechenaufwand durchgeführt werden (Abbildung 7).

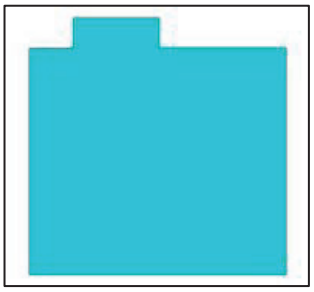


Abbildung 7: LoD0 Beispiel [1]

LoD1: „Klötzchenmodell“ – Bildet ein Gebäude in einfacher Geometrie als Flachdach-Klötzchen ab. Dies gibt die Möglichkeit für erste Planungsszenarien oder Ausbreitungsanalysen (Abbildung 8).

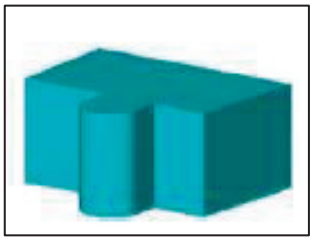


Abbildung 8: LoD1 Beispiel [1]

LoD2: „3D-Modell der Außenhülle und Dachstrukturen“ - Bildet das Gebäude in einer Näherungsform mit standardisierten Dächern ab. Dieser Detailgrad kann zu langen Renderzeiten führen und mit weiteren Bildern als Texturen verschönert werden (Abbildung 9).

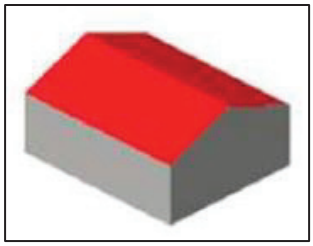


Abbildung 9: LoD2 Beispiel [1]

LoD3: „Architekturmodell“ – Bei dieser Stufe wird die Außenhülle des **LoD2** um weitere Elemente wie z.B. Türen, Fenster oder Balkone erweitert. Diese Modelle sind sehr realitätsnah und eignen sich sehr gut für die Visualisierung und Planung (Abbildung 10).

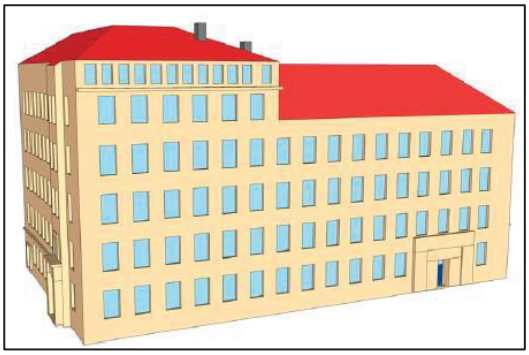


Abbildung 10: LoD3 Beispiel [1]

LoD4: „Innenraummodell“ – Die letzte Detailstufe bildet zusätzlich die Innenräume und Einrichtungselemente ab. Diese Darstellung eignet sich für Simulationen von Katastrophen oder Schadensanalysen [1] (Abbildung 11).

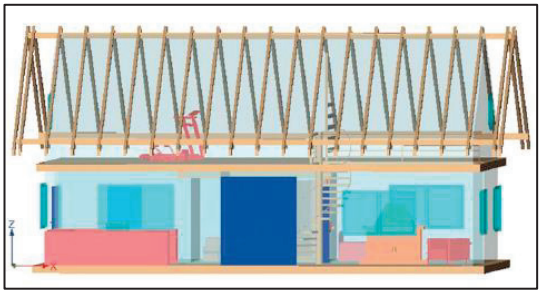


Abbildung 11:LoD4 Beispiel [1]

2.4.1.4 JSON

Ein anderes Format, welches sich sehr gut zur automatischen Verarbeitung eignet, ist JSON (**J**ava**S**cript **O**bjekt **N**otation). Dieses Format wurde erstmalig vorgestellt auf der Webseite JSON.org im Jahr 2001. Seitdem wird es immer weiterentwickelt und in Standards, wie ECMA-404 oder RFC 8259 von der IETF (**I**nternet **E**ngineering **T**ask **F**orce) übernommen. Es ist ein Textformat zum Datenaustausch, das für Menschen lesbar und einfach für Maschinen zu verarbeiten ist. Es wurde inspiriert durch JavaScript und arbeitet mit vielen Konventionen, die aus der Familie der C-basierten Programmiersprachen (C++, Java, Python und weitere) bekannt sind [14]. Auf weitere Struktureigenschaften wird in dem Punkt Gebäudekonfigurationsdatei eingegangen.

2.4.2 Sprache und Tools

2.4.2.1 Java

Java ist eine objektorientierte Programmiersprache, welche von Sun Microsystems entwickelt und 1995 veröffentlicht wurde. Seit 2010 gehört dieses Unternehmen zu einem Tochterunternehmen von Oracle. Zu den Bestandteilen von Java zählen Entwicklungswerkzeuge (JDK - **J**ava **D**evelopment **K**it) und Laufzeitumgebungen (JRE – **J**ava **R**untime **E**nvironment). Java Programme müssen jedoch nach dem Schreiben in Editoren (z.B. Netbeans) zuerst übersetzt (compiliert) werden, damit sie von einer Maschine ausgeführt werden können. Das Grundkonzept strebt nach fünf Zielen:

1. Einfach, objektorientiert, vertraut sein
2. Robust und sicher
3. Architekturneutral und portabel
4. Leistungsfähig
5. Interpretierbar, parallelisierbar und dynamisch [15].

Um in der Sprache Java programmieren zu können werden verschiedene JDK angeboten. In dieser Arbeit wurde die OpenJDK von Java genutzt.

2.4.2.2 FZKViewer

Zur Visualisierung von CityGML Gebäuden wird ein Viewer benötigt. Der FZKViewer des **K**arlsruher **I**nstitutes für **T**echnologie (KIT) ermöglicht das Darstellen von CityGML ab der Version 0.4. Dieser wird seit Release 2009 stetig weiterentwickelt. Er bietet neben der Visualisierung auch die Möglichkeit Eigenschaften und Beziehungen zwischen den Objekten textuell darzustellen. Somit lassen sich Daten auswerten und die Eigenschaften und Attribute der Objekte in Tabellen anzeigen [16]. In dieser Arbeit wurde der Viewer hauptsächlich zur Visualisierung der einzelnen Gebäudemodelle genutzt.

2.4.2.3 Netbeans

Um Programme entwickeln zu können, werden Entwicklungsumgebungen wie z.B. Apache Netbeans genutzt. Diese bietet Vorteile gegenüber einfachen Texteditoren. Dabei unterstützt die Entwicklungsumgebung das Programmieren mit Syntax-Hervorhebungen, Versionsverwaltung oder Codevervollständigung, was die Arbeit beschleunigt und vereinfacht. Netbeans ist 1996 erstmalig veröffentlich worden und wird

immer weiterentwickelt. Die aktuellste Version 11.2 wurde am 25. Oktober 2019 veröffentlicht. Netbeans wurde in Java geschrieben und wird hauptsächlich für die Entwicklung von Java Programmen verwendet [17].

2.5 Vorgaben und verwendete Formate

Für die im Folgendem beschriebenen Arbeiten wurden anhand der Analyse verschiedene Festlegungen, in Zusammenarbeit mit der Entwicklungsabteilung der Firma M.O.S.S., getroffen.

- Nutzung von Teilprozeduren aus novaFACTORY
- Verwendung der Programmiersprache Java (OpenJDK)
- Umsetzung der Software in der Entwicklungsumgebung Netbeans
- Entwicklung der Methode zur parametrisierten Erstellung von Modellen
- JSON als Konfigurationsformat zur Parameterdefinition
- Standardformat CityGML.

Für die Arbeit wurden beispielhaft drei häufige Gebäudetypen ausgewählt, die landesweit vertreten sind, da die parameterbasierte Methode unabhängig vom Standort einsetzbar sein soll. Zur visuellen Kontrolle und Darstellung der Gebäudemodelle wurde der FZK Viewer verwendet.

2.6 Workflow

Anhand der Analyse wurde ein Arbeitsablauf erstellt, welcher den Gesamtprozess einer Gebäudeerstellung beinhaltet. In der Abbildung 12 ist in schwarzen Kästchen der Arbeitsablauf gezeigt. Dabei muss zuerst eine Analyse der realen Gebäude, welche als Modell erstellt werden sollen durchgeführt werden, damit man ein Überblick über die Vielfalt der Gebäudetypen erhält. Im zweiten Punkt müssen diese Gebäudetypen definiert werden. Außerdem ist es notwendig sich Konzepte für die parametrisierte Erstellung zu überlegen (Punkt 3.1). Im dritten Schritt wird ein maschinenlesbares Format wie JSON (3.2.1) mit den nötigen Parametern erzeugt.

Der Hauptschritt ist die Modellerstellung der Gebäudetypen anhand der definierten Parameter. Die Abbildung 12 linke Seite zeigt den genauen Ablauf dieses Schrittes. Dazu ist als Ausgangsdatei die vorbereitete JSON eines Gebäudetyps notwendig. Diese wird mittels der Software novaFACTORY verarbeitet. Das Modul nfBldModel3D beinhaltet zwei Unterprogramme. Der „ModelCreator“ (roter Kasten) wurde im Rahmen dieser Arbeit entwickelt und erstellt das Grundgebäudemodell. Die zweite Komponente des Moduls „create“ erweitert das Grundgebäudemodell in LoD 2 zu einem LoD3/4 Modell. Dies ist ein wichtiger Punkt, um die Gebäudetypparameter einzustellen und zu testen.

Abschließend werden die JSON Konfigurationsdatei der vom Anwender akzeptierten Gebäudetypmodelle mittels der eineindeutigen ALKIS ID mit einem LoD2 3D Stadtmodell verknüpft. Damit ist es möglich die Gebäude dieses Stadtmodells individuell in LoD3/4 zu erweitern.

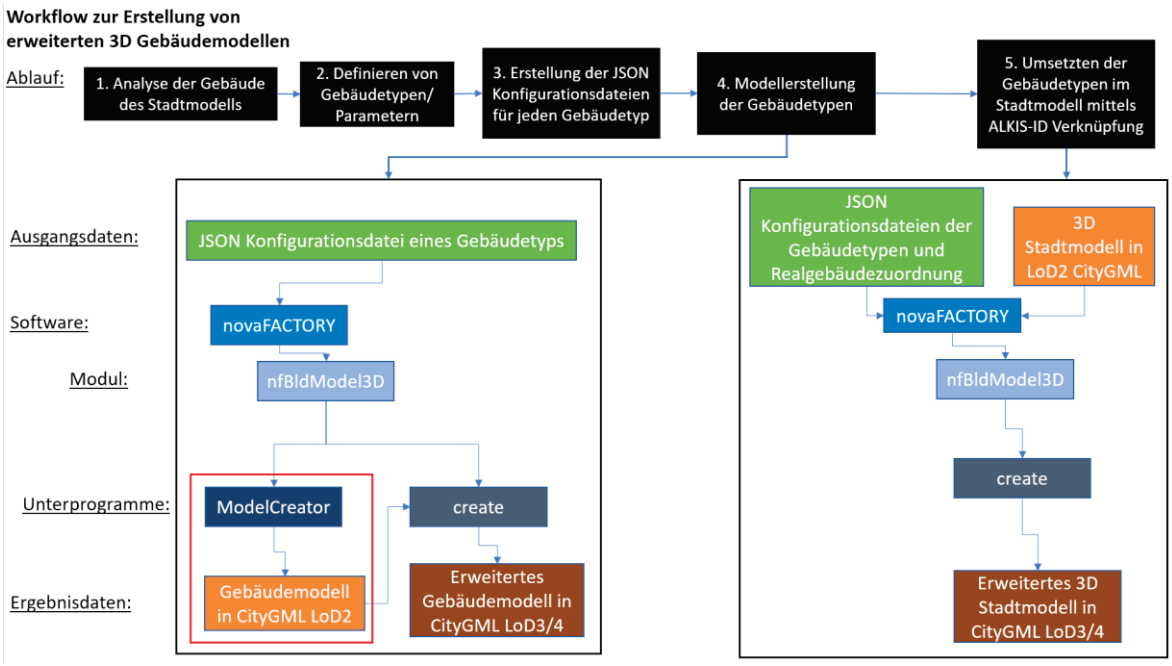


Abbildung 12: Workflow zur Gebäudeerstellung [23]

3. Praktische Arbeiten

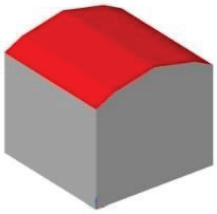
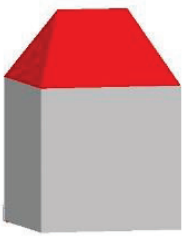
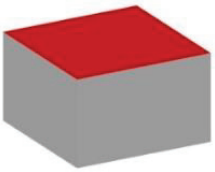
3.1 Konzepte

Die Konzepte, welche in Zusammenarbeit mit Mitarbeitern der Firma M.O.S.S. erarbeitet wurden, werden im folgenden Abschnitt beschrieben. Dabei wurde großen Wert auf die automatische Verarbeitung und die Umsetzbarkeit mit CityGML 2.0 gelegt. Im folgenden Kapitel werden vermehrt Parameterdefinitionen z.B. („RO“) benutzt. Die Parameterdefinitionen sind im Rahmen der Arbeit entstanden und setzten sich zum Großteil aus den ersten zwei Buchstaben des englischen Wortes für das Objekt zusammen (z.B. für Dach = Roof „RO“). Es gibt jedoch Ausnahmen, wo nur der erste Buchstabe (z.B. „H“ für Horizontal) oder mehr als zwei Buchstaben genutzt werden (z.B. „Len“ für die Länge eines Objektes und „Dist“ für den Abstand). Alle verwendeten Parameter sind im Anhang in den Tabelle 3 und Tabelle 4 mit den dazugehörigen Beschreibungen aufgelistet.

3.1.1 Grundgebäudekonzept

Jedes Grundgebäudemodell („BU“) hat bestimmte Voraussetzungen, die es erfüllen muss. Dazu gehören die Grundform, welche rechteckig sein sollte, eine Grundfläche, vier Wandflächen und ein Dach. Unter diesen Bedingungen kommt die Form einem realen LoD2 Modell am nächsten. Dies ist für ein späteres Anwenden der Gebäudetyp-spezifischen Parameter in einem vorhandenen 3D Stadtmodell in LoD2 wichtig, wird aber in dieser Arbeit nicht weiter thematisiert.

Tabelle 2: Dachformen

1. Sattel-	2. Pult-	3. Mansarden-	4. Walm-	5. Flachdach
				

Eine weitere Grundvoraussetzung ist die Verwendung von unterschiedlichen standardisierten Dachformen („shape“). Die Codeliste der SIG 3D in Coors, Andrae, & Böhm, 2016, (S. 134) beschreibt eine Vielzahl von Dachformen.

Zur Veranschaulichung wurden beispielhaft fünf dieser Dachtypen ausgewählt (Tabelle 2). Die gewählten Dachformen sind Sattel-, Pult-, Mansarden-, Walm- und Flachdach. Diese werden anhand eines Parameters („RoSh“) für das jeweilige Modell festgelegt. Weitere Parameter, sind Traufhöhe („BuHe“), Breite („BuWid“) und Länge („BuLen“) des Gebäudes (Abbildung 13).

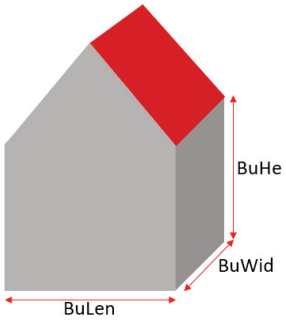


Abbildung 13: Grundgebäudekonzept [23]

3.1.2 Konzept der Gebäudeaußenhülle

Um das Grundgebäudemodell im nächsten Schritt zu erweitern, bedarf es weiterer Objekteinteilungen und Parameterzuweisungen.

3.1.2.1 Allgemeine Objekte

Fenster:

Das erste Objekt ist ein Fenster („window“). Dieses Objekt Fenster hat eigene Parameter, welche die Größe des Fensters bestimmen. Außerdem gibt es Abstandsparameter, um ein Fenster an die richtige Stelle im Geschoss zu setzen. Dabei können für jedes Geschoss unterschiedliche Werte für die Parameter angeben werden. Die Abbildung 14 zeigt die Größenparameter Fensterhöhe („WiHe“) und Fensterlänge („WiLen“). Diese gelten für jedes Fenster innerhalb einer Etage. Weitere Parameter, sind in Abbildung 15 gezeigt. Dabei gibt es Unterschiede in der Bedeutung der Parameter. Die in orange gekennzeichneten Parameter sind Sicherheitsparameter („VDistCeWi“, „HDistMinWaWi“).
Sie sorgen dafür, dass das Fenster nicht über

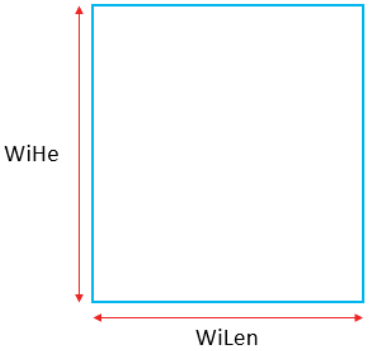


Abbildung 14: Fensterparameter [23]

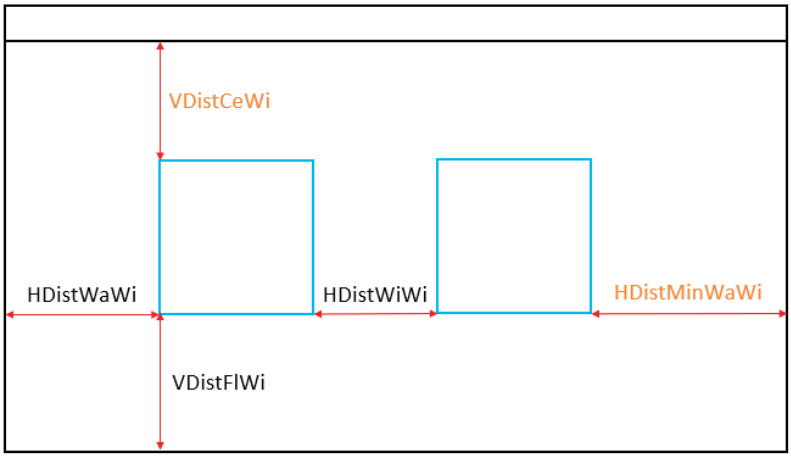


Abbildung 15: Abstandsparameter der Fenster [23]

ein Geschoss oder eine Wand hinausragt. Die anderen Parameter beschreiben die jeweiligen Abstände („Dist“) in horizontaler („H“) und vertikaler („V“) Richtung, zwischen den Objekten Fenster („Wi“), Wand („Wa“) und Boden („Fl“).

Tür:

Ein weiteres Objekt ist die Tür („door“). Sie wird nur im Erdgeschoss generiert und über die Parameter Türhöhe („DoHe“) und Türlänge („DoLen“) in ihrer Ausdehnung festgelegt (Abbildung 16). Es ist immer nur möglich pro Wand eine Tür zu erstellen. Jedoch sind weitere Türen auf den anderen Wandflächen möglich.

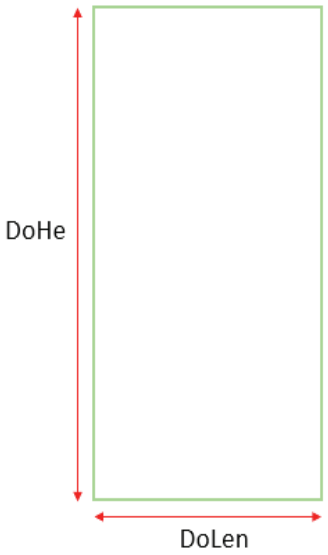


Abbildung 16: Türparameter [23]

Die Abstandsparameter sind in Abbildung 17 zusehen. Dabei entscheidet der horizontale Abstand zwischen Tür und Wand („HDistDoWa“) wo die Tür platziert wird. Die anderen beiden Parameter in orange sind weitere Sicherheitsparameter. Sie sorgen dafür, dass die Tür nicht über die Wand hinaus platziert wird oder sie aufgrund der Höhe bis ins Obergeschoss ragt. Sollten sich Fenster und Tür überschneiden wird das Fenster nicht dargestellt. Erst wenn eine weitere Erstellung möglich ist, wird ein neues Fenster in die Etage gesetzt.

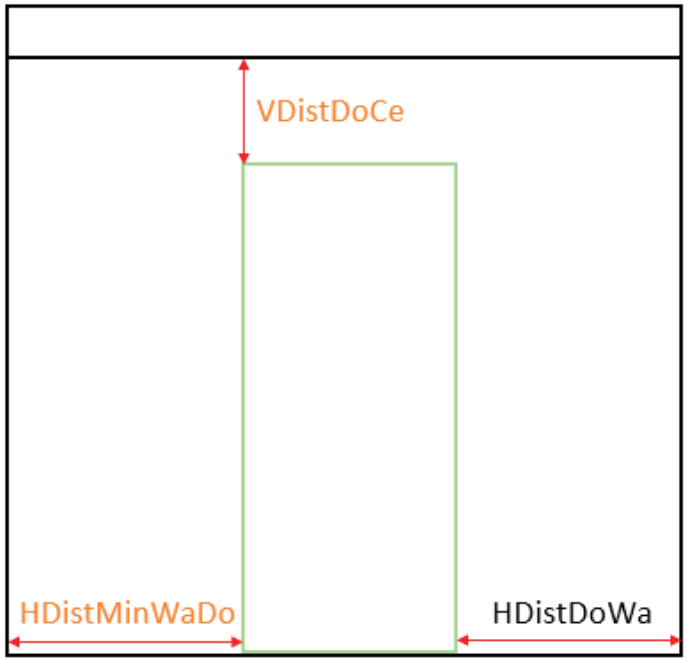


Abbildung 17: Abstandsparameter der Tür [23]

3.1.2.2 Geschossobjekte

Die allgemeine Einteilung eines Gebäudemodells erfolgt in Etagen. Diese sind Keller- („BA“), Erd- („GF“), Obergeschoss („UF“) und ein Dach („RO“).

Der Keller und das Obergeschoss müssen nicht vorhanden sein, wenn es sich um ein Gebäude mit nur einer Etage handelt. Über die in Abbildung 18 gezeigten Parameter lässt sich ein Gebäude in die Geschosse einteilen. Der Parameter („heightGr“) ist jedoch nur am Erdgeschoss anwendbar, da dieser Parameter die Höhe eines Erdgeschosses über der Erdoberfläche definiert. Des Weiteren gibt „height“ die Höhe der Etage an und „CeHe“ die Stärke der Decke. Sollte ein Gebäude hoch genug sein, um mehrere Geschosse zu beinhalten, werden die Obergeschosse mit ihren Geschossparametern in das Modell eingebaut. Der Dachboden wurde in dieser Arbeit nicht berücksichtigt, da er nicht immer ausgebaut ist. Die definierten Parameter sind die jeweiligen Gesamtparameter eines Objektes. Nicht jede Etage greift aber auf alle diese Parameter zurück. Weiterhin gibt es Unterschiede in den Geschossen.

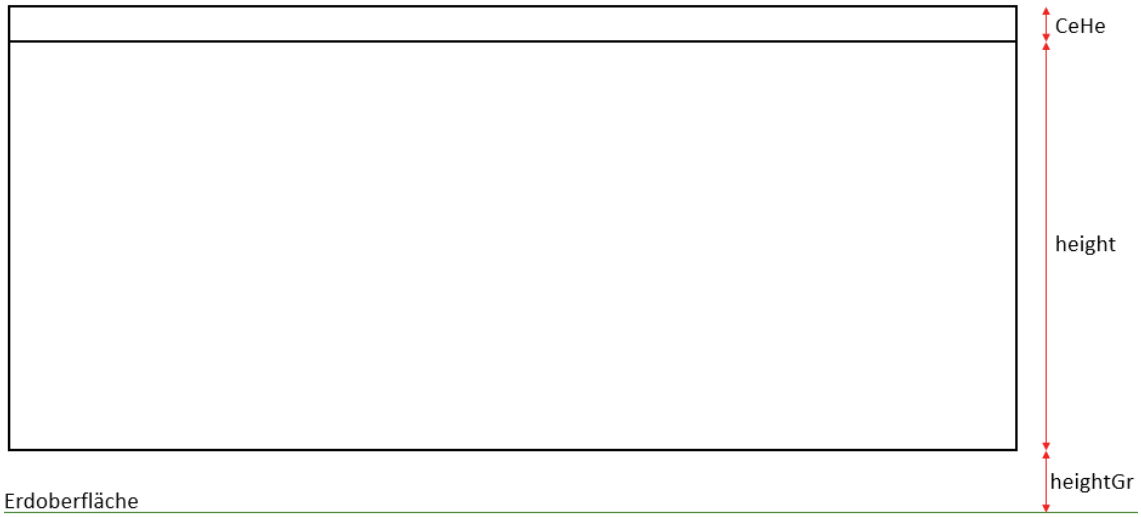


Abbildung 18: Geschossparameter [23]

Der Keller („BA“) ist in einem LoD2 Modell üblicherweise noch nicht definiert. Er soll durch Verlängerung der Wandflächen nach unten hinzugefügt werden. Damit entsteht die Möglichkeit in einen Keller Fenster und eine Tür mit den zuvor beschrieben Parametern einzupassen. Dies ist aber nicht zwingend erforderlich, sollte das Modell kein Keller besitzen.

Die nächste Etage ist das Erdgeschoss („GF“). Im Erdgeschoss stehen zwei Konzepte zur Generierung der Fenster zur Debatte.

Das erste Konzept greift auf die beschriebenen Parameter in 3.1.2.1 zurück, welche die Fenster und Türen unabhängig voneinander generiert. Dabei hat die Tür Priorität, falls sich Fenster und Tür überscheiden sollten.

Das andere Konzept verfolgt den Weg wie in Abbildung 19 dargestellt. Bei diesem Konzept werden die Türen und Fenster nicht getrennt voneinander generiert. Nacheinander werden zuerst die Tür und dann von ihr ausgehend die Fenster erstellt. Dies erfolgt gleichmäßig zu beiden Seiten angefangen mit „HDistDoWi“ und „HDistWiWi“ bis der „HDistMinWaWi“ als Sicherheitsparameter erreicht ist.

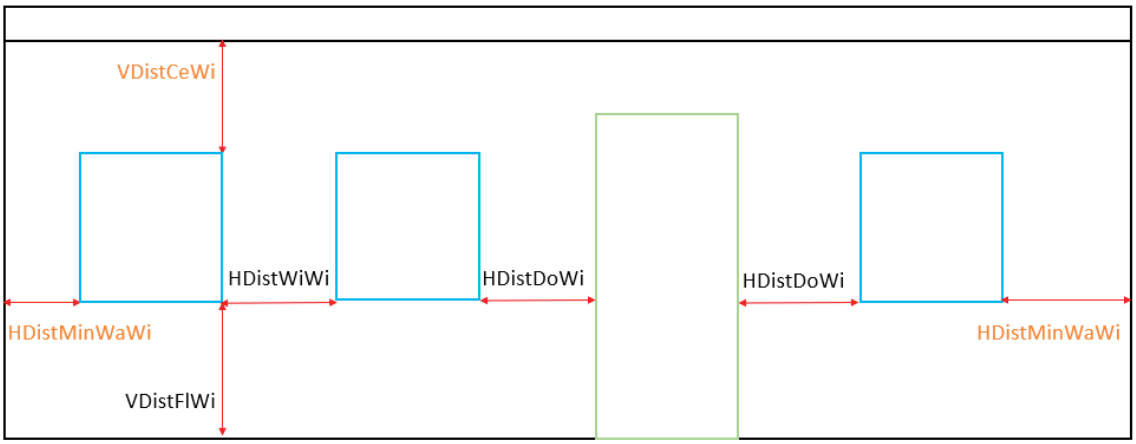


Abbildung 19: Schema der Geschossgenerierung [23]

Im Obergeschoss („UF“) finden sich die in Punkt 3.1.2.1 beschriebenen Fensterparameter wieder. Darüber hinaus sollte ein Gebäude über mehr als nur ein Obergeschoss verfügen, werden diese genauso generiert wie das erste Obergeschoss.

Das Dach schließt das Gebäude ab. Dabei werden die in 3.1.1 vorgestellten Dachformen umgesetzt. Für die meisten dieser Dächer wird nur ein zusätzlicher Parameter - die Firsthöhe („RiHe“) - benötigt. Damit lassen sich Sattel-, Pult-, und Flachdach generieren. Für das Mansarden- und Walmdach sind noch zusätzliche Parameter notwendig. Das Mansardendach hat in der Dachfläche unterschiedliche Dachneigungen. Dadurch entstehen vier Dachflächen und nicht nur zwei wie z.B. beim Satteldach. Die Neigung der Dachfläche wird nicht mit Winkeln dargestellt, da es bei Winkeloperationen zu Rundungsfehlern kommen kann, die dazu führen, dass das Dach nicht erstellt wird. Deshalb werden zur Beschreibung der Dachneigung Strecken („HSiTi“) und („VSiTi“) verwendet (Abbildung 20).

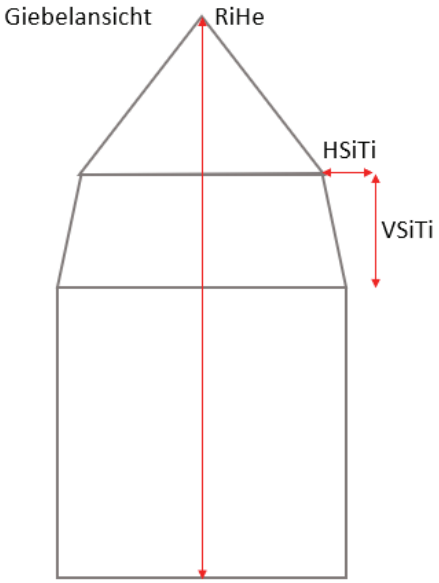


Abbildung 20: Mansardendachparameter [23]

Bei einem Walmdach werden die funktionsgleichen zusätzlichen Parameter verwendet wie beim Mansardendach, mit dem Unterscheid, dass sie nicht auf die Dachflächenseiten, sondern auf der Giebelseite angewendet werden (Abbildung 21).

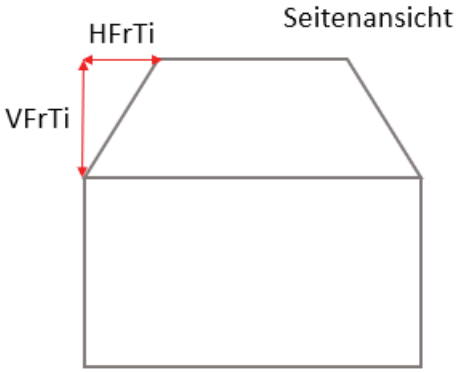


Abbildung 21: Walmdachparameter [23]

Um in einer Dachfläche Fenster erstellen zu können, werden die in 3.1.2.1 vorgestellten Fensterparameter ebenfalls genutzt.

3.1.2.3 Weitere Objekte

Balkon:

Bei den Balkonen („GA“) gibt es zwei verschiede Konzeptansätze, um die Außenhülle zu erweitern. Dabei ist zu beachten, dass Balkone nicht im Erdgeschoss und Kellergeschoss erstellt werden können.

Beim ersten Konzept liegt der Fokus auf der getrennten Generierung aller Objekte. Das bedeutet, die Fensterparameter können normal angewendet werden. Dazu kommen die Balkonparameter, welche in Abbildung 22 zu sehen sind. Diese beschreiben die Maße des Balkons mit Balkonhöhe („GaHe“), Balkonlänge („GaLen“), Balkonbreite („GaWid“).

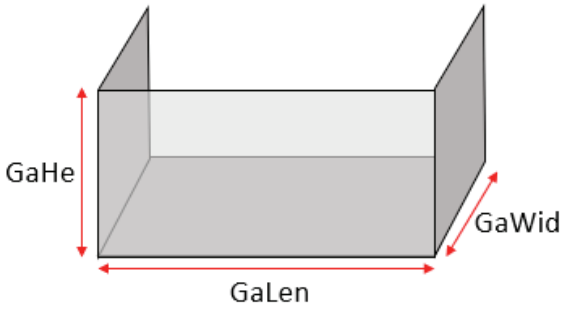


Abbildung 22: Balkonparameter [23]

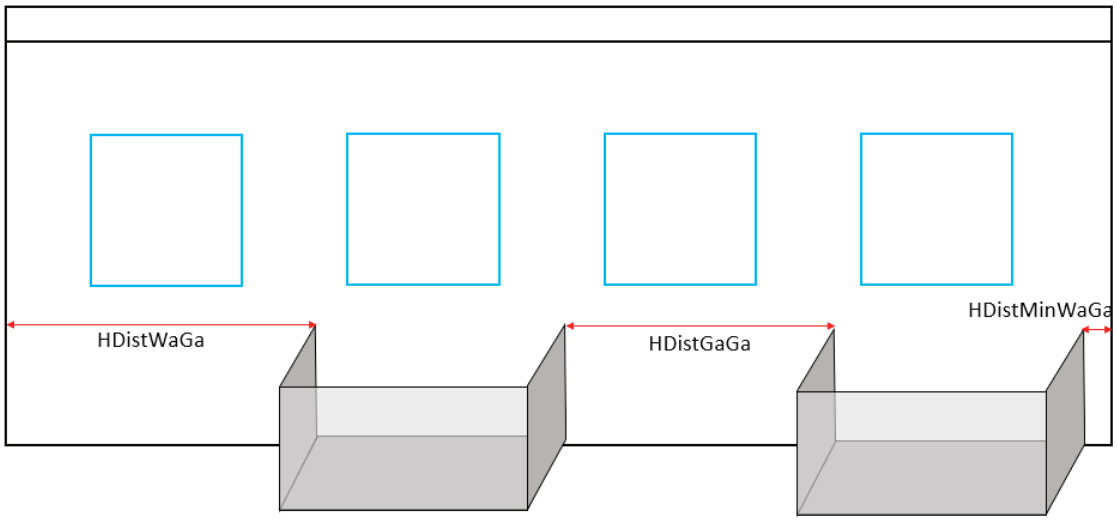


Abbildung 23: Abstandsparameter des Balkons (Konzept 1) [23]

Die Abstandsparameter zu den Wänden, Fenstern und weiteren Balkonen werden in Abbildung 23 aufgezeigt. Ein Sicherheitsparameter ist ebenfalls notwendig damit sich Wand und Balkon nicht überschneiden. Sollten sich eine Balkonseite und ein Fenster schneiden so wird der Balkon nicht dargestellt.

Im zweiten Konzept werden die Fenster wie in den Etagen üblich erstellt. Das Balkonmodul bekommt dabei immer ein festes Fenster. Das ermöglicht eine größere Flexibilität in der Anordnung von Fenster mit Balkon und normalen Fenstern. Sollte eine Überschneidung auftreten wird der Balkon mit Fenster priorisiert dargestellt. Die Parameter, die im zweiten Konzept hinzu kommen sind, („DistWiGa“) für die Position innerhalb des Balkons und („WiLen“), („WiHe“) für das neue Balkonfenster (Abbildung 24).

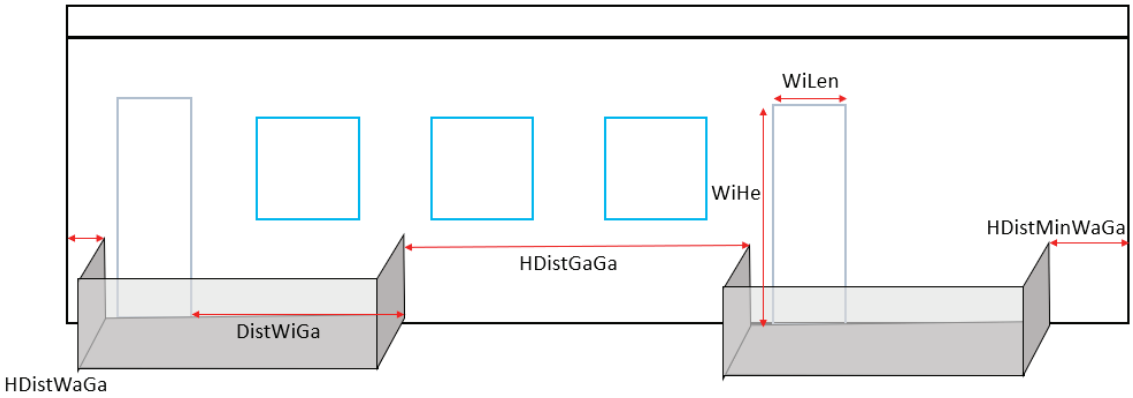


Abbildung 24: Abstandsparameter des Balkons (Konzept 2) [23]

Innenräume:

Das Konzept zur Erstellung der Innenräume („IN“) ist anders aufgebaut als bei den Etagen oder den Objekten der Außenhülle. Es werden keine Abstände oder Größen als Parameter angegeben, sondern Parameter zur Beschreibung einer Matrix. Um die Innenräume zu erstellen, wird zuerst ein umschreibendes Rechteck, welches um das vorhandene Gebäudepolygon (graue Fläche Abbildung 25) gezogen wird benötigt.

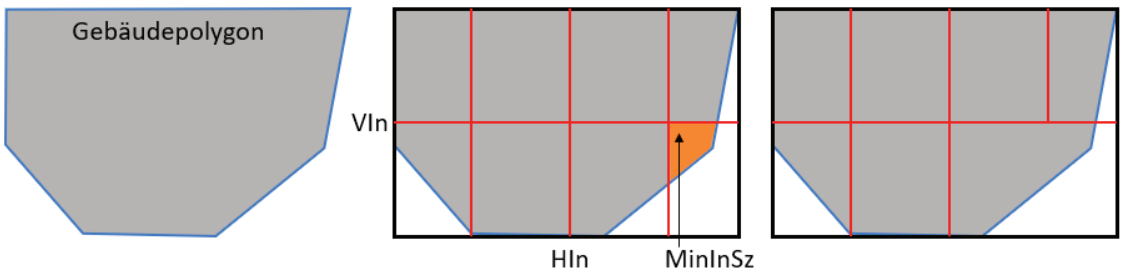


Abbildung 25: Innenraumkonzept [23]

Dann wird anhand der Parameter („HIn“) und („VIn“) festgelegt in wie viele Teile das Rechteck zerlegt werden soll (rote Linien). Dabei können Räume entstehen, die vom Anwender anhand eines Sicherheitsparameters(„MinInSz“) als zu klein definiert wurden.

Um diese Räume zu vergrößern wird eine beliebige angrenzende Wand entfernt (Abbildung 25 rechts). Die entstandenen Innenräume werden für alle Etagen übernommen.

Ein weiteres Objekt im Innenraum ist die Treppenöffnung („FL“) [18]. Sie wird mittels eines Vierecks im Geschossboden dargestellt und zieht sich an dieser Position durch alle Etagen. Dieses Objekt ist besonders wichtig für Schadensanalysen mit Wasser, um den Durchlauf zwischen den Stockwerken zu berücksichtigen. Es wird durch die Breite und Länge definiert. Außerdem kann es, je nachdem wie die Innenräume erstellt werden, mit den Abstandsparametern verschoben werden. Das ermöglicht eine individuelle Platzierung der Treppenöffnung (Abbildung 26).

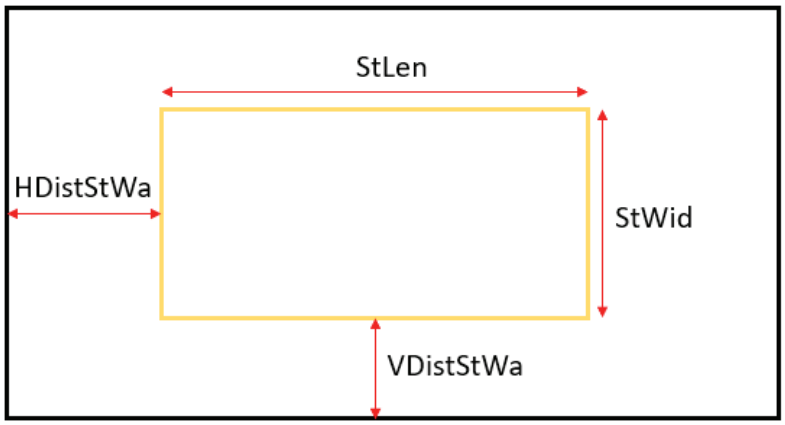


Abbildung 26: Treppenöffnungskonzept [23]

Hausanschlüsse:

Zur Kennzeichnung der Hausanschlüsse („UT“) für z.B. Strom, Internet, und Abwasser wird in einer Fläche ein Punkt über seine Abstände von Wand und Boden definiert. Diese Abstandsparameter sind universell einsetzbar und es ist sogar eine grobe Veranschaulichung von Rohr- und Wasserleitungen möglich, welche bei Hochwasser einen möglichen Schwachpunkt oder eine Gefahrenquelle darstellen können (Abbildung 27).

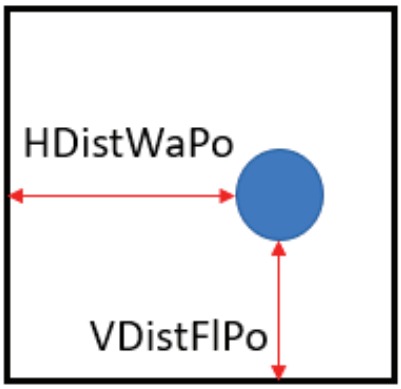


Abbildung 27: Hausanschlusskonzept [23]

3.2 Erstellungsprogramm

3.2.1 Gebäudekonfigurationsdatei

Um die Parameter automatisch verarbeiten zu können wird eine maschinenlesbare Konfigurationsdatei (z.B. im Format JSON) benötigt. Diese Datei unterliegt einer bestimmten Objektstruktur, die in Abbildung 28 dargestellt ist. Dabei wird alles als Objekt deklariert, was zwischen den geschweiften Klammern am Anfang und am Ende liegt.

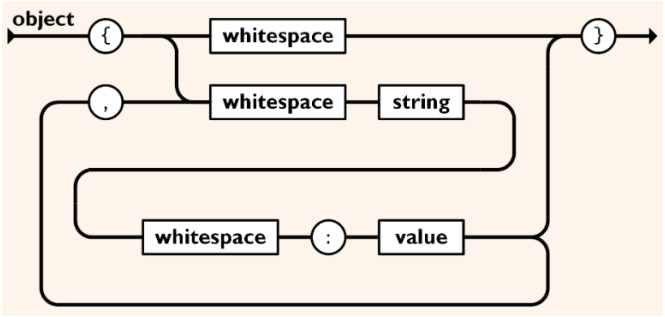


Abbildung 28: JSON Struktur [24]

Die Abbildung 29 zeigt einen Ausschnitt aus einer Konfigurationsdatei am Beispiel der Grundfläche („GF“). („GF“) ist ein String und wird durch den Doppelpunkt („:“) mit „values“ oder auch Werten gefüllt. Diese Inhalte können weitere Strings, Listen oder Objekte sein. „height: 3.10“, aus Abbildung 29 ist z.B. ein String mit dem Wert 3.10. Durch die Kommata werden die Inhalte aufgezählt. „window“: {Values} ist ein Beispiel für ein Objekt in dem Objekt („GF“). Dieses Format ist für Maschinen sehr gut zu lesen und lässt sich aufgrund der Ähnlichkeiten zur Java Struktur gut verarbeiten. Die Gebäudemodelle sind so aufgebaut, dass durch die Konfigurationsdatei die verschiedenen Geschosse (Keller-, Erd- und Obergeschoss) Objekte sind und diese weiteren Objekte für die Tür oder die Fenster enthalten. So lassen sich alle vorgestellten Konzeptparameter einbinden und verarbeiten. Dabei wird natürlich nicht immer jeder Parameter für jedes Gebäude benötigt. Aber jeder Parameter hat seine Funktion.

```
"GF": {
      "height": 3.10,
      "heightGr": 0.0,
      "CeHe": 0.2,
      "window": {
            "HDistDoWi": 0.5,
            "VDistFlWi": 0.8,
            "HDistWiWi": 2.23,
            "WiLen": 1.25,
            "WiHe": 1.60,
            "HDistMinWaWi": 1.57
      },
      "door": {
            "DoHe": 2.2,
            "DoLen": 1.8,
            "HDistDoWa": 11.1
      }
},
"UF": {
      "height": 3.10,
      "CeHe": 0.2,
      "window": {
            "HDistWaWi": 1.15,
            "VDistFlWi": 0.3,
            "HDistWiWi": 0.8,
            "WiLen": 0.7,
            "WiHe": 0.32,
            "HDistMinWaWi": 0.5
      }
},
"RO": {
```

Abbildung 29: Auszug der Konfigurationsdatei [23]

3.2.2 Programm nfBldModel3D

Im Rahmen der Entwicklung der Firma M.O.S.S. Computergrafik Systeme GmbH wurde ein neues novaFACTORY Modul mit dem Namen nfBldModel3D erarbeitet. Dieses Modul hat mehrere Unterprogramme, von denen der „ModelCreator“ im Rahmen dieser Arbeit programmtechnisch umgesetzt wurde. Das restliche Modul wurde nur konzepttechnisch mit entwickelt.

Der grobe Ablauf des Programms ist in dem Programmablaufplan (Abbildung 30) zusehen. Dabei werden als erstes die Verzeichnisse der Konfigurationsdatei und der Ausgabe dem Programm übergeben. Danach wird die Konfigurationsdatei verarbeitet, Default-Werte werden gesetzt und in Java Klassen (Properties) hinterlegt.

Der „ModelCreator“ greift auf die nötigen Properties zu und erstellt daraus das Grundgebäudemodell in CityGML.

Als letzter Verarbeitungsschritt wird das erstellte Grundgebäudemodell im Unterprogramm „create“ mit den weiteren Objekten wie Fenster und Türen erweitert und im Ausgabeverzeichnis abgelegt.

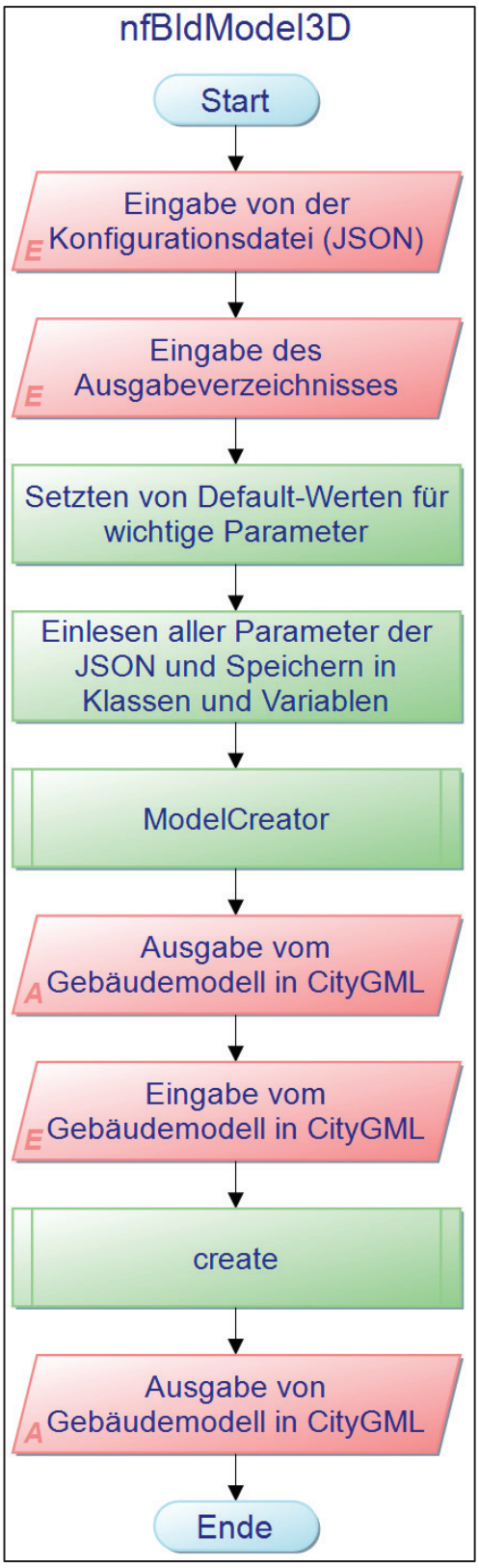


Abbildung 30: Programmablaufplan des nfBldModel3D von novaFACTORY [23]

3.2.3 „ModelCreator“

3.2.3.1 Überblick

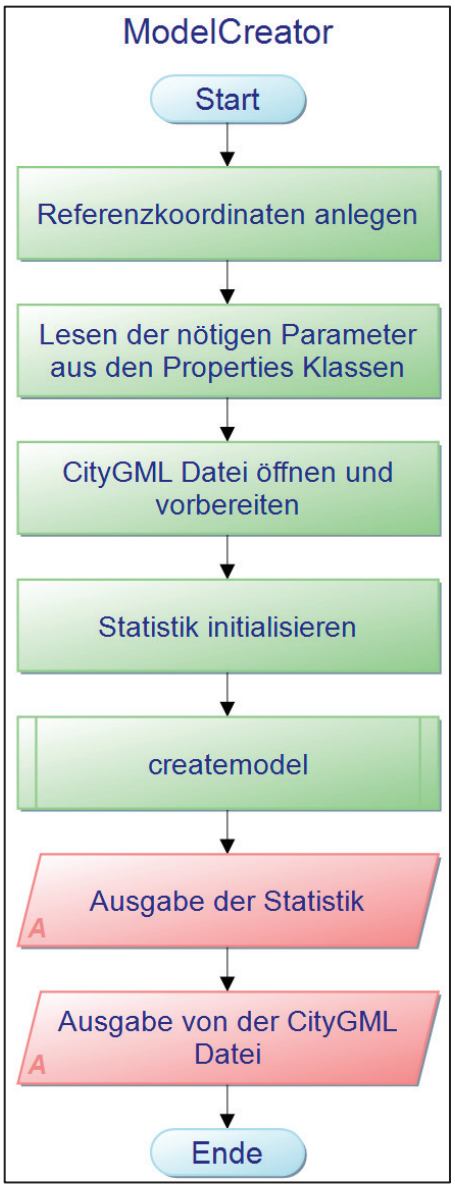


Abbildung 31: Programmablaufplan des „ModelCreator“ [23]

Der Programmablauf des „ModelCreator“ ist in Abbildung 31 dargestellt. Zuerst werden Variablen für Referenzkoordinaten angelegt. Sie haben zur Folge, dass die Modelle in dem gewünschtem Referenzkoordinatensystem erstellt werden. Danach können die Parameter aus den nötigen Properties geladen und auf neue Variablen geschrieben werden. Außerdem wird eine CityGML Datei zum Schreiben vorbereitet und eine Statistik initialisiert. Die Statistik ermöglicht es, bei der Ausgabe die veränderten und erstellten Objekte zu überprüfen. Die Methode „createmodel“ ist für die eigentliche Erstellung des Grundgebäudemodells verantwortlich. Zum Abschluss wird die Statistik ausgegeben und das Gebäudemodell in CityGML in das Ausgabeverzeichnis abgelegt.

Die im „ModelCreator“ vorbereiteten Objekte werden dann der Methode „createmodel“ übergeben, welche die einzelnen Grund-, Wand- und Dachflächen erzeugt. Der Ablauf der Methode wird in Abbildung 32 schrittweise erläutert.

3.2.3.2 Methode „createmodel“

Als Erstes wird überprüft, ob alle notwendigen Ausgangsdateien vorhanden sind. Dazu gehören die zum Schreiben vorbereitete CityGML Datei und die Konfigurationsdatei, aus der die Parameter Properties gelesen wurden. Jetzt werden Variablen vorbereitet und entweder mit dem dazugehörigen Wert aus der Konfigurationsdatei oder dem im Programm gesetzten Default-Wert gefüllt. Danach werden die einzelnen Flächen erstellt.

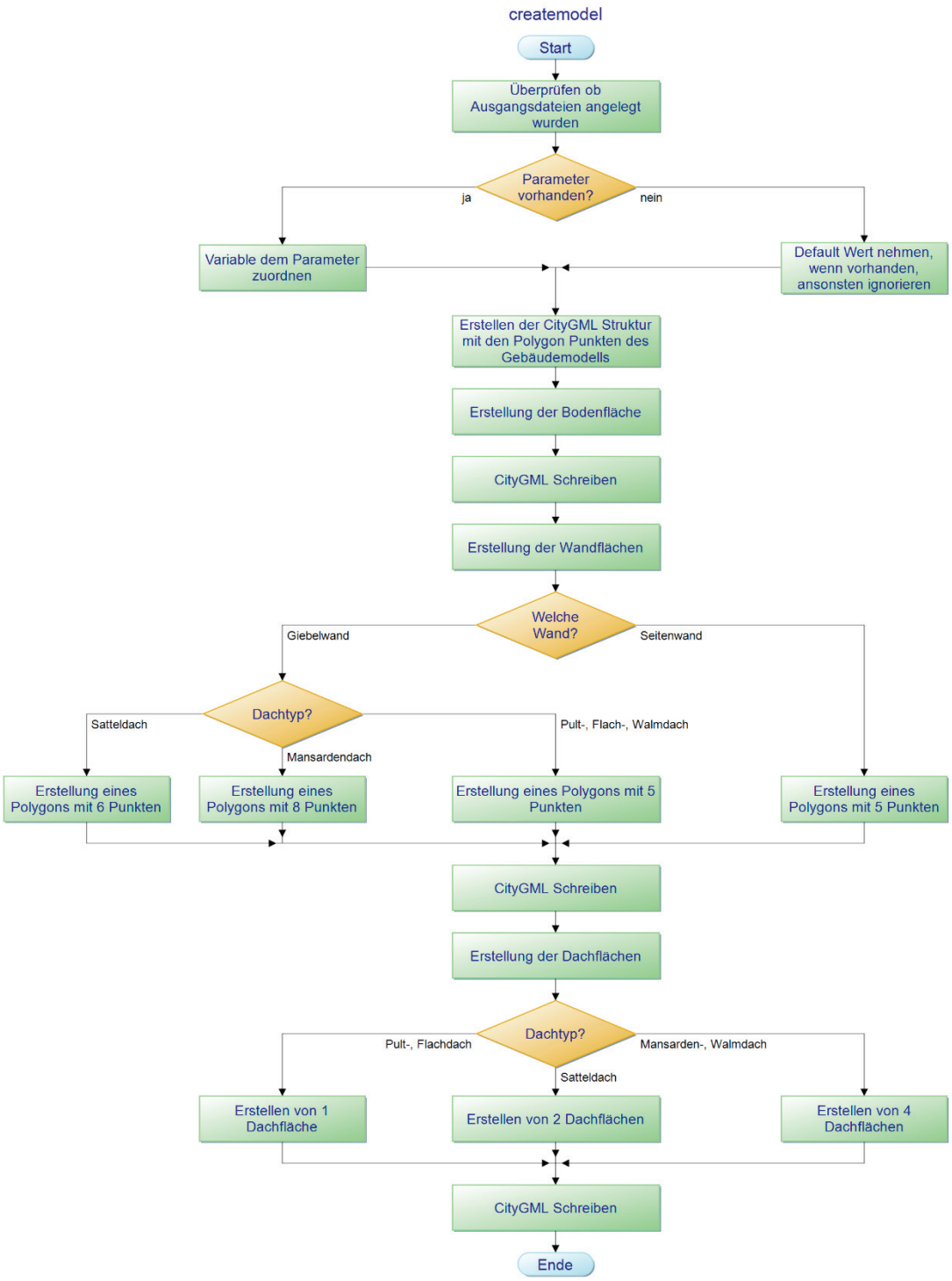


Abbildung 32: Programmablaufplan der Methode „createmodel“ [23]

Es wurde darauf geachtet, dass es sich bei der Bodenfläche der Modelle immer um ein Viereck handelt. Dieses wird durch ein Polygon aus fünf Punkten dargestellt, da der erste und letzte Punkt des Polygons immer gleich sein müssen Die erzeugte Bodenfläche wird anschließend in die vorbereitete CityGML Datei geschrieben. Im nächsten Schritt werden die vier Wände erstellt. Dabei wird zwischen Giebel- und Seitenwänden

unterschieden. Die Polygone der Giebelwände können, aufgrund der verschiedenen Dachtypen eine unterschiedliche Anzahl an Punkten aufweisen. Die Giebelwand eines Modells mit Satteldach benötigt sechs Punkte zum Zeichnen des Polygons. Die eines Modelles mit Mansardendach braucht acht Punkte und die Giebelwände der anderen drei Dachtypen fünf Punkte. Analog dazu werden die Seitenwände mit fünf Punkten dargestellt. Wenn die Wände durch ihre Polygonpunkte beschrieben worden sind werden sie in das CityGML eingetragen.

Als letzten Schritt werden die Dachflächen generiert. Die Anzahl der Dachpolygone hängt vom Dachtyp ab. Eine Fläche wird jeweils beim Pult- und Flachdach benötigt. Das Satteldach wird mit zwei Polygonen dargestellt. Die beiden aufwendigeren Dachtypen mit je vier Dachflächen sind das Walm- und das Mansardendach. Zum Abschluss werden die Dachflächen in die vorbereitete CityGML Datei geschrieben.

3.2.3.3 Berechnung der Polygonpunkte

Ausgehend von den Referenzkoordinaten („xRef“, „yRef“, „zRef“) werden alle Punkte des Modells berechnet. Bei der Flächenerstellung liegt der festgelegte Startpunkt immer unten links. Die Polygonpunktberechnung aller Flächen kann im Anhang nachgeschlagen werden.

Bodenfläche:

Wie im Programmablauf der Methode „createmodel“ beschrieben, wird als erstes die Bodenfläche erstellt. Der Punkt mit den Referenzkoordinaten dient dabei als Startpunkt. Danach werden die weiteren Punkte im Uhrzeigersinn generiert, indem die „BuWid“ zur „yRef“ Koordinate und die „BuLen“ zur „xRef“ Koordinate addiert werden. Der abschließende Punkt der Bodenfläche ist gleich dem Anfangspunkt, damit das Polygon geschlossen ist (Abbildung 33).

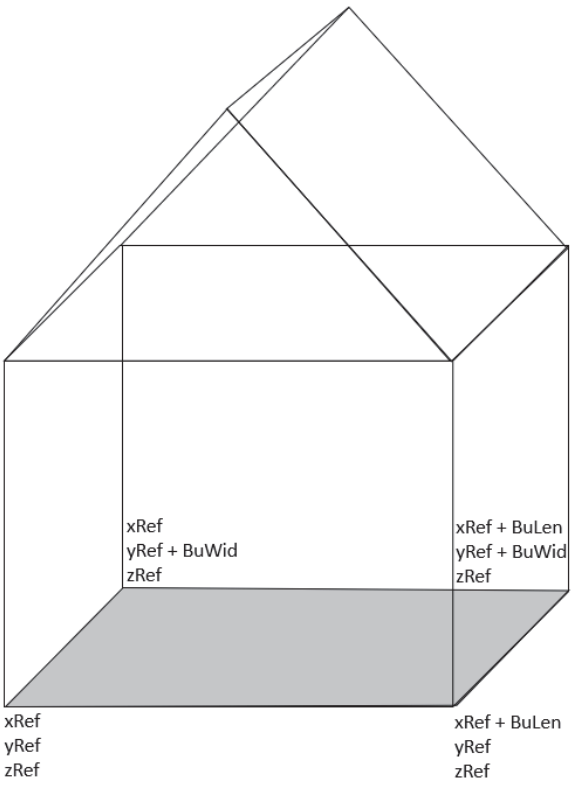


Abbildung 33: Grundflächenpolygonpunkte [23]

Wandflächen:

Im nächsten Schritt werden die vier Wände erstellt. Darunter zwei Giebelwände und zwei Seitenwände. Analog zur Punktberechnung der Bodenfläche werden die Punkte der Wandflächen im Uhrzeigersinn mit den Parametern der Länge und Breite bestimmt. Die Höhe der Wände wird durch die Addition der „zRef“ Koordinate und der „BuHe“ ermittelt. Bei den Modellen mit Flach- und Walmdach sind alle vier Wände gleich und werden mit fünf Polygonpunkten erzeugt.

Um die Giebelwand von Modellen mit Pultdach darzustellen, wird der obere Eckpunkt einer Seite mit der Firsthöhe „RiHe“ statt mit der Traufhöhe „BuHe“ berechnet. Bei der dazugehörigen Seitenwand werden beide oberen Eckpunkte mit der Firsthöhe ermittelt.

Für die Beschreibung der Giebelwände von Sattel- und Mansardendach werden zusätzliche Punkte benötigt, die mit weiteren definierten Parametern errechnet werden. Für den Giebelwandfirstpunkt wird die Länge des Gebäudes durch zwei geteilt und zur „xRef“ Koordinate addiert. Dadurch liegt er symmetrischen in der Mitte. Die Höhe des Firstpunktes erhält man durch dazuzählen der Firsthöhe „RiHe“ zur „zRef“ Koordinate. Die „yRef“ Koordinate bleibt unverändert, was zur Folge hat, dass der Firstpunkt zu der Giebelwandfläche gehört (Abbildung 34).

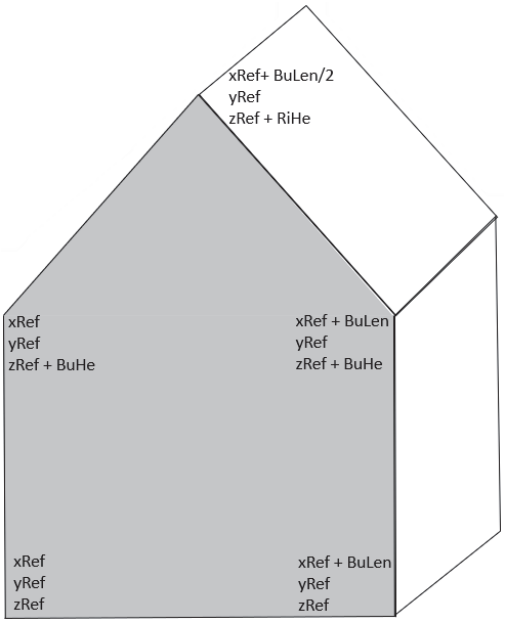


Abbildung 34: Satteldachpolygonpunkte [23]

Die komplizierteste Giebelwand entsteht aufgrund des Mansardendaches. Für die Darstellung dieser Wand ist die Berechnung von zwei weiteren Polygonpunkten notwendig. Bei diesen bleibt die „yRef“ Koordinate unverändert. Zur „zRef“ Koordinate wird bei beiden Traufhöhenpunkten, die vertikale Distanzen bis zum Knickpunkt des Daches addiert. Die x Koordinaten dieser Punkte werden mit Hilfe des Parameters „HSiTi“ errechnet, indem er für eine Seite addiert wird und für die andere Seite, nach Addition der „BuLen“ subtrahiert wird. Die im Vorfeld beschriebenen Giebelwände befinden sich auf der Vorderseite. Für die Rückseite werden die y Koordinaten der Punkte ermittelt indem die „BuWid“ zur „yRef“ dazugezählt wird. Die anderen Koordinaten ändert sich nicht.

Dachflächen:

Der letzte Punkt des Programmablaufplanes des „createmodel“ ist die Erstellung der Dachflächen. Die Dächer können aus einer, zwei oder vier Dachflächen bestehen. Die meisten Dachflächen sind Vierecke mit fünf Polygonpunkten.

Bei der Berechnung der Punkte für das Flachdach werden die Punkte für die Bodenfläche, um die Traufhöhe nach oben verschoben. Das bedeute die x und y Koordinaten der Polygonpunkte für die Bodenfläche bleiben gleich und die z Koordinate wird mit Hilfe der Traufhöhe neu berechnet (z_{Dach} = zRef + BuHe).

Die Berechnung der Punkte des Pultdachs erfolgt analog zur Berechnung des Flachdachs. Mit dem Unterschied das die Punkte auf einer Seite mit der Firsthöhe und nicht mit der Traufhöhe berechnet werden (z_{Dach} = zRef + RiHe).

Zur Darstellung der beiden Dachflächen eines Satteldaches werden die Firstpunkte als obere Polygonpunkte und die Traufhöhenpunkte als untere Polygonpunkte verwendet. (Berechnung: Abbildung 34)

Das Mansardendach besteht aus vier Dachflächen, die jeweils mit einem Polygon aus fünf Punkten dargestellt werden können. Die Berechnung der dazu benötigten Punkte wie Firstpunkt, Knickpunkt und Traufhöhenpunkt ist bereits im Abschnitt Giebelwand erklärt.

Der verbleibende Dachtyp Walmdach benötigt für die Erstellung seiner Dachflächen zwei Firstpunkte und vier Traufhöhenpunkte. Die Berechnung der Traufhöhenpunkte ist dem Abschnitt Giebelwand zu entnehmen. Um die Firstpolygonpunkte zu ermitteln werden zusätzliche Parameter genutzt. Die y Koordinaten dieser Punkte werden mit Hilfe des Parameters „HFrTi“ errechnet, indem er für eine Seite addiert wird und für die andere Seite, nach Addition der „BuWid“ subtrahiert wird. Der andere Parameter „VFrTi“ wird für die Berechnung der Höhe des Punktes gebraucht. (z_{Dach} = zRef + BuHe + VFrTi). Die dazugehörige x Koordinate wird ermittelt, indem die Länge des Gebäudes durch zwei geteilt und zur „xRef“ Koordinate dazugerechnet wird. (Abbildung 35)

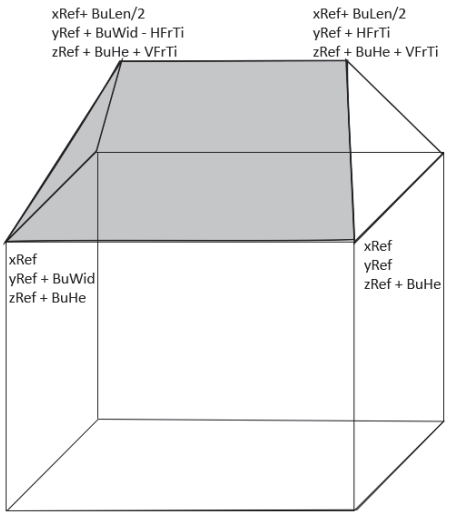


Abbildung 35: Walmdachpolygonpunkte [23]

4. Ergebnisse

4.1 Grundgebäudemodell

Um zu überprüfen, ob das nach dem Grundgebäudekonzept entwickelte Programm funktioniert, wurden die drei in Punkt 2.3 beispielhaft recherchierten Gebäudetypen als Modell dargestellt. Dazu wurden Längen und Breiten aus der Tabelle 1 verwendet und eine fiktive Höhe (von 3m je Geschoss) angenommen. Diese Parameter wurden mittels der Konfigurationsdatei dem Programm übergeben. Die generierte CityGML-Modelldatei wurde in den FZK Viewer geladen und das Gebäudemodell wurde angezeigt.

Das Grundmodell des Gebäudetyps eins (das Einfamilienhaus) ist in Abbildung 36 zusehen. Der Dachformtyp („RoSh:1“) gibt ein Satteldach vor und es wurde ein Satteldach gezeichnet, dass aus zwei Dachflächenpolygonen zusammengesetzt ist. Die Giebel- und die Seitenwand wurden entsprechend der eingegebenen Parameter dargestellt. Außerdem hat es die Grundmaße, die in der Abbildung 36 rechts in der Parameterliste stehen und eine Firsthöhe („RiHe:7“) die die Gesamthöhe von sieben Metern vorgibt.

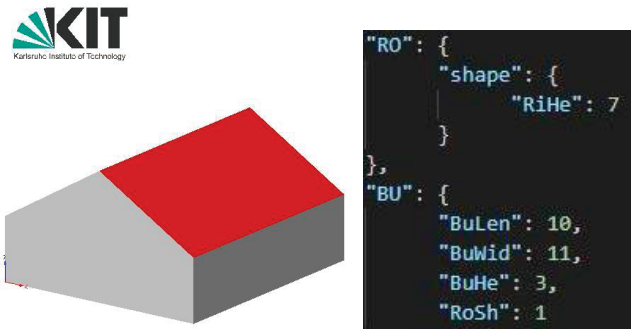


Abbildung 36: Grundgebäudemodell des Einfamilienhauses mit Parameterliste [23]

Das zweite Gebäudemodell ist das Reihenhaus. Dies ist mit einem Walmdach gedeckt und hat die Maße von 10m x 10m (siehe Abbildung 37). Aufgrund der Geschosshöhe des Gebäudemodells passen zwei Etagen hinein. Die Dachneigungsparameter sind so angegeben, dass das Walmdach minimal geneigt ist und die Gesamtgebäudehöhe acht Metern beträgt.

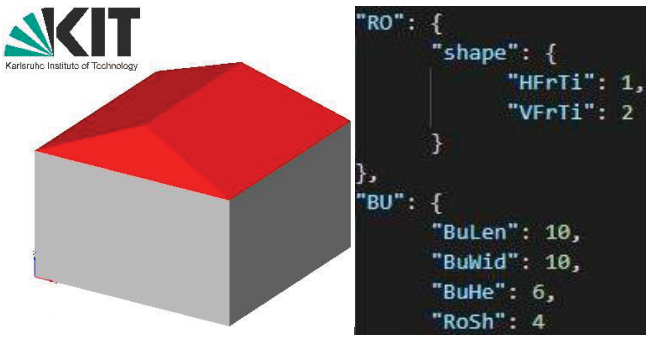


Abbildung 37: Grundgebäudemodell des Reihenhauses mit Parameterliste [23]

Das dritte Gebäudemodell ist ein großes Mehrfamilienhaus. In diesem Fall ist es ein Gebäude mit einem Flachdach und einer Höhe die acht Etagen fassen kann. Es besitzt eine Grundrissbreite und -länge von 15m x 25m (Abbildung 38).

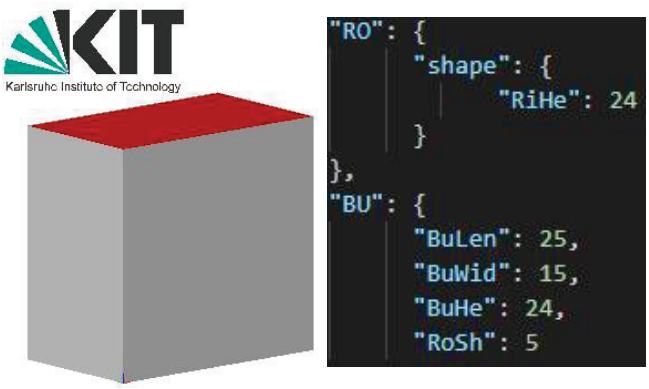


Abbildung 38: Grundgebäudemodell des großen Mehrfamilienhauses mit Parameterliste [23]

4.2 Erweitertes Grundgebäudemodell

Um die erarbeiteten Konzepte der Gebäudeaußenhülle umzusetzen, wurde das Unterprogramm „create“ durch die Entwicklungsabteilung der Firma M.O.S.S. geschrieben. Dabei wurden die vorgestellten allgemeinen Objekte 3.1.2.1 der Türen und Fenster vollständig umgesetzt.

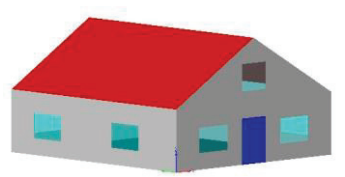
Im Weiteren wurden die Geschossobjekte 3.1.2.2 teilweise übernommen. Es wurden zwei Konzepte vorgestellt, die das Erdgeschoss beschreiben. Davon wurde nur eins bis zum Stand Dezember 2019 umgesetzt. Dieses Konzept ist das zuerst vorgestellte, welches die unabhängige Generierung von Fenster und Türen verfolgt.

Die weiteren Objektkonzepte der Balkone, Innenräume, des Treppenlochs und der Hausanschlüsse wurde in diesem Unterprogramm noch nicht bearbeitet.

Die Ergebnisse der Konzepte der Gebäudeaußenhülle sind in den Abbildung 39, 40, 41 mit ihrer dazugehörigen Konfigurationsdatei zusehen. In den drei Abbildungen wurde, die in Punkt 2.3 vorgestellten Gebäudetypen benutzt. Das Unterprogramm „create“ wurde benutzt, um die in Punkt 4.1 vorgestellten Ergebnisse der Grundgebäudemodelle, um Fenster und Türen zu erweitern.

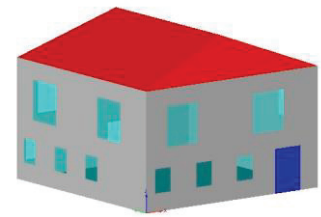
Die Abbildung 39 zeigt das Einfamilienhaus. Es hat eine Tür in der Mitte und zwei Fenster in den Geschossseiten. Das Satteldach konnte ebenfalls regelkonform dargestellt werden. Daher entsprach das Modell der Abbildung in der Publikation [11]. Außerdem konnte das Fenster im Giebel ebenfalls erzeugt werden, daher kann man von einer gelungenen Umsetzung des Realgebäudes sprechen.

Das zweite Gebäudegrundmodell, welches erweitert wurde, ist in Abbildung 40 visualisiert. Bei diesem Modell liegt die Tür am rechten Rand des Hauses. Weiterhin sind die Fenster in den Etagen unterschiedlich groß und haben unterschiedliche Abstände von den Wänden und zwischen sich selbst.



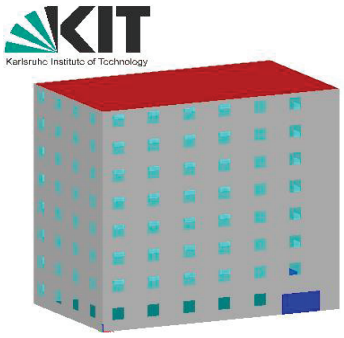

```
"GF": {
      "height": 3,
      "heightGr": 0.0,
      "window": {
            "HDistWaWi": 1.5,
            "VDistFlWi": 0.8,
            "HDistWiWi": 3.4,
            "WiLen": 1.8,
            "WiHe": 1.2,
            "HDistMinWaWi": 0.1
      },
      "door": {
            "DoHe": 2.2,
            "DoLen": 1.5,
            "HDistDoWa": 4.25
      }
},
"UF": {
      "height": 3,
      "window": {
            "HDistWaWi": 4.25,
            "VDistFlWi": 0.5,
            "HDistWiWi": 1,
            "WiLen": 1.5,
            "WiHe": 1.3,
            "HDistMinWaWi": 0.1
```


Abbildung 39: Erweitertes Einfamilienhaus mit Parameterliste [23]




```
"GF": {
      "height": 3,
      "heightGr": 0.0,
      "window": {
            "HDistWaWi": 0.5,
            "VDistFlWi": 0.9,
            "HDistWiWi": 1.5,
            "WiLen": 0.9,
            "WiHe": 1.2,
            "HDistMinWaWi": 2.5
      },
      "door": {
            "DoHe": 2.2,
            "DoLen": 1.5,
            "HDistDoWa": 1
      }
},
"UF": {
      "height": 3,
      "window": {
            "HDistWaWi": 1.2,
            "VDistFlWi": 0.1,
            "HDistWiWi": 3.2,
            "WiLen": 1.8,
            "WiHe": 2,
            "HDistMinWaWi": 0.5
```


Abbildung 40: Erweitertes Reihenhaus mit Parameterliste [23]




```
"GF": {
      "height": 3,
      "heightGr": 0.0,
      "window": {
            "HDistWaWi": 1,
            "VDistFlWi": 1.20,
            "HDistWiWi": 2.5,
            "WiLen": 1.25,
            "WiHe": 1.3,
            "HDistMinWaWi": 1
      },
      "door": {
            "DoHe": 2.2,
            "DoLen": 4,
            "HDistDoWa": 2
      }
},
"UF": {
      "height": 3,
      "window": {
            "HDistWaWi": 1,
            "VDistFlWi": 1.20,
            "HDistWiWi": 2.5,
            "WiLen": 1.25,
            "WiHe": 1.3,
            "HDistMinWaWi": 0.5
```


Abbildung 41: Erweitertes großes Mehrfamilienhaus mit Parameterliste [23]

Das rechte Modell ist das dritte vorgestellte Grundmodell, welches erweitert in der Abbildung 41 dargestellt ist. Dieses Modell hat andere Dimensionen als die vorherigen Modelle. Die Tür ist wesentlich größer und durch die 48 Fenster in der Vorderseite ist das große Mehrfamilienhaus wesentlich objektreicher als die anderen Modelle. Außerdem besitzt es ein Flachdach.

4.3 Programmtest des „ModelCreator“

Die Funktion des Unterprogramms „ModelCreator“ wurde anhand von drei grundsätzlichen Fehlerszenarien getestet. Dabei handelt es sich, um zwei verschiedene Arten von fehlerhaften Variablen in der Konfigurationsdatei und das Fehlen der Konfigurationsdatei. Um diese Fehler zu lokalisieren oder abzufangen wurden bestimmte Vorkehrungen im Programm getroffen.

Der erste erdachte Fehler ist die Eingabe eines unmöglichen Parameters wie z.B. die BuWid von (-1). Gleichbedeutend damit wäre das Vergessen des Parameters wie Abbildung 42 im Parameterfenster zu sehen ist (roter Kreis). Die Lösung des Fehlers ist, dass ein Default-Wert anstelle des Fehlenden oder unmöglichen Parameters gesetzt wird. Die Default-Werte sind so gewählt, dass sie auffallen. So war bei dem Test deutlich zu erkennen, dass etwas nicht richtig umgesetzt wurde. Im Beispiel in Abbildung 42 ist im Programm (orangener Kasten) der Default-Wert = 1 gesetzt worden. Dies konnte über den FZKViewer und der Funktion Messen bestätigt werden (grüner Kreis)

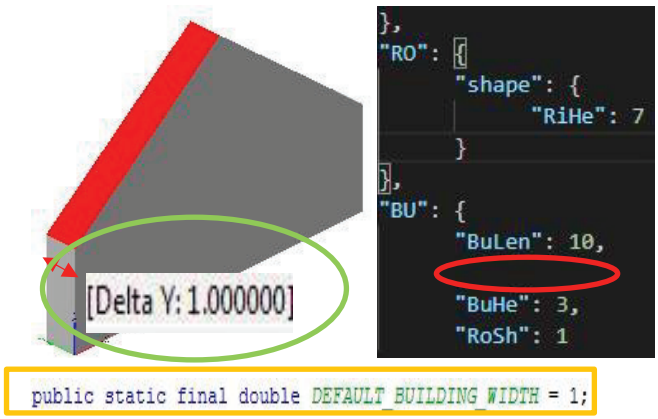


Abbildung 42: Default-Wert Fehler [23]

Ein weiterer Fehler ist das Eintragen einer ungültigen Variablen, wie z.B. ein String „wewe“ statt einer Zahl. Da bei den Parametern nur Zahlen verarbeiten werden, wird dies erkannt und die Zeile des Fehlers ausgegeben. Danach wird das Programm nicht weiter ausgeführt.

```
INFO-2020-01-24T15:21:18,867 [main] (AppMain.java:73) - nFBldModel3D  Version 0.0.2a01
INFO-2020-01-24T15:21:18,898 [main] (AppMain.java:88) - Gestartet am Fri Jan 24 15:21:18 CET 2020
INFO-2020-01-24T15:21:18,913 [main] (AppMain.java:58) - *** Erstellt das Baukoerpermodel ***
INFO-2020-01-24T15:21:19,147 [main] (ModelCreator.java:177) - Props-Datei wird gelesen: C:\work\Te
n ...

Fehler (AppException):
die angegebene Konfiguration konnte nicht geladen werden: C:\work\TestumgebungHTW_1812\JSON\Konfigu
alue of type `double` from String "wewe": not a valid Double value
 at [Source: (File); line: 37, column: 22] (through reference chain: de.moss.nf.bldmdl.props.Creati
ps.BuildingModelProperties["BuWid"])
*** ErrorCode ***
Drücken Sie eine beliebige Taste . . .
```

Abbildung 43: Ausschnitt eines Fehlers durch eine ungültige Variable [23]

Es wäre möglich bei diesem Fehler anstelle der Fehlerausschrift ein Default-Wert anzunehmen. Das Programm befindet sich jedoch in einem frühen Stadium der Entwicklung und wird noch nicht von Anwendern genutzt. Daher wurde von einer konfigurierten Fehlermeldung oder einem Default-Wert abgesehen.

Der dritte Fehler ist das Fehlen der gesamten Konfigurationsdatei. Um diesen Fehler zu erkennen gibt es eine Fehlerausschrift wie in Abbildung 44 zu sehen ist. Diese benennt den Fehler und den Pfad, wo sich die Datei befinden sollte. Das Programm wird danach beendet, da ohne Parameter das Erstellen von Gebäudemodellen nicht dem eigentlichen Sinn entspricht.

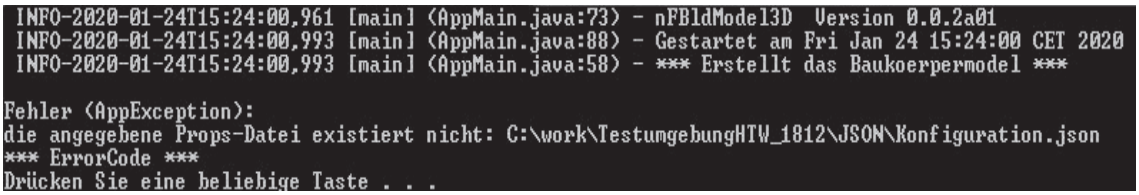


Abbildung 44: Fehlerausgabe bei fehlender Konfigurationsdatei [23]

5. Auswertung

Der Kernpunkt dieser Arbeit ist das Unterprogramm „ModelCreator“, welches als Erweiterung in das von der Firma M.O.S.S. entwickelte nfBldModel3D Programm integriert wurde. Es ist damit möglich, die in Punkt 4.1 gezeigten Grundgebäudemodelle zu erzeugen. Die dafür notwendigen Polygonpunkte für die Außenhülle konnten erfolgreich mit Hilfe der festgelegten Parameter berechnet werden. Die fünf Dachformen (Punkt 3.1.1) wurden ebenfalls mit den zugehörigen Parametern modelliert. Dadurch war es möglich die ausgewählten Gebäudetypen (Punkt 2.3) individuell mit unterschiedlichen Parameterwerten zu erstellen. Nachdem diese Grundgebäudemodelle erfolgreich erzeugt wurden, konnten sie mit dem Unterprogramm „create“ anhand der in Punkt 3.1 erläuterten Konzepte erweitert werden.

Mit dem allgemeinen Konzept des Fensters wird der Fenstertyp mit einer Höhe und einer Länge festgelegt. Dies gilt für alle Fenster im Gebäude. Eine Einschränkung dabei ist, dass pro Etage nur eine definierte Größe umgesetzt werden kann. Grundsätzlich werden die Fenster von der linken Wand aus angeordnet, solange bis kein Fenster mehr in die Wand passt. Dies kann für alle Etagen gleich umgesetzt werden, was zu einer Symmetrie zwischen den Etagen führt. Im Erdgeschoss kann es bei diesem Konzept zu einer Überschneidung mit der Tür kommen, deren Generierung von der rechten Wand mit nur drei Parametern erfolgt. Die Tür hat Vorrang, wodurch ein größerer, nicht kontrollierbarer Abstand zwischen ihr und dem nächsten Fenster entstehen kann. Um dies zu verhindern gibt es für das Erdgeschoss ein zweites Konzept. Bei diesem Konzept kann es nicht mehr zu Überschneidungen kommen, da die beiden Objekte voneinander abhängig sind. Für beide Konzepte gilt, dass nur eine Tür pro Wand möglich ist und Fenster und Tür flexibel platziert werden können (Punkt 4.2).

Weitere Objekte, mit denen das Grundgebäudemodell detailreicher dargestellt werden kann, sind Balkone, Innenräume und Hausanschlüsse. Da für diese Objekte noch keine Konzeptumsetzung vorliegt, kann noch keine Wertung erfolgen. Es ist jedoch vorstellbar, dass beim ersten Balkonkonzept die Balkonfenster nicht über dem Balkon liegen und nicht mit den Balkonboden abschließen. Bei dem zweiten Konzept ist ein Fenster direkt dem Balkon zugeordnet (Abbildung 24). Dies hat den Vorteil das zwei Fenstertypen in einer Etage dargestellt werden können.

Die parametrisierte Methode wurde so entwickelt, dass sie in vieler Hinsicht erweiterbar ist. Die drei gewählten Gebäudetypen sind nur Beispiele. Es ist möglich eine Vielzahl von Gebäudetypen zu definieren, solange diese sich voneinander unterscheiden. Außerdem können weitere Objekte wie Gauben, Loggias oder Wintergärten integriert werden. Dafür werden neue Konzepte und Parameter benötigt oder vorhandene Konzepte müssen überdacht werden. Es sollte dabei darauf geachtet werden, dass die Parameterzahl handhabbar bleibt, um die Übersichtlichkeit der Parameter für den Anwender zu gewährleisten. Neue Parameter sollten deshalb nur eingeführt werden, wenn sie nicht durch schon vorhandene berechnet werden können. Bei der Verwendung von zusätzlichen Objekten sollte je nach Anwendungsszenario abgewogen werden, ob sie generiert werden sollten oder nicht, da nicht jedes Objekt für jede Anwendung notwendig und sinnvoll ist. Viele Objekte können die Produktionszeit verlängern. Das zeigen Tests mit zwei unterschiedlichen Datensätzen. Einmal wurden 4300 Gebäudetypen mit vermehrten Einfamilienhäusern in drei min produziert. Zum anderen wurden nur 2200 Gebäude mit vielen großen Mehrfamilienhäusern und dadurch mehr Fenstern in fünf min generiert.

In dem, in Punkt 4.3 beschriebenen Test wurde überprüft, wie das Programm „ModelCreator“ auf fehlerhafte Eingaben reagiert. Bei der Eingabe unmöglicher bzw. vergessener Parameter wurde das Programm bis zum Ende ausgeführt. Das erzeugte Modell ließ klar erkennen, dass eine falsche Eingabe stattgefunden hat. Bei den anderen falschen Eingaben wurde das Programm nach der Ausgabe des Fehlers beendet. Die Fehlerausgabe dient in dieser Phase der Programmentwicklung dazu den Fehler schnell zu lokalisieren und zu beheben.

Die im Rahmen dieser Arbeit gewählten Programme und Formate haben sich als praktikabel erwiesen. Das CityGML Format ermöglichte einen strukturierten Aufbau der Gebäudemodelle. Das JSON Format lieferte aufgrund seiner objektorientieren Struktur eine schnelle Verarbeitung der Konfigurationsparameter.

Der FZKViewer ermöglichte eine interaktive 3D-Visualisierung der Gebäude. Sie waren drehbar und konnten somit von allen Seiten betrachtet werden. Die farbliche Unterscheidung der roten Dächer und grauen Wände machten die einzelnen Modellbestandteile gut sichtbar. Außerdem konnten diese auch einzeln angezeigt werden.

novaFACTORY bietet eine solide Grundsoftware zum Erstellen der 3D Gebäudemodelle. Die dazu verwendeten Teilmodule von novaFACTORY sind in der Programmiersprache Java geschrieben, welche sich sehr gut mit Netbeans bearbeiten ließ.

In dieser Arbeit wurde auf der Basis von Parametern eine neue Methode zur Erstellung von 3D-Gebäudemodellen entwickelt, mit der z.B. bestehende Gebäudemodelle aus Punktwolken mit Objekten erweitert werden können. Die Punktwolken liefern schnell generalisierte Modelle, die jedoch keine Details wie Fenster oder Türen beinhalten, da diese eine enorme Datenmenge benötigen würden. Die parametrisierte Methode hingegen erzeugt Modelle die detailreicher und typisiert sind. Dadurch ist es möglich u.a. auch Innenräume abzubilden. Dies ist notwendig, um die 3D Gebäudemodelle für anwenderspezifische, komplexe Analysen nutzbar zu machen.

Werden diese Gebäudemodelle dafür genutzt, ein bestehendes Stadtmodell zu erweitern, führt dies zu einer Erhöhung des Detailgrades (LoD2->LoD3). Es wird dabei keine reale Nachbildung erstellt, sondern nur ein detaillierteres Modell der Außenhülle. Damit ist es Anwendern aus z.B. Forschungsinstituten möglich komplexere Analysen durchzuführen. Grundsätzlich kann man sagen, dass eine Analyse genauer wird, je detaillierter das Modell ist. Die erzeugten Analyseergebnisse können zu einer besseren Veranschaulichung von Problemen bei z.B. Starkregen verwendet werden, da nun der Wassereintritt durch Fenster simuliert werden kann. Die LoD3 Gebäudemodelle sind für Visualisierungen wie z.B. im Web geeignet, um den Bürgern z.B. einen Einblick in Forschungsprojektergebnisse zu geben. Zum Schutz der Privatsphäre sollte der Detailgrad LoD4 mit einer Darstellung von Innenräumen der Öffentlichkeit nicht zugänglich sein. Allerdings sind für Katastrophenschutzbehörden Informationen zum Innenraum über Positionen von potenziellen Gefährdungspunkten wie z.B. Sicherungskästen und Feuerstellen sinnvoll.

Abschließend kann festgestellt werden, dass die generierten Gebäudemodelltypen aufgrund ihrer unterschiedlichen Parameter klar voneinander zu unterscheiden sind. Es ist daher möglich individuelle Modelle schnell und günstig zu erzeugen, was auch durch den automatischen Ablauf unterstützt wird. Der erste Schritt zur Erweiterung von Stadtmodellen wurde somit möglich gemacht.

6. Ausblick

Da das Programm sich noch im Anfangsstadium der Entwicklung befindet, sind vielfältige Erweiterungen möglich. Die vorhandenen Objektkonzepte (Balkone, Hausanschlüsse) sollten umgesetzt werden, um zu sehen, ob die Umsetzung praktikabel ist oder die Konzepte verändert werden müssen. Neue Objektkonzepte z.B. für Gauben können entwickelt und implementiert werden. Die Umsetzung dieser Objekte kann sich an bestehenden Konzepten orientieren oder nach einem anderen, modularen Ansatz erfolgen. Beim modularen Ansatz würden Modeltypen für Objekte definiert, die dann an die Positionen mit den definierten Parametern gesetzt werden. Dadurch würde es möglich werden, unterschiedliche Fenstertypen in einer Etage darzustellen.

Damit das Programm vom Anwender genutzt werden kann, muss es noch anwenderfreundlicher gestaltet werden. Dafür müssen die Programmabstütze, z.B. durch das Einsetzen von Default-Werten, verhindert werden. Außerdem sollte ein Logfile geschrieben werden in dem aufgelistet wird, welcher Fehler, wo und warum aufgetreten ist.

Wird die Entwicklung dieser Methode fortgesetzt, muss auf Formatversionsänderungen geachtet werden, damit eine stabile Nutzung durch den Anwender gewährleistet ist.

Literaturverzeichnis

[1]. **Coors, Volker, Andrae, Christane und Böhm, Karl Heinz.** *3D Stadtmodelle Konzepte und Anwendungen mit CityGML.* Berlin : VDE Verlag GmbH, 2016. ISBN: 978-3-87907-590-4.

[2]. **Tekla.** *Was ist BIM?* [Online] [Zitat vom: 30. 01 2020.] https://www.tekla.com/de/bim/was-ist-bim.

[3]. **M.O.S.S. und ibR.** *Zusammenwachsen von ALKIS und 3D Gebäuden.* s.l. : M.O.S.S Computer Grafik Systeme Geoinformationssysteme / ibR Geoinformation, 2014.

[4]. **Coors, Volker, et al.** 3D Stadtmodelle - Broschüre. [Hrsg.] InGeoForum im ZGDV e.V. und Kommission 3D Stadtmodelle der DGfK und DGPF. *Projekte im Raum.*

[5]. **M.O.S.S.** *Varianten innerhalb von novaFACTORY 3D Pro.* Taufkirchen : M.O.S.S. Computer Grafik Systeme GmbH, 2014.

[6]. **Lepik, Andreas.** *Das Architekturmodell der frühen Renaissance. Die Erfindung eines Mediums.* [Hrsg.] Bernd Evers. München : Prestel Verlag GmbH & Co, 1995. S. 10-20. ISBN: 978-3791313962.

[7]. **Bräuer, Stefanie.** Modelle von Bauwerken und baulichen/ technischen Anlagen. *Universitätssamlung in Deutschland.* [Online] [Zitat vom: 30. 12 2019.] http://www.universitaetssammlungen.de/dokumentation/vertiefendes/modelle/bauwerke

[8]. **Otepka, Johannes, et al.** Georeferenced Point Clouds: A Survey of Features and Point Cloud Management. *ISPRS International Journal of Geo-Information.* 25. Oktober 2013, S. 1038-1065.

[9]. **Madračević, Lana und Šogorić, Stjepan.** 3D Modeling From 2D Images. [Hrsg.] IEEE. *The 33rd International Convention MIPRO.* 29. July 2010.

[10]. **Saile, Johannes.** 3D-Gebäudemodelle aus Photogrammetrie- und LIDAR-Daten. *3D Stadtmodelle.org.* [Online] [Zitat vom: 05. 01 2020.] https://www.3d-stadtmodelle.org/3d-stadtmodelle-download/04_Saile_3D-Modelle_aus_Photogr_und_Lidar.pdf.

[11]. **Loga, Tobias, et al.** zweite erweiterte Auflage *Deutsche Wohngebäudetypologie, Beispielhafte Maßnahmen zur Verbesserung der Energieeffizienz von typischen Wohngebäuden.* Darmstadt : IWU Institut Wohnen und Umwelt, 2015. ISBN: 978-3-941140-47-9 .

[12]. **Bray, Tim, et al.** Extensible Markup Language (XML) 1.0 (Fifth Edition). *W3C.* [Online] 26. 11 2008. [Zitat vom: 05. 01 2020.] https://www.w3.org/TR/2008/REC-xml-20081126.

[13]. **OGC.** Geography Markup Language. *opengeospatial.org.* [Online] 2020. [Zitat vom: 05. 01 2020.] https://www.opengeospatial.org/standards/gml.

[14]. **ecma.** *The JSON Data Interchange Syntax.* Geneva : Ecma International, 2017.

[15]. **Gosling, James und McGilton, Henry.** *The Java Language Environment - A White Paper.* California : Sun Microsystems Computer Company, 1995.

[16]. **KIT.** FZKViewer. *Institut für Automation und angewandte Informatik.* [Online] 07. Juni 2016. [Zitat vom: 05. Januar 2020.] https://www.iai.kit.edu/1648.php.

[17]. Netbeans IDE. *Wikipedia.* [Online] 2019. [Zitat vom: 05. 01 2020.] https://de.wikipedia.org/wiki/NetBeans_IDE.

[18]. **WEKA.** *DIN 18065 „Gebäudetreppen – Begriffe, Messregeln, Hauptmaße“.* [Online] 2015. [Zitat vom: 30. 01 2020.] https://www.weka.de/architekten-ingenieure/din-18065/.

[19]. **News, LIDAR.** [Online] [Zitat vom: 15. 01 2020.] https://lidarnews.com/articles/mobile-mapping-systems-an-overview-and-performance-test/.

[20]. **CartouCHe.** Cartography for Swiss Higher Education. [Online] [Zitat vom: 25. 01 2020.] http://www.e-cartouche.ch/content_reg/cartouche/webservice/en/html/OGC_learningObject5.html.

[21]. **GeoRES.** www.geores.de. [Online] [Zitat vom: 25. 01 2020.] http://wiki.quality.sig3d.org/images/a/a9/Loggia-05-CityGML-LOD2-V1.gml.

[22]. **Hortonworks.** Workflow Management. [Online] [Zitat vom: 25. 01 2020.] https://docs.cloudera.com/HDPDocuments/Ambari-2.7.4.0/bk_workflow-management/content/view_xml.html.

[23]. **Schultze, Max.** Eigene Darstellungen. 2020.

[24]. **Crockford, Douglas.** Einführung in JSON. [Online] [Zitat vom: 25. 01 2020.] https://www.json.org/json-de.html.

Abkürzungsverzeichnis

BIM	Building Information Modeling
ALKIS	Amtliches Liegenschaftskataster Informationssystem
LoD	Level of Detail
GDI	Geodaten Infrastruktur
CAD	Computer-Aided Design
LIDAR	Light Detection and Ranging
EFH	Einfamilienhaus
RH	Reihenhaus
GMH	großen Mehrfamilienhaus
XML	Extensible Markup Language
W3C	World Wide Web Consortium
GML	Geography Markup Language
MathML	Mathematical Markup Language
OGC	Open Geospatial Consortium
CityGML	City Geography Markup Language
SIG3D	Special Interest Group 3D
GDI-DE	Geodateninfrastruktur Deutschlands
JSON	JavaScript Objekt Notation
IETF	Internet Engineering Task Force
JDK	Java Development Kit
JRE	Java Runtime Environment
KIT	Karlsruher Institutes für Technologie

Anhang

Tabelle 3: Parameter der Gebäudeerstellung

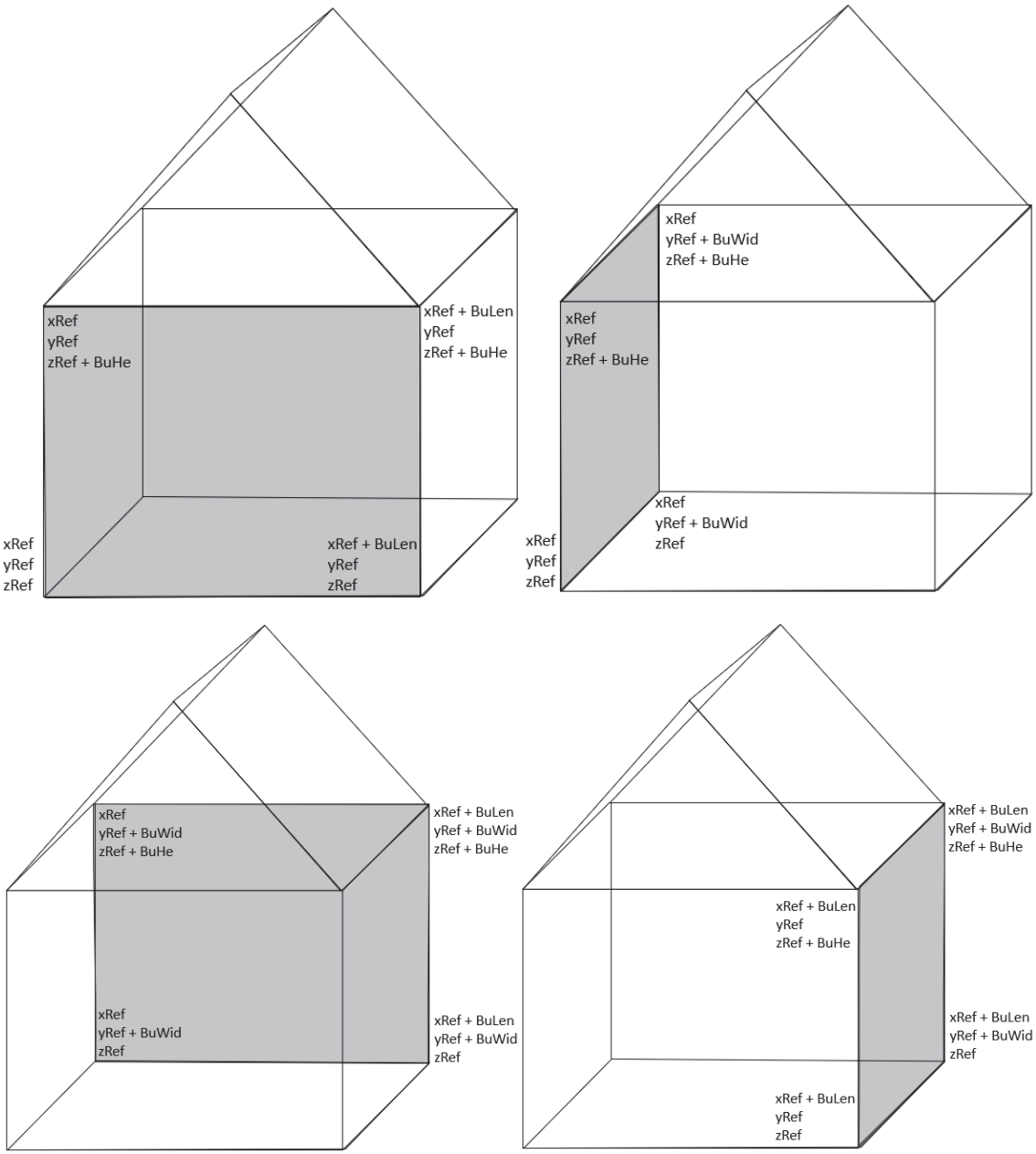
Beschreibung	**Variable im Schema in [m]**	**Legende**
Parameter für die Geschossbeschreibung		
Geschosshöhe	height	He – **he**ight
Deckenhöhe	CeHe	Ce – **ce**iling
Geschosshöhe über dem Gelände	heightGr	Gr – **gr**ound
Parameter für die Fenster Abstände		
Horizontaler Abstand Außenwand - Fenster	HDistWaWi	H – **h**orizontal
Vertikaler Abstand Decke - Fenster	VDistCeWi	Dist – **dist**ance
Horizontaler Abstand Fenster - Fenster	HDistWiWi	Wa – **wa**ll
Horizontaler, minimaler Abstand des letzten Fensters - Außenwand	HDistMinWaWi	Wi – **wi**ndow V – **v**ertical Min – **min**imal
Horizontaler Abstand Tür - Fenster	HDistDoWi	Do – **do**or
Vertikaler Abstand Fußboden - Fenster	VDistFlWi	Fl – **fl**oor
Parameter für die Fenster Beschreibung		
Fensterlänge	WiLen	Len – **len**gth
Fensterhöhe	WiHe	
Parameter für die Tür Abstände		
Horizontaler Abstand Tür - Wand	HDistDoWa	
Horizontaler minimaler Abstand Wand - Tür	HDistMinWaDo	
Horizontaler minimaler Abstand Tür - Decke	VDistMinDoCe	
Parameter für die Tür Beschreibung		
Tür Höhe	DoHe	
Tür Länge	DoLen	
Parameter für das Dach		
Firsthöhe	RiHe	Ri – **ri**dge
Horizontale seitliche Neigung	HSiTi	Si – **si**de
Vertikale seitliche Neigung	VSiTi	Ti – **ti**lt
Horizontale frontale Neigung	HFrTi	Fr – **fr**ont
Vertikale Frontale Neigung	VFrTi	

Parameter für Hausanschlüsse		
Horizontaler Abstand Wand - Punkt	HDistWaPo	Po – **po**int
Vertikaler Abstand Fußboden - Punkt	VDistFlPo	
Parameter für den Balkon		
Balkon Länge	GaLen	Ga – **ga**llery
Balkon Breite	GaWid	Wid – **wid**th
Balkon Höhe	GaHe	
Abstand des Fensters im Balkon	DistWiGa	
Parameter für die Balkon Abstände		
Horizontaler Abstand Wand zum Balkon	HDistWaGa	
Horizontaler Abstand von Balkon zum Balkon	HDistGaGa	
Horizontaler Minimaler Abstand vom Balkon zur Wand	HDistMinWaGa	
Parameter für die Innenraumaufteilung		
Horizontale Raumanzahl	HIn	In – **in**terior
Vertikaler Raumanzahl	VIn	
Minimaler Zimmergröße	MinInSz	Sz – **siz**e
Treppenloch Länge	StLen	St – **st**air
Treppenloch Breite	StWid	
Horizontaler Abstand Treppenloch Wand	HDistStWa	
Vertikaler Abstand Treppenloch Wand	VDistStWa	
Parameter zur Gebäudeerstellung		
Gebäude Länge	BuLen	Bu – **bu**ilding
Gebäude Breite	BuWid	Ro – **ro**of
Gebäude Höhe (Traufkantenhöhe)	BuHe	Sh – **sh**ape
Dachform	RoSh	

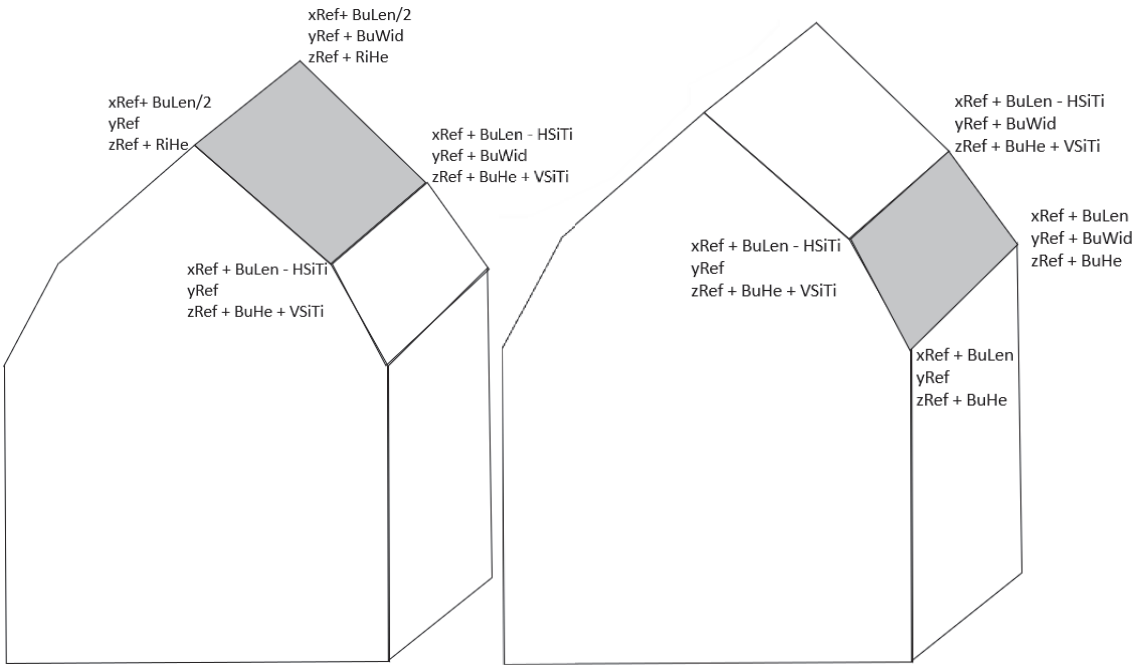
Tabelle 4: Objekte der Gebäudeerstellung

Beschreibung	**Variable**	**Legende**
Grundgebäude	BU	BU – **bu**ilding
Kellergeschoss	BA	BA – **ba**sement
Erdgeschoss	GF	GF – **g**round **f**loor
Obergeschoss	UF	UF – **u**pper **f**loor
Dach	RO	RO – **ro**of
Balkon	GA	GA – **ga**llery
Innenräume	IN	IN – **in**terior
Hausanschlüsse	UT	UT – **ut**ilities
Treppenloch	FL	FL - **fl**oor
Tür	door	
Fenster	window	
Form	shape	

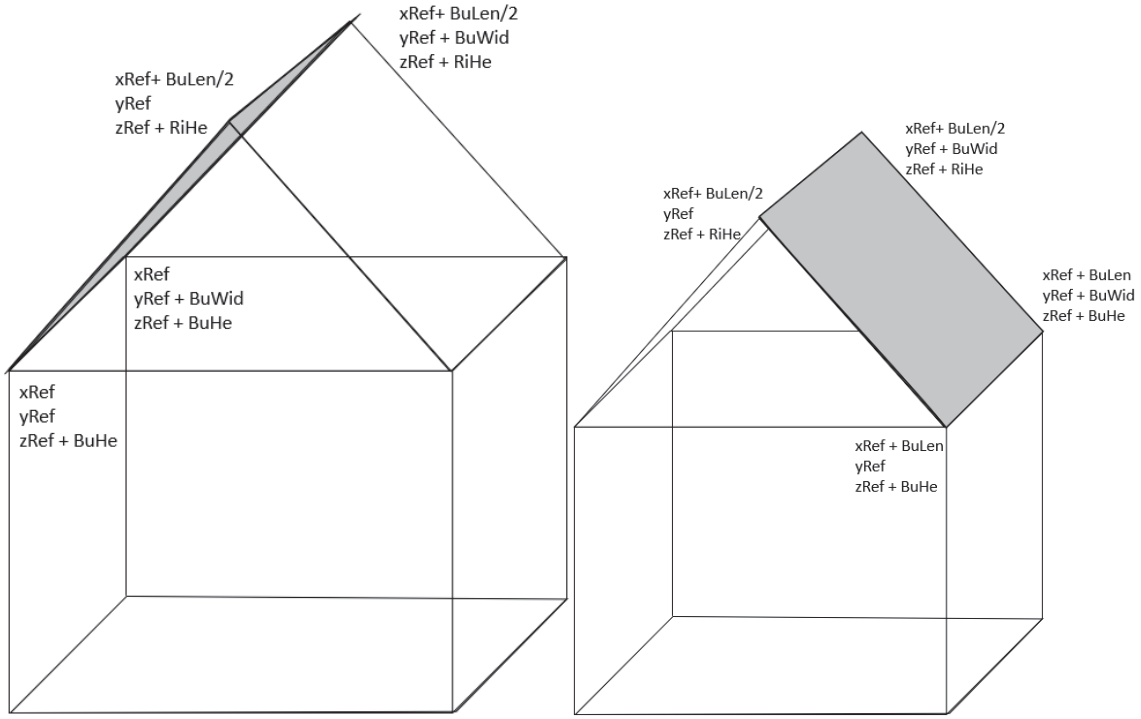
Seitenwände



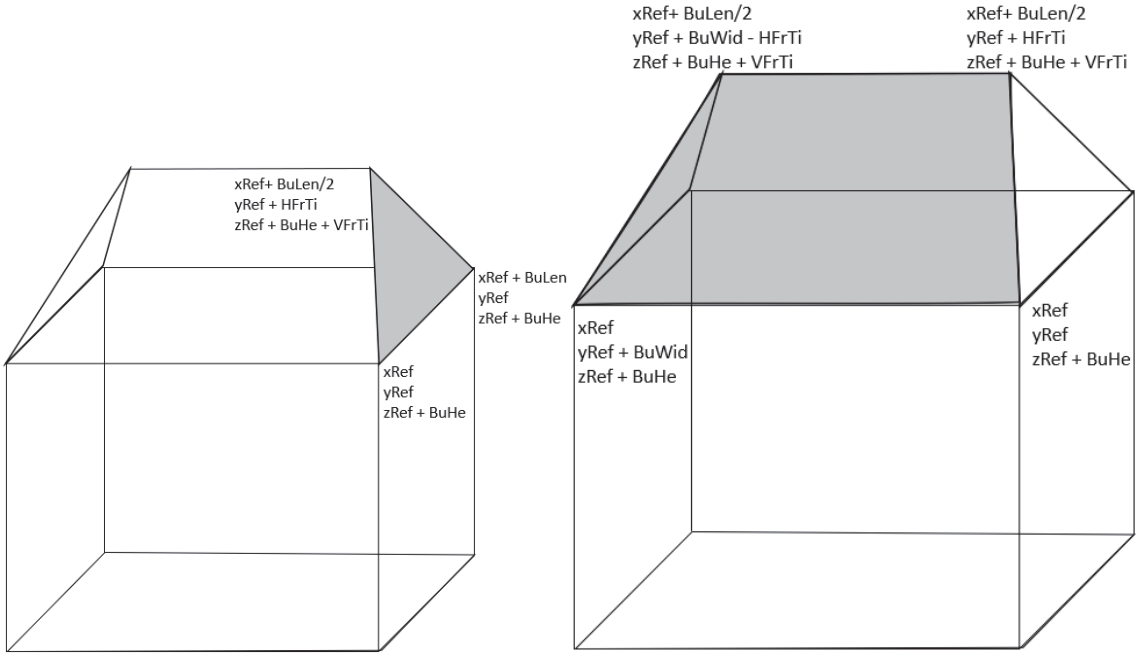
Mansardendach



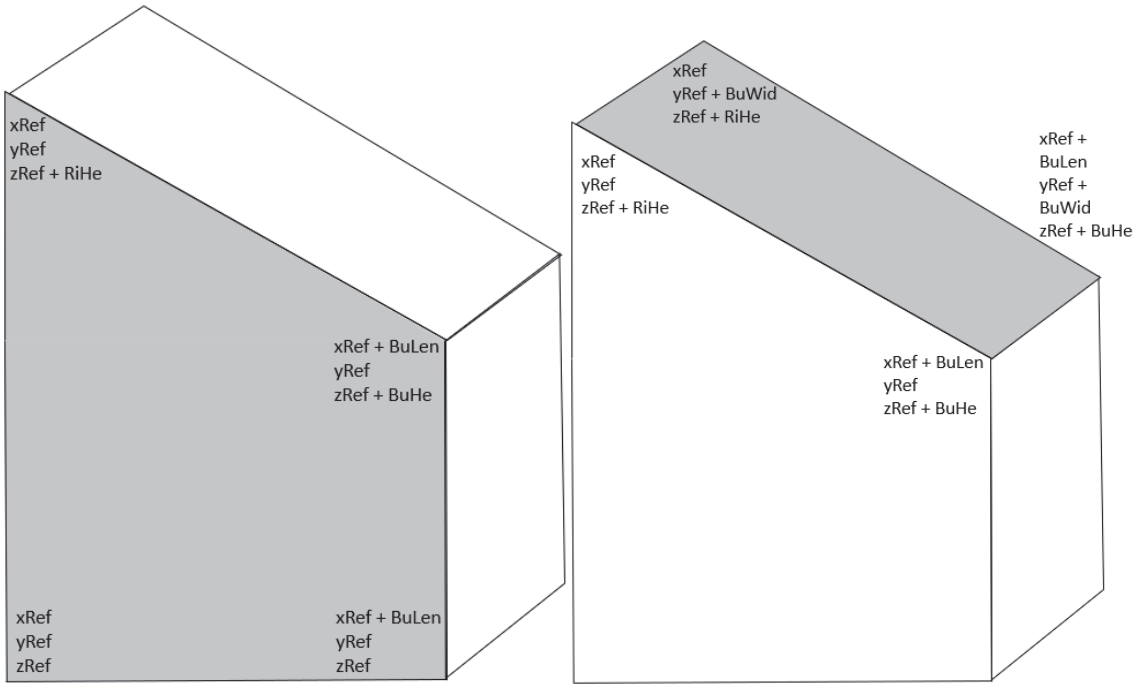
Satteldach



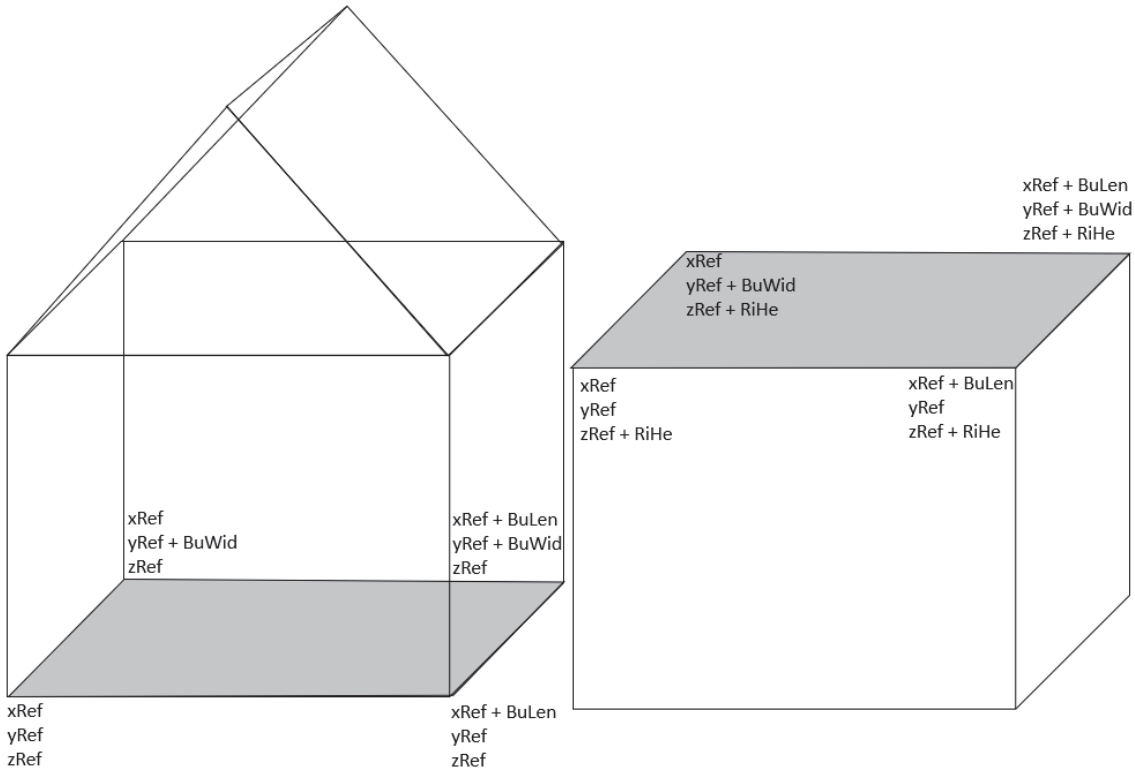
Walmdach



Pultdach



Grundfläche und Flachdach



Giebelwände Mansardendach und Satteldach