
# Bachelorarbeit

Zur Erlangung des akademischen Grades
**Bachelor of Science (B.Sc.)**

Untersuchung zur prozeduralen Modellierung urbaner Planungsszenarien in der Unreal Engine

Jade Hochschule Oldenburg
Fachbereich: Bauwesen Geoinformation Gesundheitstechnologie
Studiengang: Geoinformatik

**Eingereicht von:** Vincent-Aleister Raveling
Matrikelnummer: 6043400
Meedlandsreihe 30
26605 Aurich
E-Mail: vincent-aleister.raveling@student.jade-hs.de
Telefon: 04941 9919466

**Erarbeitet im:** 7. Semester

**Abgegeben am:** 6. Januar 2025

**Erstprüfer:** Prof. Dr. Ingrid Jaquemotte

**Zweitprüfer:** Julian Müller, B.Sc.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

**ADE** ..... Application Domain Extension

**BauGB** ..... Baugesetzbuch

**BauNVO** ..... Baunutzungsverordnung

**BIM** ..... Building Information Modelling

**BMZ** ..... Baumassenzahl

**CGA** ..... Computer Generated Architecture

**CIM** ..... City Information Modelling

**CityGML** ..... City Geography Markup Language

**COG** ..... Cloud-Optimized GeoTIFF

**DGM** ..... Digitales Geländemodell

**DHHN** ..... Deutsches Haupthöhennetz

**DOP** ..... Digitales Orthophoto

**EPSG** ..... European Petroleum Survey Group Geodesy

**ETRS** ..... Europäisches Terrestrisches Referenzsystem

**GFZ** ..... Geschossflächenzahl

**GG** ..... Grundgesetz

**GIS** ..... Geographisches Informationssystem

**GRZ** ..... Grundflächenzahl

**IFC** ..... Industry Foundation Classes

**IoT** ..... Internet of Things

**L-System** ..... Lindenmayer-System

**LGLN** ..... Landesamt für Geoinformation und Landesvermessung Niedersachsen

**LoD** ..... Level of Detail

**MCMC** ..... Markov Chain Monte Carlo

**NASA** ..... National Aeronautics and Space Administration

**OGC** ..... Open Geospatial Consortium

**PCG** ..... Procedural Content Generation

**PlanZV** ..... Planzeichenverordnung

**SUS** ..... System Usability Scale

**UTM** ..... Universale Transversale Mercatorprojektion

# 1 Einleitung

## 1.1 Motivation

Um zukünftigen Herausforderungen gewachsen zu sein, befinden sich urbane Räume in einem stetigen Wandel. Nicht zuletzt aufgrund der wachsenden Weltbevölkerung werden im Zuge der Urbanisierung fortlaufend weitere Flächen erschlossen - eine Ressource, die nur begrenzt verfügbar ist. Auch der Klimawandel und der Umgang mit Ressourcenknappheit erfordern ein gesamtheitliches Umdenken in der Planung urbaner Räume (Reicher, 2024). Über die letzten Jahrzehnte wechselte in Deutschland das städtische Leitbild von dicht bebauten Siedlungen mit autogerechtem Stadtbild zu einem nachhaltigen Bewusstsein im Städtebau (Borchard, 2018).

Gleichzeitig haben Entscheidungsprozesse in der urbanen Planung deutlich an Komplexität dazugewonnen. Solche Sachverhalte lassen sich mit herkömmlichen Plänen und Karten nur sehr abstrahiert darstellen. Um ein möglichst umfassendes Verständnis des urbanen Gebietes zum Zeitpunkt der Bestandsaufnahme und während der Entwicklung zu erhalten, ist es erforderlich, urbane Daten detailliert zu erfassen und dreidimensional abzubilden (Sabri et al., 2015). Die 3D-Modellierung urbaner Gebiete stellt nach Kolbe & Donaubauer ein interdisziplinäres Thema dar, welches Bereiche aus der Geoinformatik, der Planung und Architektur, sowie urbanen Simulationen und der Computergrafik vereint (siehe Abbildung 1) (Kolbe & Donaubauer, 2021).

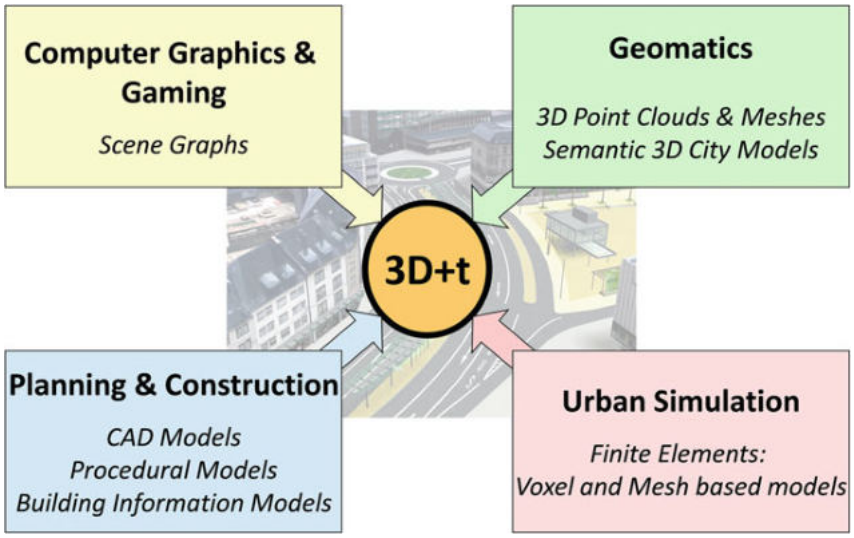


**Abbildung 1:** Verschiedene Disziplinen von urbanen 3D-Modellen (Kolbe & Donaubauer, 2021)

Bereits in der Spieleentwicklung lassen sich durch Open-Source-Engines wie der Unreal Engi-

ne, Unity und Godot, virtuelle Gebiete mit beeindruckender realistischer Grafik kreieren. Ein aktuelles Beispiel hierfür stellt das Spiel „Marvel's Spider-Man" dar, in dem der Stadtbezirk Manhattan nachmodelliert wurde (siehe Abbildung 2). Um den Entwicklungsaufwand zu erleichtern, wurden hierbei prozedurale Methoden für die Modellierung verwendet (Santiago, 2019).

Hierbei sind solche Entwicklungswerkzeuge nicht ausschließlich der Spieleentwicklung vorbehalten, sondern können auch für andere Szenarien eingesetzt werden. Beispielsweise ermöglicht die ArcGIS CityEngine vom Hersteller ESRI eine prozedurale Modellierung von urbanen Gebieten. Durch die Generierung von Straßennetzen und 3D-Geometrien für Gebäude kann die Engine für urbane Planungsszenarien eingesetzt werden (Parish & Müller, 2001). Die CityEngine lässt sich zwar über das Plugin Vitruvio in die Unreal Engine einbinden (siehe Abbildung 2), jedoch wird für den kommerziellen Nutzen eine entsprechende Lizenz benötigt. Dementsprechend mangelt es zum aktuellen Zeitpunkt an vergleichbaren Open-Source-Alternativen.

**Abbildung 2:** 3D-Modelle der Stadt New York aus „Marvel's Spider-Man" (links) (Santiago, 2019, verändert) und in der Unreal Engine mit dem Plugin Vitruvio (rechts) (ESRI, o.J. d, verändert)

Daher steht in dieser Arbeit folgende Forschungsfrage im Vordergrund:

> **Inwiefern können Game Engines für die prozedurale Generierung urbaner Planungsszenarien eingesetzt werden?**

## 1.2 Zielsetzung und Vorgehensweise

Das hauptsächliche Ziel dieser Arbeit ist es, die prozeduralen Funktionen der Unreal Engine als Game Engine für ein gewähltes Planungsszenario einzusetzen und zu evaluieren. Durch die Erstellung eines Prototyps wird untersucht, welche Chancen und Limitationen bei der Implementierung auftreten. Hierbei soll auf Grundlage eines Bebauungsplans ein 3D-Modell erstellt werden. Dabei soll ermittelt werden, welche Parameter und Algorithmen für die Anwendung

erforderlich sind, um die gesetzlichen Vorgaben der Bauleitplanung annähernd einzuhalten und den Anforderungen vergleichbarer Modelle gerecht zu werden.

Ein weiteres Ziel besteht darin, eine eigenständige Anwendung zu erstellen, die unabhängig von der Entwicklungsumgebung in Echtzeit funktioniert. Hierbei soll die Gebrauchstauglichkeit der Anwendung von den charakteristischen Funktionen einer Game Engine profitieren. Durch die Implementierung eines beweglichen Avatars soll sich an der Steuerung von modernen Computerspielen orientiert werden. Außerdem soll, zur Steuerung der Parameter und Interaktion mit der virtuellen Umgebung, eine Benutzeroberfläche zur Verfügung stehen.

## 1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in insgesamt sechs Kapitel.

Nach der Einleitung beschäftigt sich das zweite Kapitel mit allen Grundlagen, die für das Verständnis dieser Arbeit erforderlich sind. Zunächst wird beschrieben, inwiefern die Bauleitplanung in Deutschland gesetzlich verankert ist, durch welche Parameter die bauliche Nutzung beschrieben wird und welche Vorgaben bei der Errichtung von Gebäuden auf Baugrundstücken einzuhalten sind. Anschließend werden, vom 3D-Stadtmodell bis hin zum urbanen digitalen Zwilling, moderne Modellierungsansätze für urbane Gebiete vorgestellt und Anwendungsbeispiele gezeigt. Hierbei wird erklärt welche Algorithmen in der prozeduralen Modellierung eingesetzt werden. Zudem gibt es einen Einblick in die Unreal Engine als Entwicklungsumgebung.

Im dritten bis fünften Kapitel wird der eigens erstellte Prototyp präsentiert. Das dritte Kapitel beschäftigt sich mit der Konzeptionierung der Arbeit. Zunächst werden die Zielgruppen und daraus resultierenden Anforderungen aufgestellt, welche dieser Prototyp erfüllen muss. Anschließend werden die Rahmenbedingungen bei der Erstellung des Prototyps beschrieben. Außerdem gibt es einen Überblick über die verwendeten Datengrundlagen, sowie deren Aufbereitung und Integration.

Im vierten Kapitel wird die Erstellung des Prototyps in der Unreal Engine vorgestellt. Es wird zunächst erklärt aus welchen Bestandteilen sich dieser Prototyp zusammensetzt. Außerdem wird beschrieben inwiefern prozedurale Methoden für die Generierung des Siedlungsgebiets und der Gebäude verwendet wurden und wie die Benutzeroberfläche und Navigation konzipiert wurden.

Das fünfte Kapitel stellt die Ergebnisse vor und beschreibt die Evaluation des Prototyps. Zunächst wird erläutert in welcher Form die Usability-Tests verlaufen sind und welche Kriterien in einer Befragung aufgestellt wurden. Es wird die Methodik der Evaluation anhand eines standardisierten Tests erklärt und die anschließende Punktevergabe, sowie Bewertung gezeigt. Danach werden die Ergebnisse der Evaluation interpretiert und der Prototyp anhand dessen bewertet.

Im sechsten und letzten Kapitel erfolgt eine Einordnung dieser Arbeit in den Gesamtkontext, sowie einen Ausblick auf zukünftige Trends.

# 2 Grundlagen

## 2.1 Bauleitplanung

Nach dem Grundgesetz (kurz: GG) sind die Kommunen in Deutschland zur Selbstverantwortung verpflichtet und somit befähigt, im politischen Entscheidungsprozess die Entwicklung im kommunalen Raum mitzugestalten (GG, 2024, Art. 28 Abs. 2). Dabei hat die Stadtplanung sich an die Ziele der Länder und Landkreise anzupassen (BauGB, 2023, § 1 Abs. 4). Dennoch besitzt die Kommune die Planungshoheit und trägt durch Mitwirkung an der Bauleitplanung zur gesamträumlichen Entwicklung bei (ROG, 2023, § 1 Abs. 3).

Die Bauleitplanung wird durch das Baugesetzbuch (kurz: BauGB) geregelt und dient zur Leitung der baulichen und sonstigen Nutzung von Grundstücken in der Gemeinde (BauGB, 2023, § 1 Abs. 1). Als planerisches Mittel stehen der Flächennutzungsplan und der Bebauungsplan zur Verfügung (BauGB, 2023, § 1 Abs. 2). Letzterer konkretisiert die Inhalte aus dem Flächennutzungsplan, die rechtsverbindlich in Kraft treten (BauGB, 2023, § 8 Abs. 1). Die Planinhalte werden schriftlich und zeichnerisch unter Berücksichtigung der Planzeichenverordnung (kurz: PlanZV) auf Planunterlagen eingezeichnet (PlanZV, 2021, §2 Abs. 1).

### 2.1.1 Maße der baulichen Nutzung

Die festgesetzten Inhalte des Bebauungsplans werden in der Baunutzungsverordnung (kurz: BauNVO) weiter konkretisiert (BauNVO, 2023, § 9). Dort wird unter anderem die Art der baulichen Nutzung ausführlicher beschrieben und die Maße der baulichen Nutzung bestimmt. Zu den Maßen gehören die Grundflächenzahl (kurz: GRZ), die Geschossflächenzahl (kurz: GFZ), die Baumassenzahl (kurz: BMZ) und die Zahl der Vollgeschosse, sowie die Höhe baulicher

Anlagen (BauNVO, 2023, § 16 Abs. 2). Die GRZ gibt das Verhältnis zwischen der zulässigen Grundfläche der Gebäude und der tatsächlich vorhandenen Grundstücksfläche an. Das Verhältnis darf hierbei nicht überschritten werden (BauNVO, 2023, § 19 Abs. 1). Selbiges gilt auch für die GFZ. Hierbei ist das Verhältnis aller Geschossflächen und der tatsächlich vorhandenen Grundstücksfläche zu berücksichtigen (BauNVO, 2023, § 20 Abs. 2).

$$\text{GRZ} = \frac{\text{Grundfläche}}{\text{Grundstücksfläche}} \tag{1}$$

$$\text{GFZ} = \frac{\text{Geschossfläche gesamt}}{\text{Grundstücksfläche}} \tag{2}$$

Um die Höhe von baulichen Anlagen in einheitlicher Form zu beschreiben werden im Bebauungsplan Bezugspunkte definiert (BauNVO, 2023, § 18 Abs. 1). Der untere Bezugspunkt bezieht sich dabei oft auf die Niveauhöhe einer anbaufähigen Verkehrsfläche, beispielsweise eines Gehwegs. Als obere Bezugspunkte werden oft die Trauf- und Firsthöhe festgelegt. Dabei bezieht sich die Traufhöhe auf den Abstand zwischen dem unteren Bezugspunkt und dem Schnittpunkt zwischen Außenwand und Dachfläche. Die Firsthöhe bezieht sich hingegen auf den Dachfirst (siehe Abbildung 3) (Stühler & Schimpfermann, 2023).

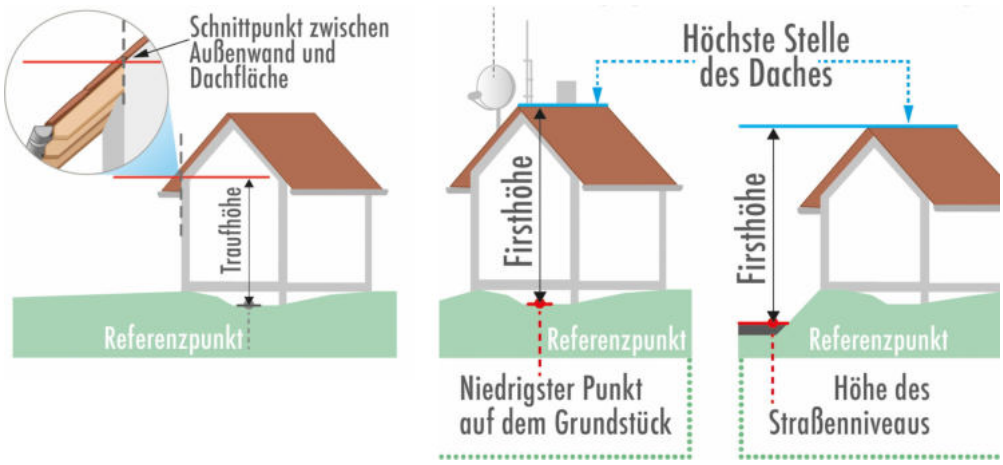


**Abbildung 3:** Definitionen von Traufhöhe (links) und Firsthöhe (rechts) (Sanier, 2024, verändert)

### 2.1.2 Überbaubare Grundstücksfläche und Bauweise

Auch wird die Überbaubarkeit von Grundstücken durch Baugrenzen und Baulinien festgelegt. Die Baugrenze gibt vor, dass Gebäude und Bestandteile davon nicht außerhalb dieser Grenze bebaut werden dürfen. Bei der Baulinie muss das Gebäude ohne Überbauung entlang der Linie errichtet werden (siehe Abbildung 4) (BauNVO, 2023, § 23).

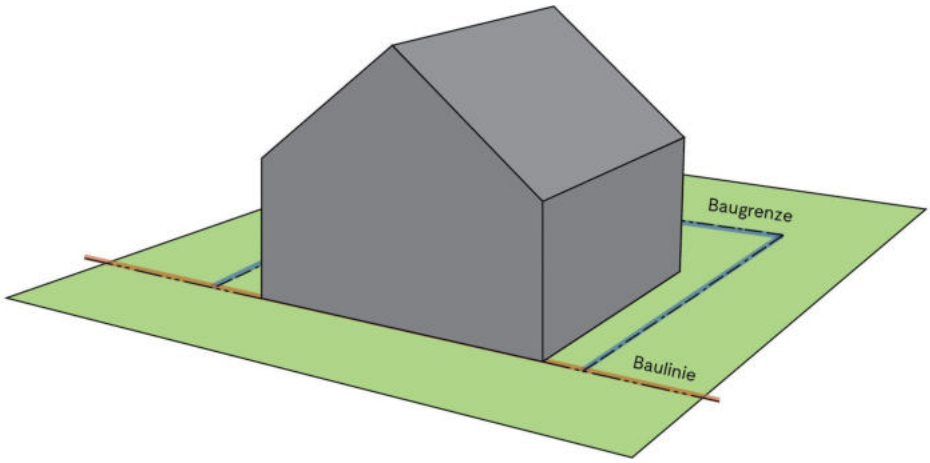


**Abbildung 4:** Übersicht zur Wirksamkeit von Baugrenzen und Baulinien (Völkl, o.J.)

Die Bauweise kann in Bebauungsplänen entweder als offen, geschlossen oder abweichend festgesetzt werden. Während bei der offenen Bauweise ein festgelegter Abstand zur Grenze bestehen muss, werden bei der geschlossenen Bauweise Gebäude ohne seitlichen Grenzabstand errichtet (siehe Abbildung 5). Auch können darüber hinaus abweichende Regelungen getroffen werden (BauNVO, 2023, § 22).

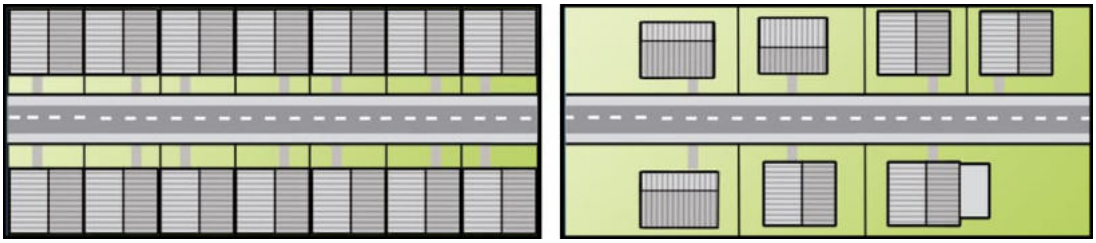

**Abbildung 5:** Übersicht geschlossene (links) und offene Bauweise (rechts) (Wagner, 2024, verändert)

### 2.1.3 Öffentlichkeitsbeteiligung und Digitalisierung

Im Rahmen der Bauleitplanung hat die Öffentlichkeit zweimal die Gelegenheit, sich am Verfahren zu beteiligen. Während der frühzeitigen Beteiligung können verschiedene Planungskonzepte gegenübergestellt werden (BauGB, 2023, §3 Abs. 1). Die Entwürfe von Bauleitplänen sind im Rahmen der öffentlichen Auslegung zu veröffentlichen, sodass die Öffentlichkeit durch die Äußerung von Stellungnahmen aktiv am Entscheidungsprozess mitwirken kann (BauGB, 2023, §3 Abs. 2).

Um den Datenaustausch zwischen den beteiligten Akteuren zu erleichtern kommt in Deutschland das Austauschformat XPlanGML zum Einsatz, dessen Struktur im Datenstandard XPlanung festgelegt wird. Hierdurch werden die Inhalte aller relevanten Raumpläne, darunter der

Bebauungsplan, unter Berücksichtigung gesetzlicher Vorgaben als digitale Klassen repräsentiert (XLeitstelle, 2023). Die digitale Repräsentation dieser Inhalte kann für die Integration in andere Anwendungen von wertvollem Nutzen sein.

Ein Framework von Zahid et al. zeigt wie Plandaten zukünftig in 3D-Stadtmodelle integriert werden könnten. Durch eine Kombination aus XPlanung und der City Geography Markup Language (kurz: CityGML) können die Planinhalte mit dreidimensionalen Repräsentationen von Gebäuden verknüpft werden. Durch einen Flächenvergleich kann evaluiert werden, ob die GRZ oder GFZ überschritten wurde. Auch kann die topologische Beziehung zwischen der Baulinie oder der Baugrenze und dem geplanten Gebäudemodell verglichen werden (siehe Abbildung 6) (Zahid et al., 2024).

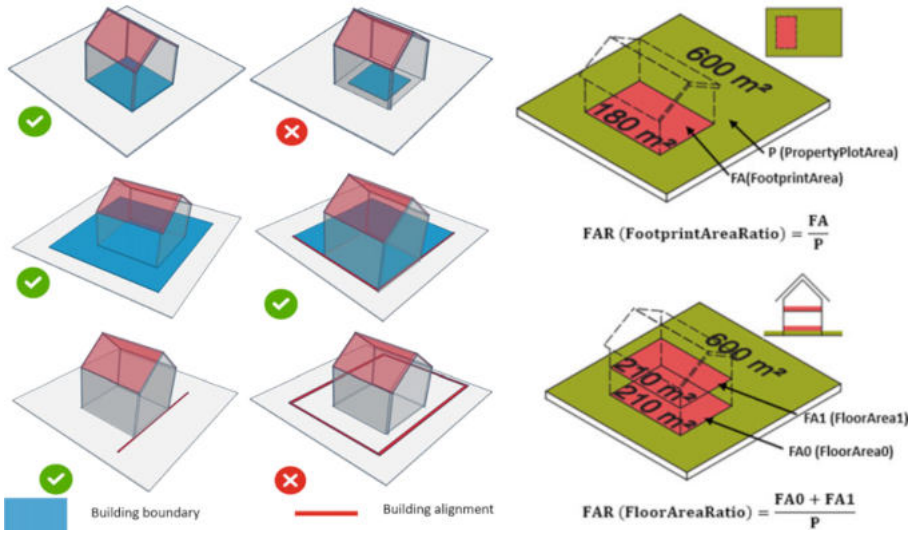


**Abbildung 6:** Einschränkungsregeln zur Baulinie und Baugrenze (links), sowie zur GRZ und GFZ (rechts) (Zahid et al., 2024, verändert)

## 2.2 Digitale Stadtmodellierung

Bei der Aufstellung von Bauleitplänen ist die Kommune als Entscheidungsträger verpflichtet, verschiedene Belange gerecht abzuwägen (BauGB, 2023, § 1 Abs. 7). Als Entscheidungshilfe bietet es sich an, den urbanen Raum möglichst realistisch und den planerischen Vorstellungen entsprechend abzubilden. Hierbei spielt die Digitalisierung in der Stadtplanung eine zunehmend wichtigere Rolle. Mittels computergestützter Visualisierung und Simulationen kann der Planungsprozess für alle Beteiligten transparenter und nachvollziehbarer gestaltet werden (Stepper & Wietzel, 2012). Besonders im Rahmen der Öffentlichkeitsbeteiligung können komplexe Zusammenhänge für fachfremde Akteure zugänglicher gemacht werden.

### 2.2.1 3D-Stadtmodelle

Bei der Modellierung von Städten kommen dreidimensionelle Stadtmodelle zunehmend zum Einsatz. Grundsätzlich stellt das 3D-Stadtmodell eine computergestützte Darstellung aller urbanen Objekte dar. Dazu gehören unter anderem Gebäude, Vegetation, Landnutzung, Verkehrsnetze, sowie Wasserwege (Ranzinger & Gleixner, 1997).

Für die Erstellung eines 3D-Stadtmodells werden Geodaten aus unterschiedlichen Quellen miteinander kombiniert. Je nach Anwendungszweck können die zur Modellierung benötigten Datenquellen und Anforderungen zur Detailgenauigkeit des Modelles voneinander abweichen (Jahnke et al., 2011). Als Datengrundlagen dienen Vektor- und Rasterdaten welche zumeist in einer 2D- oder 2,5D-Darstellung vorliegen. Häufig dazu gehören digitale Geländemodelle (kurz: DGM), Karten, digitale Orthophotos (kurz: DOP) oder Satellitenbilder, ergänzt durch Gebäudemodelle (Lancelle & Fellner, 2004). Die 3D-Modellierung von Gebäuden kann durch verschiedene Methoden erfolgen, beispielsweise durch geometrische Gebäuderekonstruktion auf Grundlage von Laserscan-Daten und Luftbildern. Alternativ dazu können Gebäude prozedural modelliert werden (Jaquemotte, 2014).

Bei 3D-Stadtmodellen wird zwischen geometrischen und semantischen Modellen unterschieden. Grundsätzlich werden urbane Objekte in einem Stadtmodell durch dreidimensionale Geometrien repräsentiert. In semantischen Modellen werden zusätzlich dazu thematische und topologische Informationen über das urbane Gebiet verwaltet und liegen strukturiert vor (Billen et al., 2014). Die urbanen Objekte werden in vorgefertigten Klassen, einschließlich ihrer Attribute, definiert und stehen in Relation zueinander.

Damit solche Modelle in homogener Form spezifiziert und interoperabel ausgetauscht werden können ist eine Standardisierung erforderlich. Hierfür hat sich CityGML vom Open Geospartial Constortium (kurz: OGC) als internationaler Standard etabliert (Kolbe & Donaubauer, 2021). Darüber hinaus hat CityGML den Vorteil, urbane Objekte in unterschiedlichen Detailgraden zu verwalten. Diese werden als Level of Detail (kurz: LoD) bezeichnet und reichen von einfachen Grundrissen (LoD0) oder Klötzchenmodellen (LoD1) mit steigender Detailstufe zu Modellen mit einfachen (LoD2) oder komplexen Dachformen (LoD3) (siehe Abbildung 7) (Benner et al., 2010).

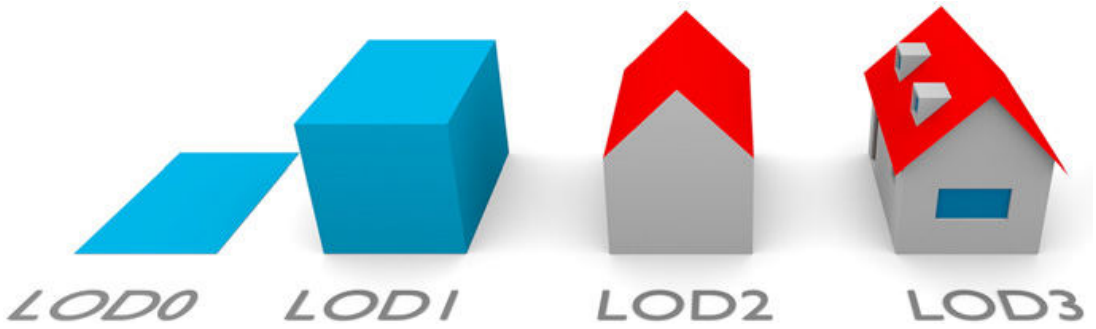


**Abbildung 7:** Übersicht der verschiedenen Level of Detail (Biljecki et al., 2016, verändert)

Seit der Einführung von CityGML 3.0 werden auch die Innenräume in verschiedenen LoD repräsentiert. Eine weitere Neuerung von CityGML 3.0 stellt die Versionierung von temporalen Veränderungen des 3D-Stadtmodells dar. Dabei können bauliche Veränderungen mit einem Zeitstempel dokumentiert werden (Kutzner et al., 2020).

Das CityGML-Format kann darüber hinaus durch Application Domain Extensions (kurz: ADE) semantisch erweitert werden. Hierdurch kann das 3D-Stadtmodell für speziellere Anwendungsfälle durch weitere Attribute und Objektarten ergänzt werden. Beispielsweise lassen sich durch die UtilityNetworkADE Informationen zu verschiedenen Versorgungsleitungen einbinden (Becker et al., 2011).

### 2.2.2 City Information Modelling

In den Bereichen Architektur, Ingenieur- und Bauwesen wird hingegen für die detaillierte Modellierung von Gebäuden das Building Information Modelling (kurz: BIM) praktiziert. Mithilfe von BIM werden Gebäude im Laufe ihres Lebenszyklus geplant, analysiert und überwacht (Sacks et al., 2018). Durch ein eigenes semantisches Datenmodell wird die Gebäudearchitektur hochdetailliert modelliert wobei Industry Foundation Classes (kurz: IFC) als internationaler Datenstandard zum Einsatz kommt. Im Gegensatz zu CityGML liegen die Daten ausschließlich in einer Detailstufe vor. Zudem werden ausschließlich Bauwerke als solches repräsentiert (Gröger & Plümer, 2012).

Um das gesamte Stadtbild noch realitätsgetreuer abbilden zu können werden Daten aus BIM und geographischen Informationssystemen (kurz: GIS) zusammengeführt. Hierdurch ist der Begriff City Information Modelling (kurz: CIM) entstanden. Darüber hinaus kann das CIM auch dynamische Objekte enthalten. (X. Xu et al., 2014). Durch die Einbindung des Internet of Things (kurz: IoT) können dynamische Daten in die digitale Modellumgebung einfließen

(Z. Xu et al., 2021). Bedingt durch die digitale Transformation von Städten durch Smart City Konzepte nimmt die Verfügbarkeit solcher urbanen Daten stetig zu. Dies führt dazu, dass die Entwicklung von digitalen Zwillingen im urbanen Raum stark durch das CIM vorangetrieben wurde (Omrany et al., 2023). Inzwischen stehen in der Literatur beide Begriffe in einem engen Zusammenhang zueinander (Jeddoub et al., 2023).

### 2.2.3 Urbaner Digitaler Zwilling

Die Grundidee eines Zwillings besteht darin, ein möglichst identisches Spiegelbild von Objekten zu entwerfen. Die ersten Zwillinge kamen bereits durch die NASA im Apollo-Programm zum Einsatz. Damals wurden zwei identische Raumfahrzeuge hergestellt, von denen das auf der Erde verbliebene Fahrzeug zur Simulation der Zustände des im Weltraum befindlichen Fahrzeuges diente (Boschert & Rosen, 2016). Später war der Begriff Digital Twin (dt. Digitaler Zwilling) für die Raumfahrt geboren (Glaessgen & Stargel, 2012). Eine weitere Definition von Tharma et al. verallgemeinert den Begriff, wobei durch ein virtuelles Spiegelbild die Eigenschaften eines beliebigen Produkts in seinem Lebenszyklus abgebildet werden und Informationen über das Produkt abgerufen werden können (Tharma et al., 2018).

Zur Veranschaulichung eines digitalen Zwillings entwarf Grieves ein Modell, welches den realen und den virtuellen Raum gegenüberstellt (siehe Abbildung 8). Der reale Raum enthält Objekte aus der realen Welt, deren Daten im virtuellen Raum als digitaler Zwilling widergespiegelt werden. Umgekehrt stellt der digitale Zwilling Informationen zur Verfügung, aus denen sich Rückschlüsse auf die Objekte der realen Welt ziehen lassen (Grieves, 2022).

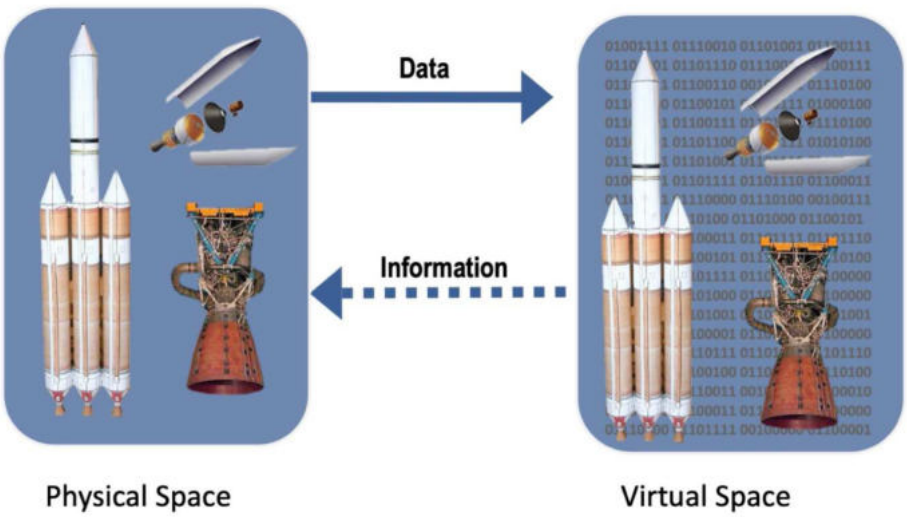


**Abbildung 8:** Modell zum Aufbau eines digitalen Zwillings (Grieves, 2022, verändert)

Auch wenn eine perfekte Kopie technisch niemals umsetzbar sein wird hat der digitale Zwilling dennoch den Anspruch dem Idealbild möglichst nahe zu kommen (Batty, 2018). Bei einem digitalen Zwilling ist der stetige Informationsaustausch zwischen den realen und digitalen Objekten hervorzuheben. Kritzinger et al. unterscheiden die Entwicklung des digitalen Zwillings in drei Ebenen (siehe Abbildung 9). Findet kein automatisierter Austausch zwischen beiden statt spricht man von einem Digital Model (dt. digitales Modell). Bei einem automatisierten Datenfluss vom realen Objekt zur digitalen Repräsentation ohne Rückfluss ist hingegen von einem Digital Shadow (dt. digitaler Schatten) die Rede (Kritzinger et al., 2018).

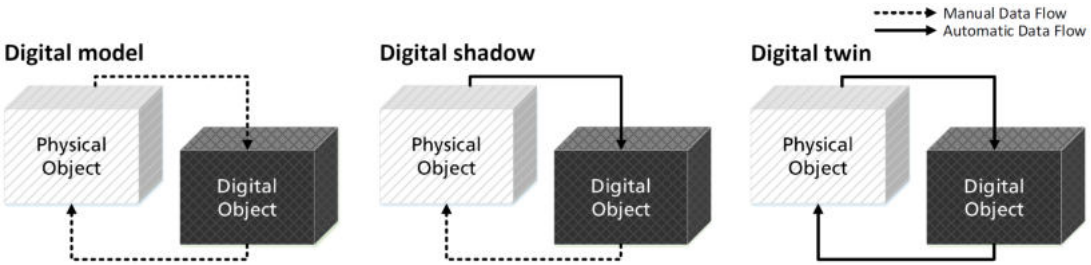


**Abbildung 9:** Ebenen zur Entwicklung eines digitalen Zwillings (Kritzinger et al., 2018, verändert)

Digitale Zwillinge finden bereits in verschiedenen Fachbereichen ihre Anwendung, darunter in der Prozessindustrie, im Produktdesign, im Structural Health Monitoring, im Recycling und in der Landwirtschaft (Ferré-Bigorra et al., 2022). Auch in der urbanen Planung und Modellierung gewinnt die Erstellung eines digitalen Zwillings zunehmend an Bedeutung (Somanath et al., 2023).

Für die Erstellung digitaler Zwillinge ganzer Städte ist der Begriff urbaner digitaler Zwilling entstanden. Dieser stellt ein semantisch erweitertes 3D-Stadtmodell dar, das mit der realen Stadt gekoppelt ist. Durch die regelmäßige Übertragung historischer Daten und Sensordaten in Echtzeit lassen sich, durch Analysen und Simulationen, fundierte Vorhersagen über die Stadt treffen (Jeddoub et al., 2023). Dabei gewinnt auch die Integration künstlicher Intelligenz in solche Entscheidungsprozesse zunehmend an Bedeutung (Matei & Cocoșatu, 2024). Die durch den urbanen digitalen Zwilling unterstützten Maßnahmen fließen als Feedback in nachfolgende Analysen und Modellierungen zurück (Mazzetto, 2024). Davon profitiert insbesondere die Stadtplanung, da Planungs- und Bauvorhaben im Laufe ihres Lebenszyklus überwacht und optimiert werden können. Schließlich unterliegen urbane Räume einem ständigen Wandel, wodurch herkömmliche 3D-Stadtmodelle schnell veralten und für Entscheidungsprozesse an Relevanz verlieren.

Die Visualisierung des urbanen digitalen Zwillings erfolgt dabei über webbasierte Plattformen oder durch Game Engines (Jeddoub et al., 2023). Game Engines bieten dabei eine vergleichsweise hohe Qualität in der Grafik und sind interaktiver gestaltet. Die Integration von Geodaten und Echtzeitdaten erwies sich jedoch bislang als schwierig und ließ sich nur ansatzweise durch Plugins umsetzen (Rantanen et al., 2023).

### 2.2.4 Anwendungen in der Stadtplanung

Für die Umsetzung von urbanen digitalen Zwillingen liegen für den Anwendungsbereich der Stadtplanung einige aktuelle Beispiele vor. Inzwischen wird eine Vielzahl größerer Städte durch einen urbanen digitalen Zwilling repräsentiert, wobei die Qualität hinsichtlich der Datengrundlagen und -integration, sowie der Funktionen zur Simulation und Analyse stark voneinander variieren (Caprari et al., 2022). Ein Vergleich von Jeddoub et al. zeigt, dass derzeit nur bei wenigen Umsetzungen die Datenintegration eines digitalen Zwillings erreicht wurde (Jeddoub et al., 2023).

Ein Vorreiter stellt der urbane digitale Zwilling aus Singapur dar, in welchem sowohl Gebäude, als auch Bäume in LoD3 repräsentiert werden. Dabei stellt das Projekt Virtual Singapore eine Vielzahl von Analyse- und Simulationswerkzeugen zur Verfügung (Caprari et al., 2022). Um gezielte Maßnahmen zur Klimaanpassung treffen zu können lassen sich im digitalen Zwilling Energiemodelle aus integrierten BIM-Daten erstellen und damit städtische Wärmeinseln identifizieren (Ignatius et al., 2019).

In dem Projekt Connected Urban Twins werden in Kooperation der Städte Hamburg, Leipzig und München verschiedene Anwendungsfälle zur Stadtentwicklung erprobt und die daraus gewonnenen Erkenntnisse miteinander ausgetauscht (Connected Urban Twins, o.J.). In diesem Rahmen wurde der 3DProjektplaner als Erweiterung für das Masterportal entwickelt. Bei dem Masterportal handelt es sich um ein Geoportal für die Stadt Hamburg, welches als Open Source verfügbar ist (Winter, o.J.). Im 3DProjektplaner liegt als Datengrundlage ein amtliches 3D-Stadtmodell der Stadt Hamburg vor, dessen Gebäude mit einer Detailstufe von LoD3 dargestellt werden. Dazu kann zwischen verschiedenen Hintergrundkarten gewechselt und Geodaten aus verschiedenen Fachbereichen eingeblendet werden. Als Toolbox wird der sogenannte 3D-Modeller bereitgestellt, welcher einen Import von 3D-Objekten ermöglicht. Zudem steht für die 3D-Umgebung ein Zeichentool zur Verfügung. Darüber hinaus können Schattenwürfe simuliert

werden und das Gebiet sowohl aus einer 360°-Perspektive als auch aus Sicht eines Fußgängers navigiert und betrachtet werden (Winter, 2023).

Für den kommerziellen Gebrauch wurde vom Hersteller ESRI, als Entscheidungshilfe für urbane Planungsszenarien, ArcGIS Urban entwickelt. Dabei handelt es sich um eine Webanwendung, mit der sich städtebauliche Pläne in eine 3D-Stadtumgebung einbetten lassen und GIS-Werkzeuge für die Planung bereitgestellt werden (ESRI, o.J. e). Dort werden sowohl Flächennutzungspläne als auch Bebauungspläne unterstützt. Dabei können verschiedene Parameter zur baulichen Nutzung in 3D dargestellt und für die Ermittlung des Flächenpotentials, sowie für zukünftige planerische Festsetzungen ausgewertet werden (ESRI, o.J. b). Darüber hinaus können Strecken- und Flächenmessungen, sowie Sichtbarkeits- und Schattenwurfanalysen durchgeführt werden. Auch lassen sich Querschnitte ausgewählter Bereiche anzeigen und als Höhenprofil darstellen (ESRI, o.J. a). Innerhalb von ArcGIS Urban können zudem modellierte urbane Daten aus der ArcGIS CityEngine eingebunden werden (Badwi et al., 2022).

## 2.3 Prozedurale Modellierung

Die Modellierung größerer urbaner Gebiete erfordert auf manuelle Weise einen hohen Bearbeitungsaufwand. Um diesen Prozess effizienter und wirtschaftlicher zu gestalten gewinnen Methoden zur Erstellung von prozeduralen Inhalten an Popularität. Bereits seit den 1980er Jahren werden prozedurale Methoden in der Spieleentwicklung eingesetzt (Hendrikx et al., 2013).

Bezüglich der Erstellung von 3D-Modellen ist die prozedurale Modellierung als Teildisziplin entstanden, bei der dreidimensionale Geometrien, Texturen und Animationen modelliert werden (Demir & Koramaz, 2018). Im Rahmen der digitalen Stadtmodellierung werden 3D-Modelle von urbanen Objekten angefertigt. Hierbei wird für die Erstellung in der Regel auf grammatikbezogene Methoden wie L-Systeme und Shape Grammars zurückgegriffen (Jesus et al., 2012). Alternativ dazu gibt es eine Vielzahl weiterer Methoden, die für die prozedurale Modellierung von Städten zum Einsatz kommen (Kim et al., 2018).

### 2.3.1 Procedural Content Generation

Durch Procedural Content Generation (kurz: PCG) lassen sich jegliche Inhalte größtenteils automatisch und mit wenigen Benutzereingaben erstellen, indem auf vordefinierte Muster und Algorithmen zurückgegriffen wird (Freiknecht & Effelsberg, 2017). Von der Strukturierung

simplerer Gebiete bis hin zur Erstellung von Gegenständen und Aufgaben wirkt sich PCG auf eine Vielzahl verschiedener Inhalte in Spielen aus (siehe Abbildung 10) (Togelius et al., 2013).

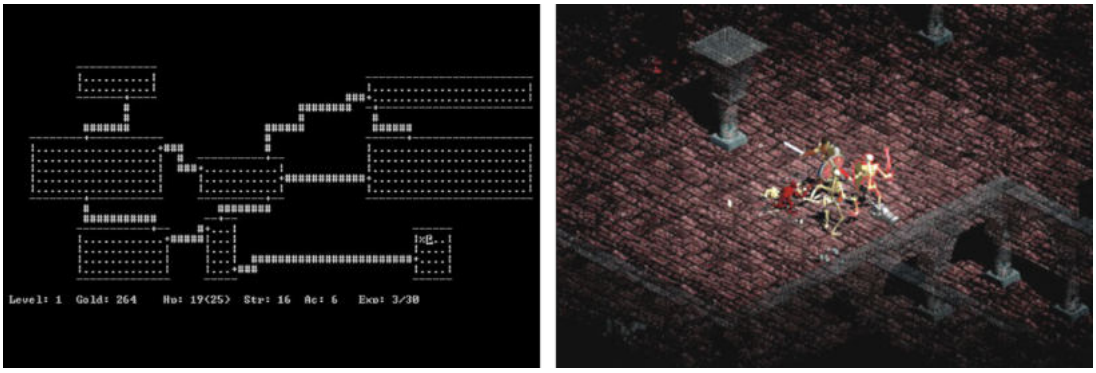


**Abbildung 10:** Prozedural generierte Gebiete aus Computerspielen: Rogue aus dem Jahr 1980 (links) und Diablo aus dem Jahr 1996 (rechts) (Amato, 2017, verändert)

Nach Ebert et al. zählen dabei die Abstraktion, Flexibilität und die Steuerung durch Parameter zu den wesentlichen Eigenschaften von PCG (Ebert et al., 2002). Dabei hat die Nutzung von PCG den Vorteil, dass Entwicklungszeiträume und -kosten reduziert werden können und eine hohe Variation an Inhalten erhalten bleibt (Togelius et al., 2013). Dadurch bleiben die Spiele abwechslungsreich und erzielen langfristig einen höheren Wiederspielwert (Smith et al., 2011).

Die prozedural generierten Inhalte werden als Produktionsabfolgen gespeichert, aus denen später Objekte instanziiert werden können (Hofmann & Heller, 2013). Dadurch können große Mengen an Speicherplatz gespart und die Inhalte in verschiedenen Detailstufen dargestellt werden. Durch Parametereingaben lässt sich das prozedural generierte Ergebnis maßgeblich beeinflussen. Jedoch besteht häufig das Problem, dass sich das generierte Ergebnis schwierig voraussagen lässt und das gewünschte Ergebnis nur durch wiederholtes Ausprobieren erzielt werden kann (Frade et al., 2012).

### 2.3.2 L-Systeme

In der Natur lassen sich selbstähnliche Strukturen beobachten, die sich mithilfe von mathematischen Modellen beschreiben lassen. Um Vorgänge innerhalb von Pflanzenzellen nachzumodellieren, führte Lindenmayer als Grammatik das Lindenmayer-System (kurz: L-System) ein (Lindenmayer, 1968). Dabei handelt es sich um Produktionsabfolgen, bestehend aus Zeichenketten, die sich rekursiv selbst ersetzen. L-Systeme setzen sich aus einer Menge an Variablen und Konstanten zusammen, die durch eine Startregel initiiert werden. Durch eine Menge an

Regeln wird beschrieben, in welcher Weise die Variablen rekursiv durch andere Variablen und Konstanten ersetzt werden (Kelly & McCabe, 2006).

Dieser Vorgang lässt sich durch Fraktale visuell nachvollziehen, bei denen durch Rekursion selbstähnliche geometrische Muster entstehen (Mandelbrot, 1982). Durch die Turtle-Interpretation können die Zeichenabfolgen von L-Systemen durch Graphen visualisiert werden. Dabei werden eine konstante Länge und Winkel sowie positive und negative Richtungsänderungen vorgegeben. Als Ergebnis lassen sich Geometrien mit selbstähnlichen Strukturen erzeugen (siehe Abbildung 11). Mit steigender Rekursionstiefe vergrößert sich die Auflösung, in der sich alle vorhergehenden Strukturen wiederfinden lassen (Prusinkiewicz & Lindenmayer, 1990).

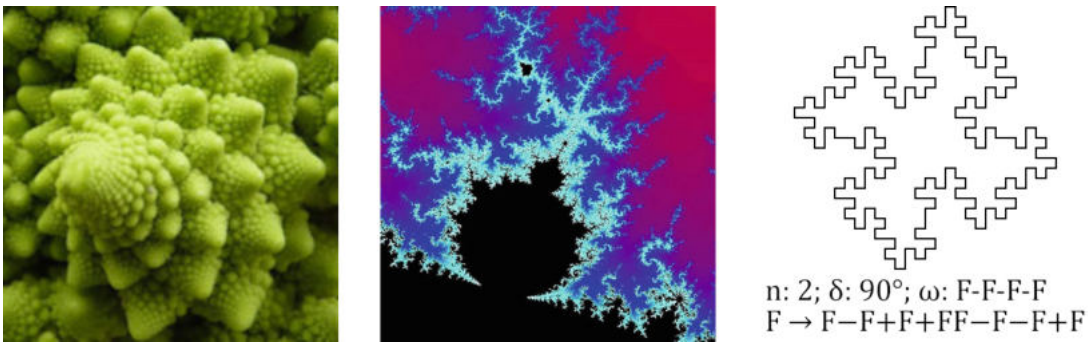


**Abbildung 11:** Selbstähnliche Strukturen in Romanesco (links), Mandelbrot-Menge (mitte) (Blatz & Korn, 2017, verändert) und Darstellung von L-Systemen nach Turtle-Interpretation (Prusinkiewicz & Lindenmayer, 1990, verändert)

Durch L-Systeme können bereits mit wenigen Produktionsvorgaben komplexe Ergebnisse in beliebigen Detailstufen erzeugt werden. Solche grammatikbasierten Konzepte wurden später für die prozedurale Modellierung von urbanen Gebieten in die Praxis umgesetzt. In der CityEngine können Straßenstrukturen durch eine erweiterte Form von L-Systemen generiert werden (Parish & Müller, 2001).

### 2.3.3 Shape-basierte Grammatiken

Später wurde die prozedurale Modellierung von Gebäuden, basierend auf der Grundidee von Shape Grammars, für die CityEngine implementiert (Müller et al., 2006).

Das Shape Grammar wurde erstmals von Stiny & Gips vorgestellt, wobei in dieser Grammatik anstelle von Zeichenabfolgen Shapes verwendet werden (Stiny & Gips, 1971). Bei den Shapes handelt es sich dabei nach Stiny um gerade Linien und Zusammensetzungen solcher im kartesischen Koordinatensystem (Stiny, 1980). Mithilfe von Produktionsregeln wird beschrieben,

welche Ausprägungen von Shapes von der Transformation betroffen sind. Sie definieren die räumliche Ausdehnung solcher Shapes vor und nach der Transformation (Hofmann & Heller, 2013).

Durch die Transformation selbst können Bestandteile des Shapes erweitert, entfernt oder verändert werden. Auch können räumliche Beziehungen zu anderen Shapes hergestellt werden. Durch Anwendung der Produktionsregeln können die eingehenden Shapes dann schrittweise transformiert werden (Gu & Maher, 2014). Dabei kann es innerhalb des Shape Grammars zufallsbedingt zu einer Vielzahl unterschiedlicher Ergebnisse kommen, da die Transformation mitunter auf unterschiedliche Bestandteile der Shapes angewendet werden kann (siehe Abbildung 12) (Al-kazzaz & Bridges, 2012).

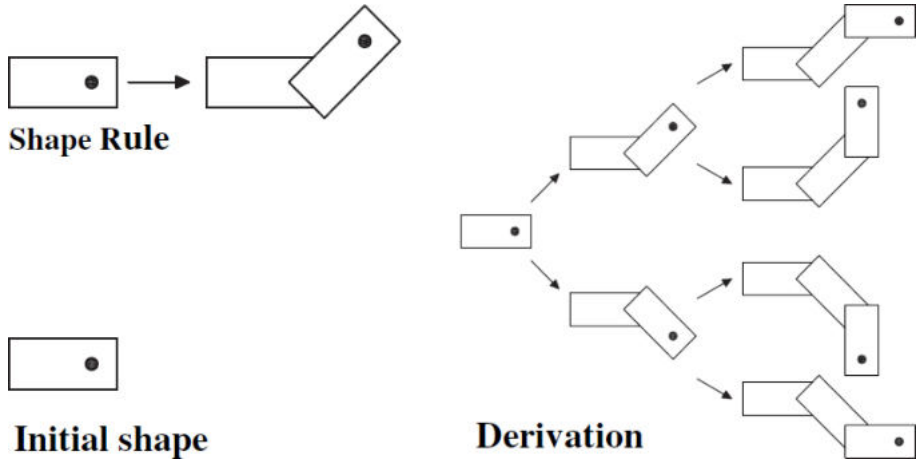


**Abbildung 12:** Beispiel eines Shape Grammars in unterschiedlichen Variationen (Al-kazzaz & Bridges, 2012, verändert)

Beim Shape Grammar wird sich grundsätzlich auf Shapes als einzelne Objekte bezogen. Eine Grammatik, die mit einer Menge von Objekten operiert, wird als Set Grammar bezeichnet. Basierend auf der Grundidee, Shapes als Mengen für die Modellierung von Architektur zu nutzen, wurde das Split Grammar von Wonka et al. eingeführt. Diese Grammatik spezialisiert sich auf die Modellierung von Fassaden, die in kleinere Bestandteile zerlegt und detailgetreu dargestellt werden (Wonka et al., 2003).

Solche Mengenoperationen durch Set Grammars haben den Vorteil, dass sie sich computergestützt performant umsetzen lassen. Um Gebäude in der CityEngine prozedural zu modellieren, wurde durch Müller et al. das Computer Generated Architecture (kurz: CGA) Format eingeführt. Hierbei lassen sich verschiedene Transformationsregeln auf Shapes anwenden. Zudem werden grundlegende Zerlegungsregeln zur Fassadenmodellierung eingeführt (Müller et

al., 2006). Auf zweidimensionale Grundrisse kann eine solche CGA-Datei referenziert werden, wodurch die Regeln automatisch angewendet werden und ein komplexes 3D-Modell entsteht (siehe Abbildung 13) (Badwi et al., 2022).


**Abbildung 13:** Grundkonzept zur Erstellung eines 3D-Modells durch Anwendung einer Regel-Datei auf 2D-Grundrisse (Badwi et al., 2022)

Die Vielzahl an komplexen Variationen, die durch das Shape Grammar entstehen können und die mengenbezogene Anwendung durch das Set Grammar, ermöglichen eine flächenhafte Modellierung von Siedlungsblöcken.

### 2.3.4 Weitere Modellierungsmethoden

Grammatikbasierte Ansätze setzen ein gewisses Maß an Vorwissen beim Nutzer voraus. Bei Stadtplanern, die ggf. über wenig Programmierkenntnisse verfügen, gestaltet sich dieser Ansatz der prozeduralen Modellierung als schwierig. Hierbei stellt die inverse Modellierung eine mögliche Alternative dar, bei der anhand eines gewünschten Ergebnisses die dafür benötigten Parameter und Regeln ermittelt werden. Der Ansatz von Vanegas et al. verwendet für die Parametersuche Markov Chain Monte Carlo (kurz: MCMC). Anhand urbaner Indikatoren, lassen sich die Stadtmodelle nachträglich editieren (Vanegas et al., 2012).

Andere Ansätze beschäftigen sich mit einer interaktiven Fassadenmodellierung auf Grundlage von Zeichnungen oder Bildern. Müller et al. zeigen in ihrer Arbeit, wie 3D-Modelle und Produktionsregeln aus Bildern von Fassaden erzeugt werden. Durch manuelles Einfügen von vertikalen und horizontalen Linien können abgebildete Fassaden in einzelne Bestandteile zerlegt werden (Müller et al., 2007). Diese Funktion steht in der CityEngine durch den Facade Wizard zur Verfügung (ESRI, o.J. c). Eine weitere Möglichkeit stellt das Sketch Based Modeling nach Hu et al. dar. Durch das Anfertigen einer groben Zeichnung von Fenstern, wird durch neuronale Netzwerke automatisch eine Fensterform erkannt und dazu ein 3D-Modell mit

Produktionsregeln erzeugt (Hu et al., 2022).

Als weitere Methoden zur prozeduralen Modellierung nennen Kim et al. stochastische, datengetriebene und simulationsbasierte Ansätze, sowie die Nutzung von Tensorfeldern (Kim et al., 2018). Auf stochastischem Wege können 3D-Modelle unter anderem durch Perlin Noises erzeugt werden. (Blatz & Korn, 2017). Dadurch lassen sich ganze Landschaften, einschließlich Wolken und Gewässer, sowie Texturen erzeugen (Kelly & McCabe, 2006). Eine weitere Alternative stellen Graph Grammars dar. Die Grundidee besteht darin, Shapes in primitive Einzelteile zu zerlegen und jede mögliche Kombination schrittweise in einer Hierarchie festzuhalten, wodurch Produktionsregeln erstellt werden. Auf Grundlage von Graph Grammars lassen sich aus eingehenden Shapes, ähnliche komplexere Shapes als Ergebnis erzeugen (Merrell, 2023).

Als Alternative zu den zumeist textbasierten Shape Grammars, lassen sich Produktionsregeln auch visuell über Graphen erstellen. Die node-basierte Form der prozeduralen Modellierung wird beispielsweise durch Houdini von SideFX und der Unreal Engine von Epic Games unterstützt (Patow, 2012). Bei Houdini lassen sich über eine Anordnung einfacher mathematischer oder geometrischer Funktionen größere Workflows erstellen, welche sich als eigene Nodes gruppieren lassen (Smelik et al., 2014). Diesen Ansatz verfolgt auch das PCG Framework der Unreal Engine (Epic Games, o.J. e).

## 2.4 Unreal Engine

Bei der Unreal Engine handelt es sich um eine Grafik-Engine vom Softwareunternehmen Epic Games, die in erster Linie zur Entwicklung von Computerspielen dient. Inzwischen wird die Engine auch außerhalb der Gamingbranche eingesetzt, darunter in der Filmindustrie, sowie in der Architektur und Simulation (Epic Games, o.J. c).

Die Unreal Engine stellt dem Entwickler mit Blueprints und C++ zwei unterschiedliche Ansätze für die Softwareentwicklung zur Verfügung. Bei den Blueprints handelt es sich um eine node-basierte Programmiersprache, die exklusiv für die Unreal Engine konzipiert wurde zu können. Alternativ kann auch direkt in C++ programmiert werden, insbesondere um auf tiefgreifendere Prozesse der Engine zugreifen zu können (Epic Games, 2023).

In einem Unreal Engine Projekt wird die Welt des Spiels gespeichert, die aus einem oder mehreren Leveln, also Gebieten des Spiels, besteht. Hierbei stammen alle Objekte der Unreal Engine von der Basisklasse Object ab. Alle Objekte, die sich innerhalb eines Level platzieren lassen,

werden als Actor bezeichnet. Weiterführend kann der Avatar eines Spielers als Pawn mit dem Level und den dort platzierten Actors interagieren. Zudem stellen Actors Container dar, die aus einem oder mehreren Komponenten bestehen können und das Erscheinungsbild, sowie Verhalten des Actors prägen (siehe Abbildung 14). Durch Blueprints lassen sich Actors und andere Klassen erweitern, wodurch sich komplexere Verhaltensweisen implementieren lassen (Epic Games, o.J. j).

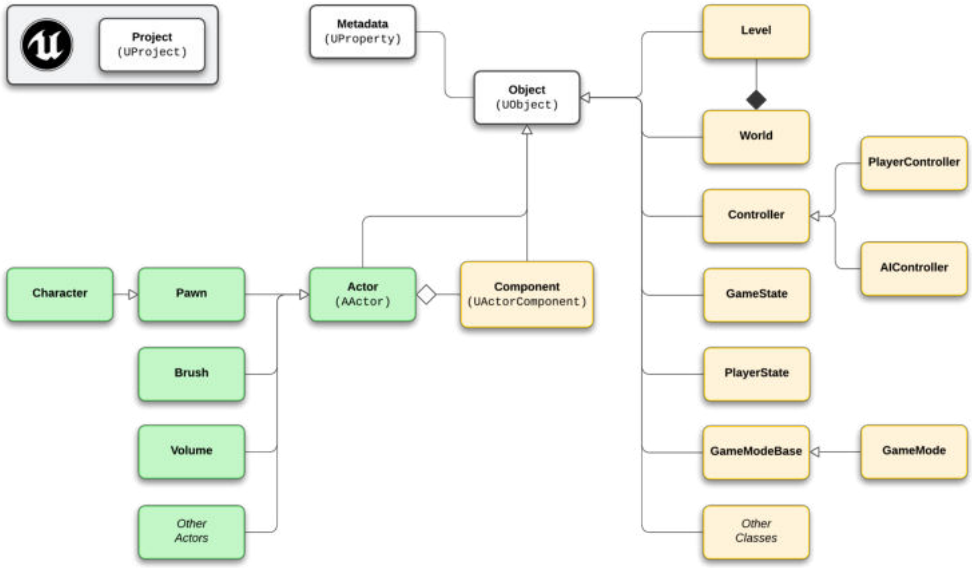


**Abbildung 14:** Übersicht zur Vererbungshierarchie in der Unreal Engine (eigene Darstellung, erstellt mit Lucidchart)

# 3 Konzeptionierung

## 3.1 Anforderungen

Im Rahmen dieser Arbeit wird ein Prototyp eines dreidimensionalen Bebauungsplans konzipiert, der die grundlegenden Anforderungen urbaner digitaler Zwillinge und vergleichbarer Anwendungen erfüllen soll. Um diese Anforderungen zu ermitteln, zu dokumentieren und zu analysieren, wird in der Softwareentwicklung das Requirements Engineering angewendet. Grundsätzlich wird dabei zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden. Funktionale Anforderungen beziehen sich auf den Input und Output eines Systems sowie dessen einzelne Funktionen, während sich nicht-funktionale Anforderungen auf das System als Ganzes und dessen Qualität beziehen (Partsch, 2010).

### 3.1.1 Stakeholder

Im Rahmen der Öffentlichkeitsbeteiligung in der Bauleitplanung kann die gesamte Öffentlichkeit am Verfahren mitwirken, wodurch die Anzahl der beteiligten Akteure breit gefächert ist. Grundsätzlich wirken im Planungsprozess, Stadtplaner, Träger öffentlicher Belange, sowie betroffene und beteiligte Bürgerinnen und Bürger mit (Pahl-Weber & Schwartze, 2018). Um deren Bedürfnisse besser zu verstehen und im Prototypen zu berücksichtigen, wurden Personas erstellt, welche stellvertretend für die verschiedenen Nutzergruppen stehen (siehe Anlage 1):

(1) Stadtplaner: Er stellt sowohl funktionale als auch nicht-funktionale Anforderungen an die Software. Sie sollte ihm eine breite Palette an Funktionen bieten, die seinen Horizont im Planungsprozess erweitern können. Um die Integrität der Anwendung zu gewährleisten, müssen die Planinhalte gesetzeskonform dargestellt werden. Zudem sollten die Analyse- und Simulationsfunktionen möglichst präzise und zuverlässig arbeiten. Dabei möchte er eine enge Zusammenarbeit mit der Öffentlichkeit und den Trägern öffentlicher Belange anstreben. Hierfür möchte er die Entscheidungsprozesse innerhalb der Planung transparent und nachvollziehbar vermitteln.

(2) Umweltgutachterin: Stellvertretend für die Träger öffentlicher Belange stellt sie ähnliche Anforderungen wie der Stadtplaner. Fachbezogene Daten, wie beispielsweise Umweltbelastungen, sollten in solchen Anwendungen interoperabel integriert und dargestellt werden können. Zudem sollten zeitliche Veränderungen visualisiert und mittels Monitoring überwacht, sowie dokumentiert werden können.

(3) Erwerberin des Baugrundstücks: Stellvertretend für die Öffentlichkeit stellt sie hauptsächlich nicht-funktionale Anforderungen an die Software. Sie möchte das Planungsvorhaben nachvollziehen können, ohne sich intensiv mit der Darstellung von Planinhalten und gesetzlichen Vorschriften auseinandersetzen zu müssen. Das Planungsgebiet sollte für sie einfach navigierbar sein und sich aus verschiedenen Perspektiven betrachten lassen. Außerdem möchte sie mit dem Gebiet interagieren und Informationen darüber aufrufen können.

### 3.1.2 Funktionale Anforderungen

Aus den Bedürfnissen der Stakeholder lassen sich konkrete Anwendungsfälle ableiten, die durch die Anwendung abgedeckt werden sollten. Hierfür wurde ein Use Case Diagramm (dt. Anwendungsfalldiagramm) erstellt (siehe Anlage 2). Für die Umsetzungen urbaner digitaler

Zwillinge in deutschen Kommunen wurden durch den Standard DIN SPEC 91607 bereits vergleichbare Anwendungsfälle definiert, die sich auf die digitale Bauleitplanung übertragen lassen (DIN, 2024). Bei dieser Anwendung wurde sich schwerpunktmäßig auf die Planung und Simulation, sowie die digitale Beteiligung beschränkt. Weiterführend wurden dazu funktionale Anforderungen formuliert, die das System erfüllen kann, soll oder muss, um die Bedürfnisse der Stakeholder zu erfüllen:

(1) Planung und Simulation: Die Anwendung muss es dem Stadtplaner ermöglichen, verschiedene Planungsszenarien gegenüberzustellen und das Planungsgebiet parametergesteuert zu bearbeiten. Dabei müssen die gesetzlichen Anforderungen der Bauleitplanung durch die Anwendung berücksichtigt werden. Sie soll Funktionen zur Verfügung stellen, um urbane Objekte zu platzieren und zu bearbeiten, sowie die Durchführung von Strecken- und Flächenmessungen zu ermöglichen. Außerdem soll die Anwendung verschiedene Simulations- und Analysefunktionen bereitstellen, darunter Sichtbarkeitsanalysen und Beschattungs- und Tageslichtsimulationen. Durch die Anwendung sollen den Trägern öffentlicher Belange fachbezogene Planungs- und Simulationsfunktionen zur Verfügung gestellt werden.

(2) Digitale Beteiligung: Die Anwendung muss es allen beteiligten Akteuren ermöglichen, miteinander zu kommunizieren und zu kollaborieren. Das Planungsgebiet soll sich aus verschiedenen Perspektiven betrachten lassen. Außerdem können durch die Anwendung Stellungnahmen erfasst werden, die im Rahmen der Öffentlichkeitsbeteiligung geäußert werden. Auch können Bürgerinnen und Bürger Auskünfte über das Planungsgebiet erhalten.

### 3.1.3 Nicht-funktionale Anforderungen

Damit die Anwendung eine möglichst hohe Qualität erzielt, gilt es, eine Vielzahl an Herausforderungen zu bewältigen. Dabei geht es nicht nur darum die herkömmlichen Anforderungen für moderne Anwendungen zu erfüllen, sondern auch den zukunftsgerichteten Anforderungen für einen urbanen digitalen Zwilling gerecht zu werden. Daraus ergeben sich folgende nichtfunktionale Anforderungen, die es zu berücksichtigen gilt:

(1) Gebrauchstauglichkeit: Nach der Definition der ISO 9241-11 muss das Programm effektiv, effizient und zufriedenstellend funktionieren, um als gebrauchstauglich zu gelten (DIN, 2018). Die Anwendung muss intuitiv gestaltet sein, sodass Stakeholder bereits mit wenig Fachwissen die Anwendung bedienen können. Unter anderem sollte sich die Anwendung einfach navigie-

ren lassen, bei Benutzereingaben mit Feedback reagieren und eine übersichtliche Menüführung aufweisen.

(2) Datenintegration und Semantik: Die Anwendung sollte urbane Daten in semantischer Form beschreiben und verwalten, sowie interoperabel austauschen können. Die Datenintegration sollte auf verschiedenen Ebenen unterstützt werden und eine Erweiterbarkeit des urbanen digitalen Zwillings für verschiedene Fachbereiche ermöglichen. Gerade bei Game Engines stellt die Integration von Geodaten zurzeit noch ein Problem dar (Rantanen et al., 2023).

(3) Skalierbarkeit und Unterstützung dynamischer Daten: Caprari et al. nennen als weitere Eigenschaften eines digitalen Zwillings die Skalierbarkeit und die Integration dynamischer Daten (Caprari et al., 2022). Die Anwendung sollte in der Lage sein, über die Cloud auf externe Datenquellen zuzugreifen und Ressourcen nach Bedarf skalieren zu können. Dabei soll Datenredundanz vermieden werden und größere Datenmengen ohne Speicherprobleme abrufbar sein. Zudem sollte die Anwendung neben statischen Daten auch dynamische Daten aus dem IoT integrieren können und in einem automatischen Datenaustausch stehen.

## 3.2 Projektplanung

Nach der Anforderungsanalyse wurde im Rahmen des Projektmanagements der Entwicklungszeitraum festgelegt, die erforderliche Software und Hardware beschafft, sowie Ort und Dimension des Projektgebiets festgelegt.

### 3.2.1 Zeitmanagement

Für die Erstellung des Prototypen war ein Entwicklungszeitraum von 13 Wochen vorgesehen. Bei der Projektplanung wurde sich am Scrum-Modell orientiert, indem der Projektfortschritt in wöchentlichen Sprints geplant wurde. Als Hilfsmittel wurde hierfür die Software Jira vom Unternehmen Atlassian verwendet. Die Anforderungen ließen sich als sogenannte Epics formulieren, welche sich hierarchisch in einzelne Tasks (dt. Aufgaben) und Stories (dt. Geschichten) unterteilen ließen. Auch zu korrigierende Bugs (dt. Programmfehler) können dort dokumentiert und berücksichtigt werden (Rehkopf, o.J.).

In jedem wöchentlichen Sprint wurden die formulierten Aufgaben und Fehlerkorrekturen abgearbeitet. Der wochenübergreifende Fortschritt lässt sich über Berichte in verschiedenen Diagrammen und Listen visualisieren und abschließend dokumentieren.

### 3.2.2 Systemanforderungen

Während der Entwicklung wurde die Unreal Engine in der Version 5.4.4 verwendet. Damit die Entwicklung des Prototyps ohne technische Schwierigkeiten verläuft, mussten die Software- und Hardwareanforderungen der Unreal Engine 5 beachtet werden (Epic Games, o.J. a). Für die Hardware gelten Mindestanforderungen, die mit dem verwendeten PC verglichen wurden (siehe Tabelle 1).

**Tabelle 1:** Mindestanforderungen und verwendete Hardware für die Unreal Engine 5 bezogen auf Windows-PCs (Epic Games, o.J. a)

| **Komponente** | **Mindestanforderungen** | **Verwendete Hardware** |
|---|---|---|
| Betriebssystem | Windows 10 (64-bit) / 11 | Windows 10 (64-bit) |
| Prozessor (CPU) | Quad-Core Intel / AMD (2.5 GHz) | AMD Ryzen 9 3900X (3.79 GHz) |
| Grafikkarte (GPU) | DirectX 11 / 12<br>(8 GB) | DirectX 12<br>NVIDIA GeForce RTX 2070 (8 GB) |
| Arbeitsspeicher (RAM) | *(keine Angabe)* | 32 GB |

Darüber hinaus wurde das Projekt durch Plugins erweitert. Um die prozedurale Modellierung innerhalb der Engine zu ermöglichen, wurde das PCG Framework der Unreal Engine verwendet. Zur Integration und Darstellung dreidimensionaler Geodaten dient hingegen das Plugin Cesium for Unreal.

### 3.2.3 Projektgebiet

Für die Wahl des Projektgebiets kamen ausschließlich unbebaute Neubausiedlungen aus dem Bundesland Niedersachsen infrage. Der dazugehörige Bebauungsplan sollte rechtsverbindlich und möglichst aktuell sein. Zu Demonstrationszwecken sollte der Plan nicht vorhabensbezogen sein und hauptsächlich Einfamilienhäuser beinhalten. Das Projektgebiet bezieht sich auf den Bebauungsplan Nr. 629 „In der Steiniger Heide“. Dieser liegt in der Gemarkung Voxtrup in der Stadt Osnabrück, nördlich nahe der Autobahn A30 (siehe Anlage 4). Der Bebauungsplan ist im Jahr 2022 in Kraft getreten und unterliegt keinen zwischenzeitlichen Änderungen. Im Internet steht der Plan als PDF-Datei von der Stadt Osnabrück öffentlich zur Verfügung (Stadt Osnabrück, 2022). Für dieses Projekt wird sich ausschließlich auf den zeichnerischen Teil des Bebauungsplans bezogen (siehe Abbildung 15). Die Festsetzungen des textlichen Teils wurden dabei nicht beachtet.

**Abbildung 15:** Übersicht zum Projektgebiet: DOP20 (links) (LGLN, o.J.) und zeichnerischer Teil des Bebauungsplans Nr. 629 (rechts) (Stadt Osnabrück, 2022)

## 3.3 Projektdaten

Analog zu 3D-Stadtmodellen werden verschiedene Geobasisdaten in den Prototyp eingebunden und miteinander kombiniert. Für die Integration der Geodaten wurde Cesium verwendet. Cesium stellt eine Open Source Plattform dar, mit der sich dreidimensionale Geodaten hosten lassen (Cesium, o.J. c).

### 3.3.1 Datenbeschaffung und -aufbereitung

Um eine hohe Zuverlässigkeit der Metadaten zu gewährleisten, wurden öffentlich verfügbare Daten des Landesamtes für Geoinformation und Landesvermessung (kurz: LGLN) von der OpenData-Plattform OpenGeoData.NI bezogen. Diese stehen flächendeckend für das Land Niedersachsen kostenlos zur Verfügung. Zudem sind die Metadaten bezüglich der Datenqualität und -aktualität öffentlich einsehbar (LGLN, o.J.).

Als Terraindaten für den Prototypen wurden ausgewählte Kacheln des DGM mit einer Rasterweite von 1 Meter verwendet. Derzeit liegen für das Projektgebiet ausschließlich Daten aus dem Jahr 2017 vor. Die Kacheln sind als Cloud-Optimized GeoTIFF (kurz: COG) formatiert und bereits georeferenziert. Damit die Daten später reibungslos über Cesium in die Anwendung integriert werden können, war eine Aufbereitung der Daten erforderlich. Um das gesamte Projektgebiet mit Terraindaten abzudecken, wurden vier benachbarte Kacheln bezogen und gemeinsam aufbereitet. Dabei wurden die Kacheln in das GIS-Programm QGIS importiert und zu einer zusammenhängenden Kachel verschmolzen. Zudem wurde eine flächendeckende Kachel vom DOP mit einer Bodenauflösung von 20 Metern eingebunden. Die Kachel entspricht einem

TrueDOP im GeoTIFF-Format und stammt aus dem Jahr 2023.

Zur Repräsentation der umliegenden Gebäude im Projektgebiet wurden 3D-Gebäudemodelle mit einer Detailstufe von LoD2 verwendet, wobei die Gebäude eine standardisierte Dachform besitzen. Die Datei liegt im CityGML-Format vor und enthält Daten aus dem Jahr 2020. Damit die Daten mit Cesium kompatibel sind, war es erforderlich, einen Header zu Angabe eines „gml:BoundedBy"-Elements einzufügen. Erst dann ließen sich die CityGML-Daten in Cesium interpretieren und als 3D-Tiles verwalten.

Durch die unterschiedliche Auflösung und Aktualität der Daten kann es zu räumlichen und temporalen Abweichungen kommen. Alle bezogenen amtlichen Daten liegen im Lagebezugssystem ETRS89/UTM32 (EPSG: 25832) und im Höhenbezugssystem DHHN2016 (EPSG: 7837) vor (siehe Tabelle 2).

**Tabelle 2:** Übersicht der im Projekt verwendeten amtlichen Geodaten (eigene Darstellung)

| **Bezeichnung** | **Dateiformat** | **Jahr** | **Genauigkeit** |
|---|---|---|---|
| 3D-Gebäudemodell (LoD2) | CityGML | 2020 | Standardisierte Dachformen |
| Digitales Geländemodell (DGM1) | GeoTIFF | 2017 | Rasterweite von 1 m |
| Digitales Orthophoto (DOP20) | GeoTIFF | 2023 | Bodenauflösung von 20 m |
| **Lagebezugssystem:** ETRS89/UTM32 (EPSG: 25832)<br>**Höhenbezugssystem:** DHHN2016 (EPSG: 7837) | | | |

Um den Bebauungsplan in das Projekt einzubinden, wurde der zeichnerische Teil des Planes mit dem Bildbearbeitungsprogramm GIMP ausgeschnitten. Darauffolgend wurde der Bildausschnitt im GIS-Programm ArcGIS importiert und anhand visueller Übereinstimmungen mit dem DOP20 georeferenziert. Das Ergebnis wurde anschließend als GeoTIFF exportiert.

Für die Unreal Engine standen über die Plattform Quixel Bridge, Megascans als texturierte und hochauflösende 3D-Modelle kostenlos zur Verfügung. Inzwischen sind diese Modelle teilweise kostenpflichtig über die Plattform Fab erhältlich (Epic Games, o.J. f). Für dieses Projekt wurden ausschließlich Modelle mit einer Texturauflösung von 2K (2048 x 2048 px) verwendet. Diese konnten direkt in die Unreal Engine importiert werden. Dabei kamen ausschließlich Modelle von Gebäudeteilen in verschiedenen Variationen zum Einsatz (siehe Abbildung 16).

**Abbildung 16:** 3D-Modelle von Außenwänden als Quixel Megascans (Epic Games, o.J. g)

### 3.3.2 Datenintegration mit Cesium

Die Pipeline zur Datenintegration mit Cesium setzt sich aus der Plattform Cesium ion, der Konvertierung in 3D-Tiles und dem Plugin Cesium for Unreal zusammen (Cesium, o.J. c). Das Plugin Cesium for Unreal ermöglicht die Einbindung verschiedener Geodaten (Cesium, o.J. b). Hierbei kann sowohl auf öffentliche als auch eigene Inhalte über die Plattform Cesium ion zugegriffen werden. Darüber lassen sich über 3D-Tiles, Punktwolken, Photogrammetrie, 3D-Modelle und Gebäude, sowie Satellitenbilder und Terraindaten beziehen (Cesium, o.J. d). Bei den 3D-Tiles handelt es sich um OGC-konforme Datensätze, die in Cesium für die Darstellung dreidimensionaler Daten verwendet werden (Cesium, o.J. a).

Um eigene Daten über Cesium ion einzubinden, ist ein eigener Account erforderlich. Dieser kann mit dem Plugin Cesium for Unreal verknüpft werden, wodurch sich die hochgeladenen Daten nahtlos in die Engine einbinden lassen. Die aufbereiteten Daten wurden auf Cesium ion hochgeladen und stehen dort unter einer eigenen ID zur Verfügung (siehe Tabelle 3).

**Tabelle 3:** Verwendete Daten in Cesium ion (eigene Darstellung)

| ID | Name | Datentyp |
|---|---|---|
| **Eigene Daten** | | |
| 2769919 | 3D-Gebäudemodell (LoD2) | 3D Tiles |
| 2770011 | Digitales Geländemodell (DGM1) | Terrain |
| 2769940 | Digitales Orthophoto (DOP20) | Imagery |
| 2721002 | Bebauungsplan Nr. 629 (zeichnerischer Teil) | Imagery |
| **Öffentlich verfügbare Daten** | | |
| 1 | Cesium World Terrain | Terrain |
| 2 | Bing Maps Aerial | Imagery |

Über die Angabe eines Tokens und einer ID können die gehosteten Daten schließlich in die Engine geladen werden. Beim Import der Terraindaten in die Unreal Engine liegen diese zunächst ohne Textur vor. Damit Satellitenbilder oder Karten, unter Berücksichtigung der Terraininformationen, dargestellt werden, lassen sich Daten vom Typ Imagery als Komponente dem Terrain anhängen. Dadurch erhält das Terrain eine Textur. In diesem Fall wurden für das aufbereitete DGM1 sowohl das DOP20 als auch der georeferenzierte Bebauungsplan eingebunden.

Das Terrain erstreckt sich über die gesamte Erdöberfläche. Um bestimmte Bereiche des Terrains auszuschneiden, stellt Cesium Polygone zur Verfügung. Dabei kann festgelegt werden, ob Bereiche innerhalb oder außerhalb des Polygons entfernt werden. Entsprechend wurde für den zeichnerischen Teil des Bebauungsplans der unbeplante Bereich außerhalb der Plangrenze ausgeschnitten. Für das DOP20 hingegen wurde der Bereich entfernt in dem sich der Bebauungsplan befindet. Dadurch konnte ein fließender Übergang zwischen dem georeferenzierten Bebauungsplan und dem DOP20 hergestellt werden (siehe Abbildung 17).

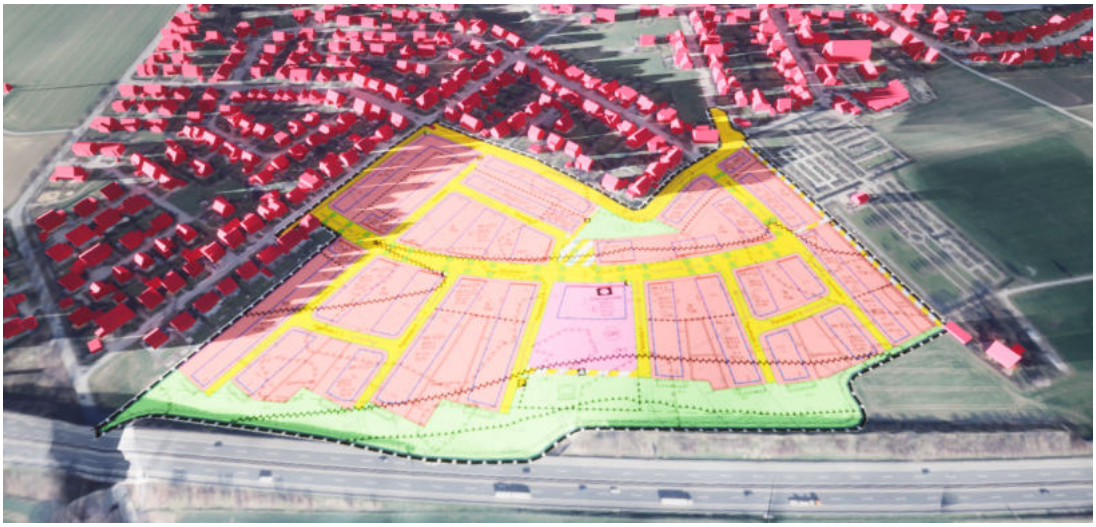

**Abbildung 17:** Integration eigener Daten über das Plugin Cesium for Unreal (eigene Darstellung)

Die eingebundenen Daten werden durch Cesium in das Koordinatensystem WGS84 transformiert und dargestellt (Cesium, o.J. b). Da sich die amtlichen Daten ausschließlich auf einen begrenzten Bereich beziehen wurden die außerhalb liegenden Bereiche durch unsichtbare Barrieren unzugänglich gemacht. Außerhalb des Projektgebiets werden öffentlich verfügbare Daten von Cesium zu Darstellungszwecken verwendet. In diesem Fall basieren die Terraindaten auf das Cesium World Terrain. Als Imagery wurden die Satellitenbilder von Bing Maps Aerial verwendet.

# 4 Realisierung

## 4.1 Softwarearchitektur

Die Umsetzung des Prototyps folgt einer objektorientierten Programmierung. Dementsprechend war es dafür erforderlich, die wesentlichen Bestandteile als Klassen zu identifizieren und deren Relation zu beschreiben. Grundsätzlich werden die meisten Klassen in diesem Projekt als Actors verwaltet und mittels Blueprint-Skripting mit komplexeren Funktionen versehen. Darüber hinaus wurden diese mit PCG-Graphen verknüpft, die für die prozedurale Modellierung erforderlich sind. Zudem werden Data Assets und Structs als Datenstrukturen verwendet, um eine strukturierte Verwaltung von Attributen und eine modulare Austauschbarkeit zu ermöglichen.

### 4.1.1 Blueprints

Zunächst wurden die Planinhalte des Bebauungsplans in der Unreal Engine digitalisiert, als Klassen identifiziert und auf Blueprints übertragen. Hierbei wird die prozedurale Modellierung der Gebäude maßgeblich durch die bauliche Nutzung beeinflusst. Ausgewählte Planzeichen aus dem Bebauungsplan wurden dementsprechend digitalisiert und jeweils zusammenhängend unter einer Klasse gespeichert (siehe Abbildung 18).

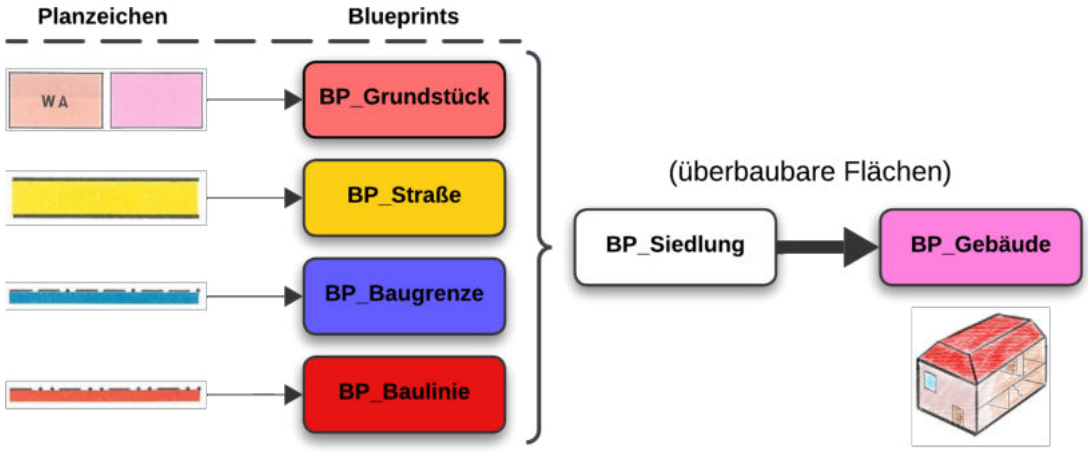


**Abbildung 18:** Übersicht zur Überführung der Planzeicheninhalte in Blueprints (eigene Darstellung, erstellt mit Lucidchart)

Die farblichen Markierungen für allgemeine Wohngebiete und Flächen für den Gemeinbedarf wurden als Polygonflächen digitalisiert. Da auf dem Plan noch keine zerlegten Flurstücke gekennzeichnet sind, werden die Gesamtflächen stellvertretend für Grundstücke dem Blueprint BP_Grundstück zugeordnet. Entsprechend werden die Flächen für die Berechnung der GRZ

und GFZ hinzugezogen. Um die überbaubaren Flächen festzulegen, in denen die Gebäude prozedural modelliert werden dürfen, werden die Baugrenzen flächenförmig als BP_Baugrenze verwaltet. Noch konkreter wird die Ausrichtung der Gebäude durch die Baulinien vorgegeben, die linienförmig unter BP_Baulinie vorliegen. Die Straßen wurden als linienförmige Splines unter BP_Straße nachmodelliert und dienen als Ausschlussflächen, auf denen keine Bebauung stattfinden darf.

Das gesamte Planungsgebiet wird flächenförmig unter dem Blueprint BP_Siedlung repräsentiert und greift auf alle innerhalb liegenden Objekte der digitalisierten Planzeichen zu. Dort werden die überbaubaren Flächen ausgewertet und prozedural generierte Objekte vom Blueprint BP_Gebäude unter Berücksichtigung der Maße der baulichen Nutzung und der Bauweise platziert.

### 4.1.2 PCG-Graphen

PCG wird workflowbasiert über Graphen gesteuert, welche den Blueprints zugeordnet werden können. Hierbei wurde für die Klassen BP_Siedlung, BP_Gebäude und BP_Straße jeweils ein gleichnamiger PCG-Graph erstellt. Innerhalb des PCG-Graphen kann, durch die Referenz, sowohl auf die geometrische Ausdehnung als auch auf die Attribute des Blueprints zugegriffen werden. Hierdurch lassen sich komplexere Sachverhalte abbilden, welche sich über die Parameter des Blueprints steuern lassen.

### 4.1.3 Datenstrukturen

Um komplexe Datenstrukturen abbilden zu können wurden sogenannte Structs und Primary Data Assets verwendet. Durch Structs werden die Attribute zusammenhängend gruppiert. Die Primary Data Assets dienen hingegen als Interface, aus denen Data Assets instanziiert werden können. In diesem Fall werden Structs als Attributtyp für die Attribute der Primary Data Assets verwendet, wodurch sich eine verschachtelte Struktur ergibt. Auf die Data Assets kann letztendlich innerhalb der PCG-Graphen zugegriffen werden.

Der Struct S_Building_Components besteht aus einem Feld an Static Mesh-Objekten, die in der Engine 3D-Modelle darstellen. Zwischen den Einträgen lässt sich eine Gewichtung festlegen mit welcher Häufigkeit ein Objekt ausgewählt wird. Zudem lässt sich die Position und Skalierung dieses Objekts transformieren und bei StaticMesh-Objekten mit offener Rückseite

diese schließen. In den Data Assets DA_SimpleBuilding und DA_RealisticBuilding, wird der Struct jedem Bestandteil eines Gebäudes zugeordnet (siehe Abbildung 19).

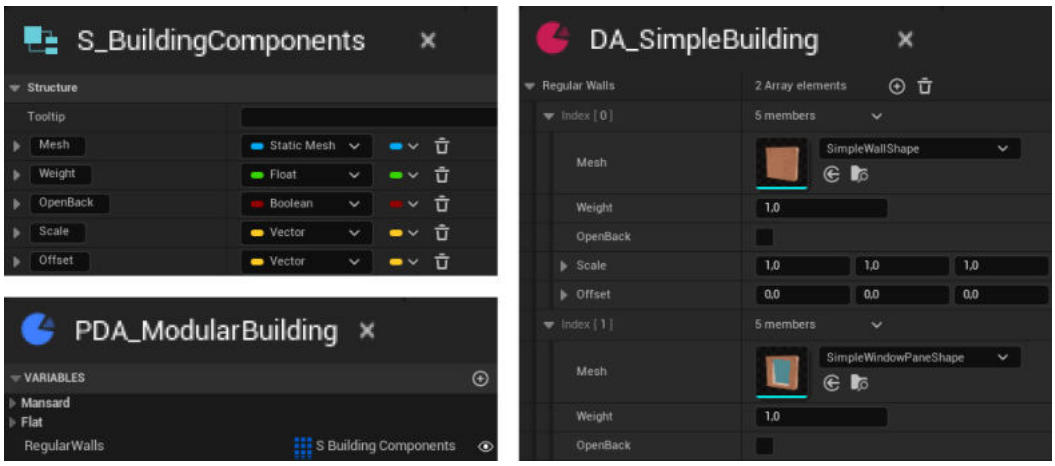


**Abbildung 19:** Struct S_Building_Components und Primary Data Asset PDA_ModularBuilding (links) und Data Asset DA_SimpleBuilding (eigene Darstellung)

Durch Enumerationen lassen sich Auswahlfelder für ein bestimmtes Attribut definieren. Dadurch stehen durch die Enumeration E_RoofTypes nur die Dachtypen Flachdach und Mansardflachdach zur Auswahl. Hingegen bezieht sich E_BuildingType auf die Form des Gebäudes, wobei zwischen Quadrat, Rechteck und einer undefinierten Form gewählt werden kann.

### 4.1.4 Klassen und Relationen

In einem UML-Klassendiagramm wird dargestellt in welcher Relation die definierten Blueprints, PCG-Graphen und Datenstrukturen zueinander stehen (siehe Abbildung 20).

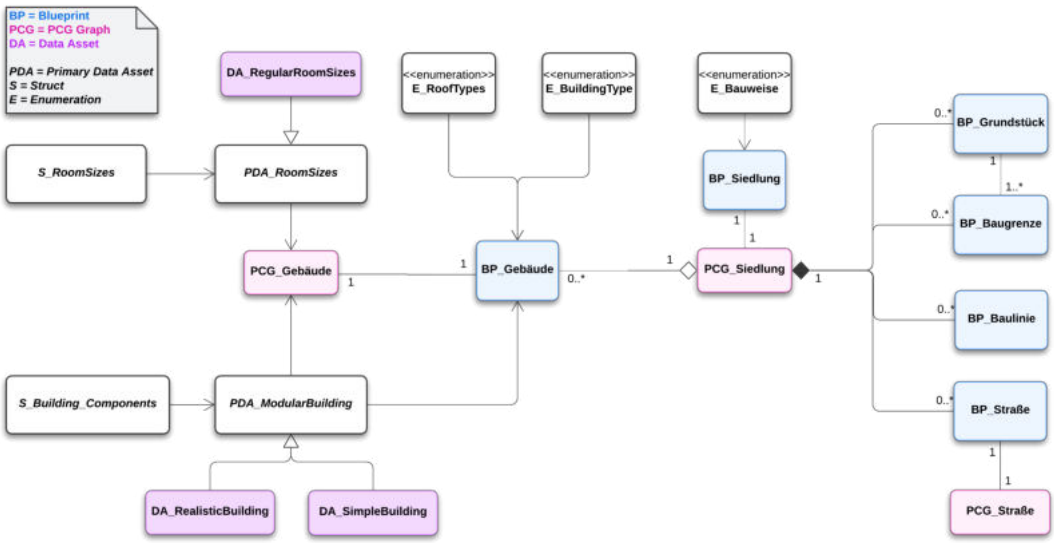


**Abbildung 20:** UML-Klassendiagramm (eigene Darstellung, erstellt mit Lucidchart)

Darüber hinaus wurden die Attribute und Funktionen der Klassen definiert (siehe Anlage 3). Im Schwerpunkt steht die Klasse PCG_Siedlung, welche sich aus den digitalisierten Planinhalten zusammensetzt. Hieraus werden die überbaubaren Flächen bestimmt, worauf Objekte der Klasse BP_Gebäude platziert werden. Die Gebäude werden ebenfalls prozedural modelliert und für das Erscheinungsbild, sowie die Innenraumgrößen durch Data Assets gestützt.

## 4.2 PCG-Workflow

Die prozedurale Generierung von Inhalten erfolgt in der Unreal Engine durch Punktdaten. Diese lassen sich im Debug-Modus der Engine als würfelförmige Geometrien darstellen. Dabei besitzen die Punktdaten geometrische Attribute zur Position, Rotation, Skalierung, sowie der minimalen und maximalen Ausdehnung. Zusätzlich werden den Punktdaten sachliche Attribute zugeordnet, darunter der RGB-Farbwert, die Intensität, die Steilheit und einen Seed. Diese Attribute lassen sich innerhalb einer Attributtabelle anzeigen (siehe Abbildung 21).

| Index | Position.X | Position.Y | Position.Z | Rotation.Roll | Rotation.Pitch | Rotation.Yaw | Scale.X | Scale.Y | Scale.Z |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2.190 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 2.490 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |
| 2 | 2.790 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |
| 3 | 3.090 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |
| 4 | 3.390 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |
| 5 | 3.690 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |
| 6 | 3.990 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |
| 7 | 4.290 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |
| 8 | 4.590 | -2.140 | 100 | -0 | 0 | 0 | 1 | 1 | 1 |

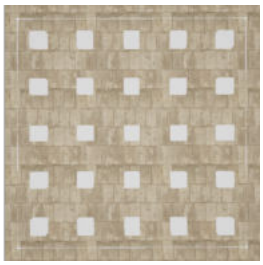

**Abbildung 21:** Attributtabelle zu den Punktdaten (links) und Darstellung im Debug-Modus (rechts) (eigene Darstellung)

Der PCG-Workflow der Unreal Engine baut dabei durchgehend auf diese Punktdaten auf, indem sie erzeugt, manipuliert und zum Output weiterer Daten verwendet werden. Dadurch bedingt setzt sich der PCG-Workflow aus vier grundlegenden Schritten zusammen (siehe Abbildung 22) (Epic Games, o.J. e):

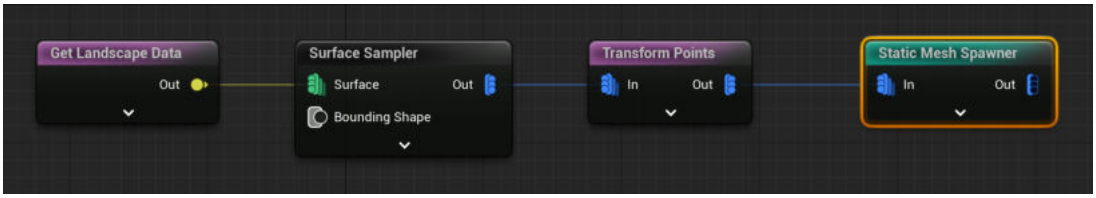


**Abbildung 22:** Grundlegender Aufbau eines PCG-Workflows (Epic Games, o.J. e)

(1) Input: Als Erstes werden die Eingangsdaten eingelesen. Dabei kann ausgewählt werden, auf welche Datenquelle zugegriffen werden soll. Unter anderem lassen sich Daten zum Terrain,

zu Splines oder zu bestimmten Actors einlesen. Der zurückgegebene Datentyp ist dabei davon abhängig welche Datenquelle eingelesen wurde. Für das Einlesen von Terraininformationen (Get Landscape Data) ist es erforderlich, dass im Level ein PCG-Volume-Actor platziert wird. Dabei handelt es sich um ein begrenzten Raum, der die Oberfläche des Terrains abtastet und die Daten im PCG-Graphen verwertet. Beim Einlesen von Splines ist es hingegen erforderlich, dass der PCG-Graph einen Bezug zu einem Actor mit einer Spline als Komponente hergestellt hat.

(2) Sampler: Im nächsten Schritt werden durch einen Sampler Punkte generiert. Der zu verwendende Sampler hängt dabei von den Eingangsdaten ab. Beispielsweise erfordert das Einlesen von Terraindaten einen Surface Sampler, der die Punkte entlang der Oberfläche des Terrains mit unterschiedlicher Intensität und Anordnung generiert. Beim Spline Sampler hingegen kann ausgewählt werden ob die Punkte entlang des Linienverlaufs oder innerhalb von geschlossenen Splines in rasterförmiger Anordnung generiert werden.

(3) Datenmanipulation: Anschließend steht eine große Bandbreite an Funktionen zur Verfügung, mit denen sich die räumlichen und sachlichen Attribute der Punktdaten verändern lassen. Durch die Funktion Transform Points lassen sich die Parameter zum Offset, zur Rotation oder Skalierung von Punkten verändern. Dabei kann ausgewählt werden ob der eingestellte Wert zufällig innerhalb eines Intervalls liegt oder fest vorgegeben ist. So kann sowohl regelbasiert als auch zufallsbasiert auf die Punktdaten zugegriffen werden. Es ist auch möglich, sich selbst überlappende Punkte zu entfernen und auszudünnen (Self Pruning). Dabei lässt sich auf Attribute zugreifen, sowie verändern oder entfernen. Durch Noises lassen sich zufällige Werte für ein gewähltes Attribut erzeugen. Mittels Filtern können Punkte herausgefiltert werden, die die Filterbedingung erfüllen. Auch lassen sich Differenzoperationen und Verschmelzungen durchführen. Des Weiteren stehen auch bedingte Anweisungen zur Verfügung.

(4) Output: Nachdem die Positionen der Punktdaten fertig verarbeitet wurden lassen sich, unter Berücksichtigung der Attribute, beliebige Actors platzieren. Durch den Static Mesh Spawner lassen sich beliebige 3D-Modelle als Static Mesh Actor generieren. Aber auch beliebige Actors lassen sich durch die Funktion Spawn Actor setzen. Dies wird in diesem Projekt genutzt um prozedural generierte Actors an prozedural generierten Positionen zu platzieren.

Dieser Workflow ist grundsätzlich sehr simpel gehalten, kann jedoch durch das visuelle Blueprint-Skripting mitunter sehr komplex und unübersichtlich ausfallen. Daher wurden in diesem Pro-

jekt sogenannte Subgraphs und Loop-Subgraphs verwendet. Dabei handelt es sich um eigene PCG-Graphen, die einzelne Nodes gruppieren und als eigene Nodes in anderen PCG-Graphen aufgerufen werden können (Epic Games, o.J. e). Die Subgraphs können auch als Loops (dt. Schleifen) verwendet werden.

## 4.3 Generierung der Gebäude

Bei der prozeduralen Modellierung wird im Regelfall so vorgegangen, dass komplexere urbane Objekte hierarchisch in kleinere Bestandteile zerlegt werden. Daher wurden als Erstes die Bestandteile eines Gebäudes identifiziert (siehe Abbildung 23).

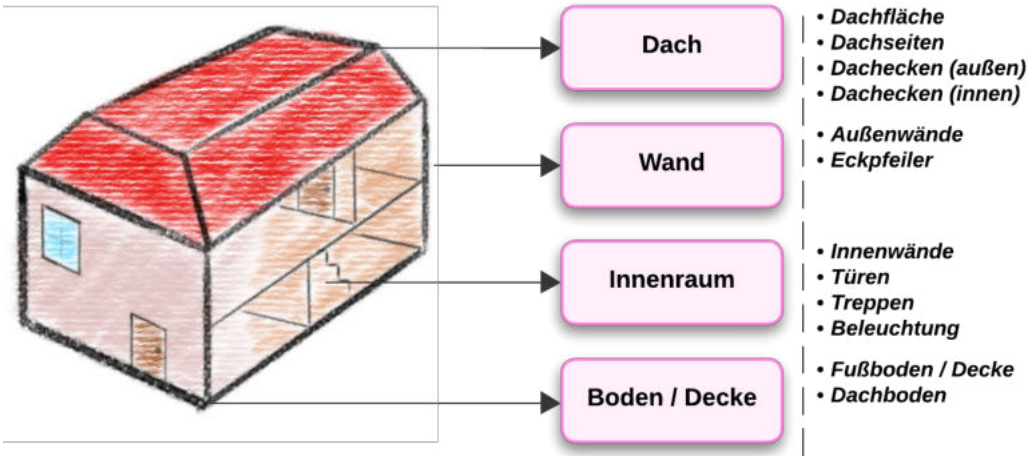


**Abbildung 23:** Übersicht zu den Bestandteilen eines Gebäudes (eigene Darstellung, erstellt mit Lucidchart)

Grob lässt sich ein Gebäude in flächenförmige und räumliche Strukturen aufteilen. Dazu gehören Böden und Decken eines Gebäudes, sowie Wände und Dächer. In Architekturmodellen wird zudem noch eine Modellierung der Innenräume vorgenommen. Diese Strukturen lassen sich wiederum in noch kleinere Bestandteile unterteilen, die bei der Gebäudegenerierung ineinandergreifen.

### 4.3.1 Punktraster

Die prozedurale Generierung beruht auf einer Spline, die in dem Blueprint BP_Gebäude gemeinsam mit einem PCG-Graphen verwaltet wird. Die Spline wurde so konfiguriert, dass sie sich wie ein geschlossenes Polygon verhält. Im PCG-Graph PCG_Gebäude werden die Daten der Spline eingelesen (Get Spline Data) und innerhalb der Polygonfläche Punkte mit regelmäßigem Abstand generiert (Spline Sampler) (siehe Abbildung 24). Die Dichte des Punkteras-

ters lässt sich parametergesteuert anpassen und dient als Datengrundlage für die nachfolgenden Auswertungen.


**Abbildung 24:** Erstellung des Punkterasters: PCG-Workflow (links) und Debug-Darstellung (rechts) (eigene Darstellung)

### 4.3.2 Bodenflächen

Damit die Punktdaten über mehrere Stockwerke hinweg generiert werden können, werden die Anzahl und relative Höhe der Vollgeschosse als Parameter benötigt. Für die Berechnung der absoluten Höhe der Punkte kommen Loops zum Einsatz, bei denen der Schleifenzähler mit der relativen Höhe multipliziert wird. Hierdurch wird das Punkteraster nach oben für jedes Stockwerk kopiert (siehe Abbildung 25).

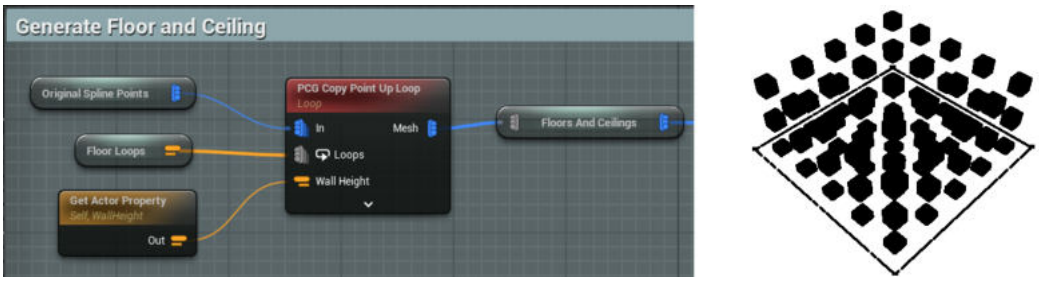


**Abbildung 25:** Erstellung der Punktdaten für Böden und Decken: PCG-Workflow (links) und Debug-Darstellung (rechts) (eigene Darstellung)

Um die Böden und Decken zu erhalten werden die Punktdaten so skaliert, dass eine zusammenhängende, geschlossene Fläche entsteht (siehe Abbildung 26).

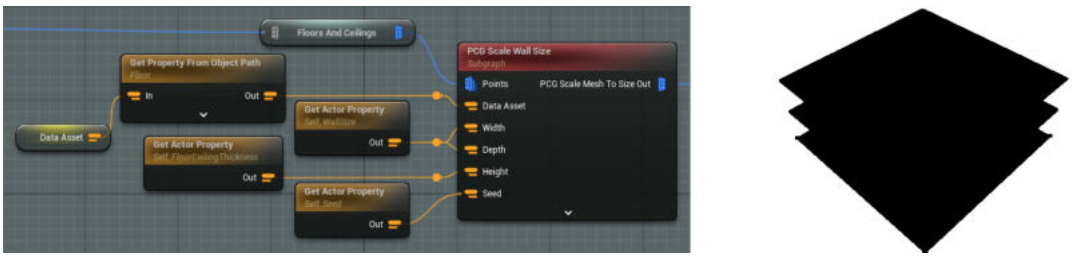


**Abbildung 26:** Skalierung der Bodenflächen: PCG-Workflow (links) und Debug-Darstellung (rechts) (eigene Darstellung)

### 4.3.3 Wandflächen

Im nächsten Schritt werden mithilfe des Punkterasters die Außenwände und Eckpfeiler für das Gebäude erzeugt. Hierfür wurde das Punkteraster dupliziert und jeweils in alle vier Himmelsrichtungen verschoben (Transform Points). Durch Differenzbildung (Difference) bleiben nur noch die äußeren Ränder des Punkterasters übrig, welche als Grundlage für die Wände dienen (siehe Abbildung 27). Weiterführend werden auch die Ecken für die Eckpfeiler extrahiert.

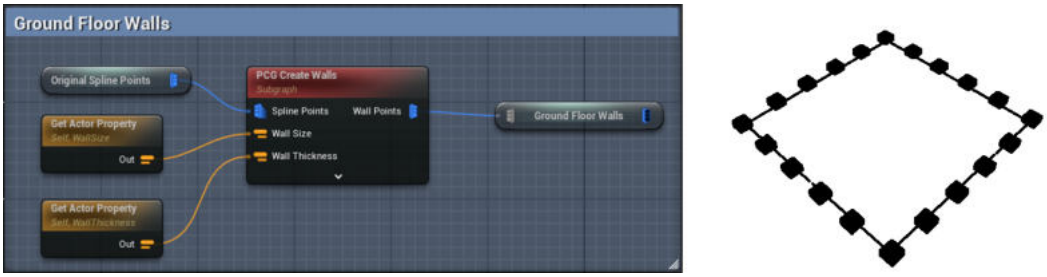


**Abbildung 27:** Erzeugung der Wand- und Eckpunkte: PCG-Workflow (links) und Debug-Darstellung (rechts) (eigene Darstellung)

Die zugrundeliegenden Punkte werden ebenfalls für jedes Stockwerk kopiert und zu zusammenhängenden Flächen skaliert. Im Erdgeschoss werden zufallsbasiert ein oder mehrere Punkte ausgewählt, an denen eine Tür anstelle einer Wand generiert werden soll (siehe Abbildung 28). Diese Bereiche wurden von den Wandflächen herausgetrennt. Für die Tür wurde ein Blueprint mit eigener Animation erstellt mit der interagiert werden kann (BP_DualDoor).

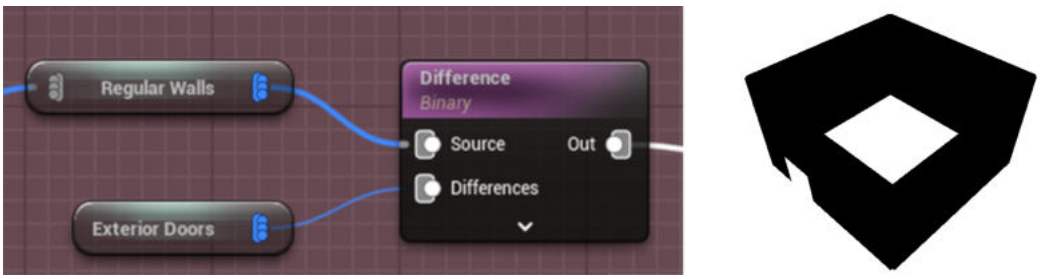


**Abbildung 28:** Auftrennung zwischen Wandflächen und Türen: PCG-Workflow (links) und Debug-Darstellung (rechts) (eigene Darstellung)

### 4.3.4 Dachformen

Im Blueprint BP_Gebäude kann zwischen einem Flachdach und einem Mansardflachdach als Dachform gewählt werden. Über eine bedingte Anweisung im PCG-Graphen wird die ausgewählte Dachform generiert. Beim Flachdach werden im obersten Geschoss die Dachkanten und -ecken erzeugt. Die Generierung eines Mansardflachdachs verläuft deutlich komplexer. Zum

einen muss neben der Traufhöhe auch die Firsthöhe eines Gebäudes beachtet werden. Die Differenz beider Werte fließt letztendlich in die Dachhöhe ein. Zum anderen setzt sich das Mansardflachdach aus Dachseiten, -innen und -außenecken, sowie einer -platte zusammen (siehe Abbildung 29).

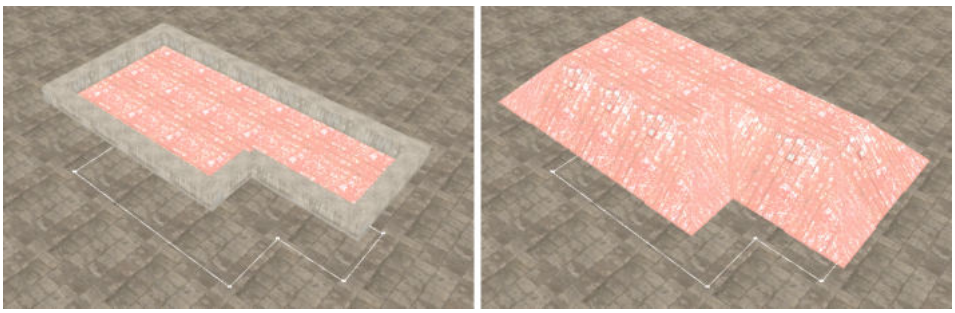

**Abbildung 29:** Texturiertes Flachdach (links) und Mansardflachdach (rechts) (eigene Darstellung)

### 4.3.5 Texturierung

Während der Prozessierung erhalten die Punktdaten eine Referenz zu einer Static Mesh-Komponente aus dem Data Asset. Durch den Static Mesh Spawner kann auf diese Referenz zugegriffen und darüber das passende Static Mesh instanziiert werden. Dabei steht die Möglichkeit zur Verfügung zwischen verschiedenen Data Assets zu wechseln, sodass zwischen einem einfach texturierten Gebäude (DA_SimpleBuilding) und einem realistisch texturierten Gebäude (DA_RealisticBuilding) gewählt werden kann (siehe Abbildung 30).

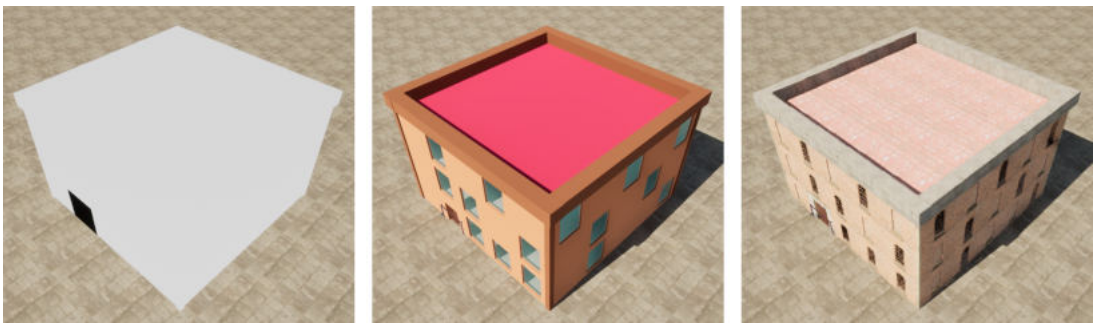

**Abbildung 30:** Fertiges Gebäude als PCG Debug-Darstellung (links), DA_SimpleBuilding (mitte) und DA_RealisticBuilding (rechts) (eigene Darstellung)

### 4.3.6 Innenräume

Damit ein architekturähnliches 3D-Modell solcher Gebäude entsteht wurden, zusätzlich zu den außen sichtbaren Bereichen, auch Innenräume modelliert.

Bei Gebäuden mit zwei oder mehr Stockwerken war es erforderlich diese durch Treppen miteinander zu verbinden, damit jedes dieser Stockwerke betreten werden kann. Dabei wurde, analog zur zufallsbasierten Auswahl der Türen, eine zufällige Gruppe von Bodenpunkten für das unterste Stockwerk gewählt. Diese Punkte wurden für jedes Stockwerk nach oben kopiert und stellen die Bereiche dar, an denen anstelle eines Fußbodens ein Treppenhaus erzeugt wird. Die Bereiche der Fußböden und Decken wurden dort herausgetrennt.

Übersteigt das Gebäude eine festgelegte Breite, werden innerhalb des Gebäudes Innenräume generiert. Hierbei wird ein weiteres Punktraster mit niedrigerer Punktdichte erzeugt. Von den verfügbaren Punkten wird eine Teilmenge durch Zufall ausgewählt und unterschiedliche Intensitätswerte zugeordnet. Im regulären Punkteraster werden die Intensitätswerte auf die Fläche übertragen und gruppiert. Anschließend werden die gruppierten Punktenmengen iteriert, die Wände extrahiert und auf die Fläche skaliert. Durch zufällige Punktauswahl werden Innentüren platziert (siehe Abbildung 31).

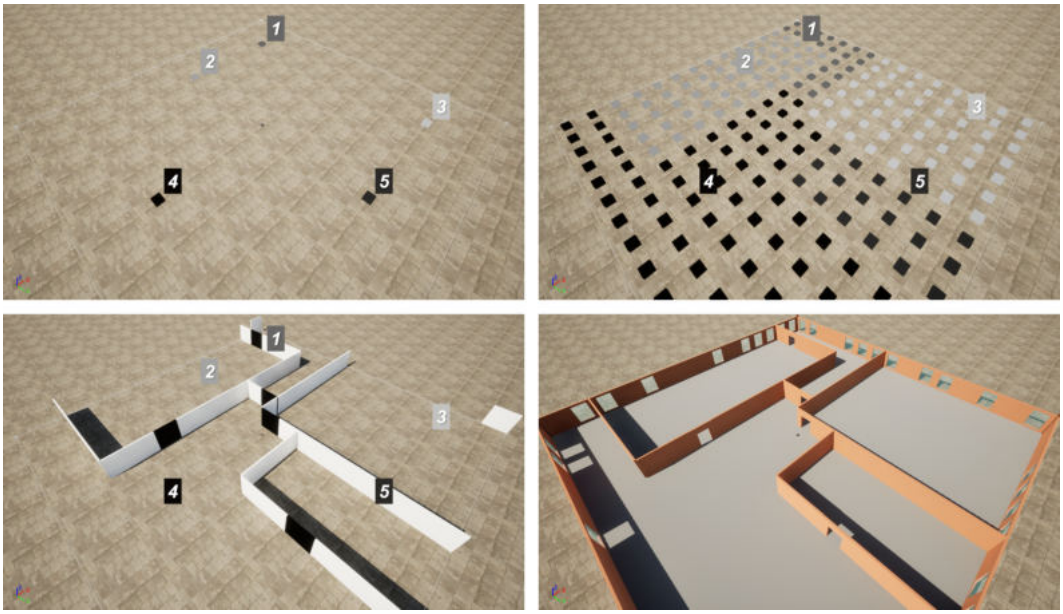


**Abbildung 31:** Erzeugung der Innenräume: ausgewählte Punkte mit Intensitätswerten (oben links), Übertragung auf ein reguläres Punkteraster (oben rechts), Erzeugung der Innenwände (unten links) und DA_SimpleBuilding (unten rechts) (eigene Darstellung)

Damit das Gebäude mit seinen Innenräumen ausreichend beleuchtet ist, wird für jeden Innenraum ein Actor der Klasse Point Light platziert. Da es bei manchen Gebäuden keine Innenräume gibt muss eine weitere Lichtquelle für den Gesamtraum platziert werden. Für die Platzierung wurde der Zentroid dieser Räume verwendet, sodass sich die Lichtquelle genau mittig im Raum befindet (siehe Abbildung 32).


**Abbildung 32:** Platzierung der Beleuchtung: Ausdehnung eines Innenraumes (links) und Darstellung eines Point Light-Actors (rechts) (eigene Darstellung)

## 4.4 Generierung des Siedlungsgebiets

Als Nächstes wurden die digitalisierten Planinhalte ausgewertet, um die überbaubaren Flächen zu bestimmen (siehe Abbildung 33).

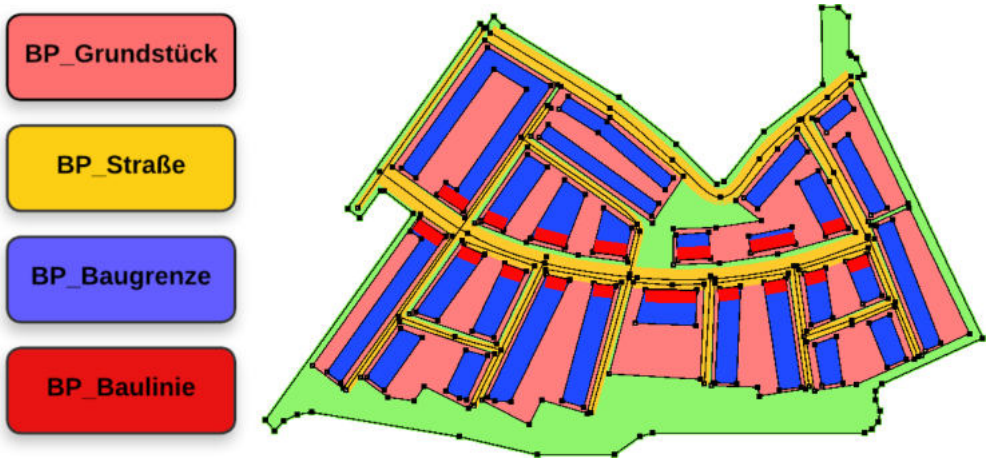


**Abbildung 33:** Übersicht zu den digitalisierten Bestandteilen des Siedungsgebiets (eigene Darstellung)

Dabei wurde bei der Generierung auf die Grundidee von Zahid et al. aufgebaut, um Einschränkungsregeln für den digitalisierten Bebauungsplan zu definieren (Zahid et al., 2024).

### 4.4.1 Einschlussflächen

In der Bauleitplanung wird gesetzlich vorgegeben, dass die baulichen Anlagen innerhalb der Baugrenze und entlang der Baulinie errichtet werden. Diese Bereiche werden als Einschlussflächen behandelt. Dabei muss berücksichtigt werden, dass es zu keinen Überlappungen innerhalb dieser Flächen kommt. Entsprechend wurden die Bereiche entlang der Baulinie von dem Be-

reich innerhalb der Baugrenze herausgetrennt. Dadurch ist klar definiert in welchen Bereichen sich an die Baulinien orientiert wird und wo ausschließlich an die Baugrenzen.

Umgekehrt werden Bereiche, an denen nicht bebaut werden darf, darunter Straßen, als Ausschlussflächen gewertet. Durch die Differenzfunktion werden die Ein- und Ausschlussflächen miteinander verschnitten, wodurch sich der überbaubare Bereich ergibt.

### 4.4.2 Positionsbestimmung

Im nächsten Schritt werden die Positionen und Ausdehnungen der zu platzierenden Gebäude ermittelt. Auf Grundlage der Einschlussflächen wird ein Punkteraster erzeugt. Durch Angabe der Gebäudeseiten und der Seitenabstände wird überprüft wie die Punkte angeordnet werden müssen, damit die größtmögliche Anzahl an Gebäuden platziert werden kann. Dabei wird Wert darauf gelegt, dass die zu platzierenden Gebäude später innerhalb einer Flucht liegen. Zwischen diesen Kandidaten wird zufallsbasiert eine Punktgruppe ausgewählt, die für die Positionierung weiterverwendet wird (siehe Abbildung 34). Dabei kann über eine bedingte Anweisung zwischen einer offenen und geschlossenen Bauweise gewechselt werden. Im Falle der geschlossenen Bauweise wird der Seitenabstand ignoriert und die Gebäude direkt nebeneinander platziert. Jedoch wird zwischen den Gebäuden der Baulinie und der Baugrenze ein undefinierter Abstand beibehalten. Das liegt daran, dass die Gebäude in den Bereichen der Baulinie und Baugrenze seperat generiert und erst später zusammengeführt werden.

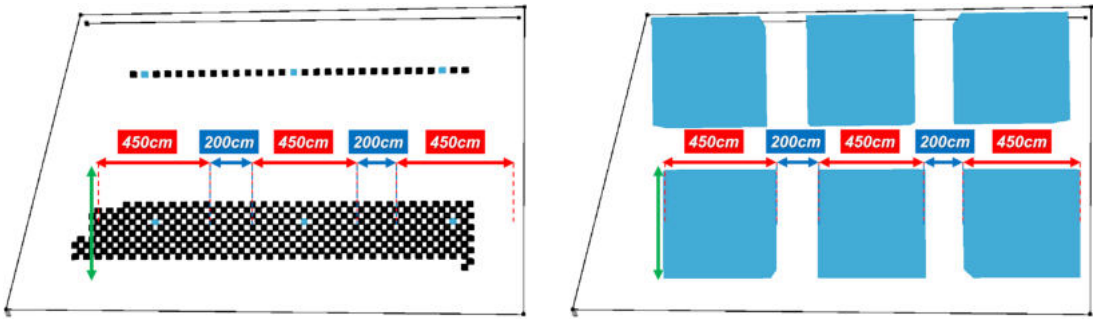


**Abbildung 34:** Berechnung des Punktegitters (links) und Platzierung der Gebäude (rechts) am Beispiel einer Länge und Breite von 4,5 m sowie einem Seitenabstand von 2 m (eigene Darstellung)

Damit auch die Maße der baulichen Nutzung bei der Siedlungsgenerierung berücksichtigt werden wird, für alle Baugrenzen die innerhalb eines Grundstücks liegen, ein Flächenverhältnis berechnet. Wird die angegebene GRZ bzw. GFZ überschritten, werden die Kandidaten zufallsbasiert gefiltert, sodass die Anzahl der zu generierenden Gebäude vermindert wird. Dabei wer-

den nicht die einzelnen Gebäude- oder Geschossflächen berücksichtigt, sondern die Flächen der Baugrenzen. Es soll hierbei lediglich eine Approximation erfolgen.

Anschließend werden die gefilterten Punktdaten auf die Erdoberfläche projiziert. Dies geschieht über eine sogenannte World Ray Hit Query. Hierbei wird eine Gerade senkrecht zur Erdoberfläche für die Berechnung verwendet. Die Punktdaten werden dort projiziert, wo es zum Schnitt zwischen der Geraden und der Ebene kommt (siehe Abbildung 35). Zum Schluss wird auf Grundlage der Punktdaten jeweils ein Actor der Klasse BP_Gebäude erzeugt. Hierbei fließen die Angaben zu den Seitenlängen der Gebäude und die Anzahl der Vollgeschosse in die Erzeugung ein. Für jeden Actor wird individuell ein Gebäude generiert und an den vorgesehenen Orten platziert.

**Abbildung 35:** Übersicht zu den projizierten Positionen für die Gebäude (eigene Darstellung)

## 4.5 Navigation und Interaktion

Um während der Laufzeit mit der Anwendung interagieren zu können, werden sowohl eine intuitive Steuerung als auch eine übersichtliche Benutzeroberfläche benötigt. Diese sollten möglichst auf den Funktionsumfang des Programms zugeschnitten sein (Herczeg, 2018). Dabei wird versucht, die Steuerung eines Avatars innerhalb einer Game Engine mit einer menübasierten Steuerung zusammenzuführen.

Hierfür wurden zwei verschiedene Modi zur Steuerung implementiert. Der Spielemodus, der ausschließlich zur Navigation innerhalb der Anwendung dient und der Editormodus, der Funktionen zur Editierung des Siedlungsgebietes und der Gebäude bereitstellt.

### 4.5.1 Avatar

Im Spielemodus wird ein spielbarer Avatar verwendet, der sich durch Maus- und Tastatureingaben steuern lässt. Der Avatar wurde aus dem Third-Person-Template der Unreal Engine importiert und für den Zweck dieser Anwendung erweitert (Epic Games, o.J. h).

Die Steuerung orientiert sich an modernen Computerspielen, beispielsweise Minecraft (Mojang, 2023). Der Avatar bewegt sich mit einer voreingestellten Animation in Richtung der Mauszeigerposition und in Abhängigkeit von der Kamera. Unter Berücksichtigung der Physik innerhalb der Engine bewegt er sich auf dem Boden entlang. Um die Bewegung flexibler zu gestalten, wurde zusätzlich eingestellt, dass der Charakter springen, fliegen und schneller rennen kann. Außerdem kann der Charakter, durch das Öffnen und Schließen von Türen, mit bestimmten Objekten in der Umgebung interagieren. Zudem steht eine Respawn-Funktion zur Verfügung, mit der die Position des Avatars zurückgesetzt werden kann.

Hierfür ist es erforderlich, entsprechende Eingabebefehle (Input Actions) zu definieren und eine Tastenbelegung über das Input Mapping Context zuzuweisen. Die Anwendungslogik für die Eingabebefehle wird innerhalb des Blueprints BP_Character implementiert.

### 4.5.2 Kameras

Virtuelle Kameras stellen in Computerspielen und vergleichbaren Anwendungen eine wichtige Schnittstelle zwischen Anwender und System dar. Um eine große Bandbreite an Informationen zu vermitteln, ohne den Nutzer zu belasten, sollte die Kameraperspektive, dem Anwendungszweck entsprechend, sorgfältig gewählt werden (Yannakakis et al., 2010).

Um mehrere Kameraperspektiven zu ermöglichen, wurden neben der bereits implementierten Third-Person-Perspektive zwei weitere Kameras hinzugefügt. Eine Kamera befindet auf Blickhöhe des Avatars, wodurch eine First-Person-Perspektive ermöglicht wird (siehe Abbildung 36).

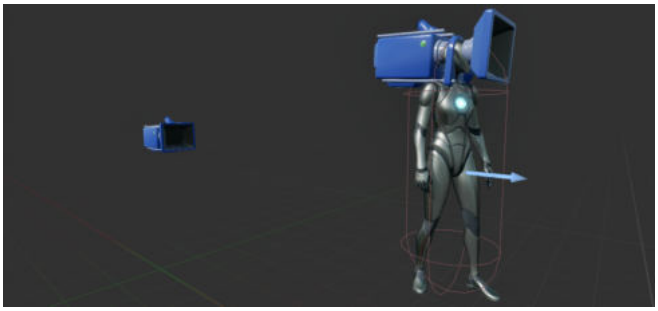

**Abbildung 36:** Kameras für Third- (links) und First-Person-Perspektive (rechts) (eigene Darstellung)

Zwischen beiden Perspektiven kann während der Laufzeit gewechselt werden. Eine weitere Kamera befindet sich ohne Rotierbarkeit und mit großen Abstand oberhalb des Avatars und fängt das Spielgeschehen aus einer Top-Down-Perspektive ein.

### 4.5.3 Benutzeroberfläche

Durch die Einbindung eines Widgets werden visuelle Informationen bereitgestellt, die die Handhabung der Anwendung erleichtern. Hierfür wurden Icons erstellt, die zur Barrierefreiheit des Programms beitragen sollen. Aufgrund der symbolischen Darstellung kann auf eine mehrsprachige Unterstützung verzichtet werden. Das Widget WB_Menu besteht aus mehreren Untermenüs, die sich nach Bedarf ein- oder ausblenden lassen (siehe Abbildung 37).


**Abbildung 37:** Übersicht zum Widget WB_Menu mit Editormenü (links), Hilfemenü (rechts) und Pausemenü (oben) (eigene Darstellung)

Über Schaltflächen wird eine Interaktion zwischen Computer und Nutzer ermöglicht. Hierfür stehen durchgehend zwei verschiedene Schaltflächen am oberen Bildschirmrand zur Verfügung. Der rote Druckknopf mit dem Stift- oder Pfeilsymbol löst einen Wechsel zwischen dem Spiele- und dem Editormodus aus. Durch den grünen Druckknopf mit dem Fragezeichensymbol wird das Hilfemenü ein- oder ausgeblendet. Im Hilfemenü wird die Tastenbelegung und die Maussteuerung erklärt. Zudem wird beschrieben, welche Bedeutung die darstellten Icons besitzen.

Im Editormodus steht ein Editormenü zur Verfügung. Dieses besteht aus einer Druckknopfleiste mit Editierfunktionen und einer Attributtabelle. Hierbei werden die weniger relevanten

Attribute unter einem Reiter ausgeblendet und lassen sich bei Bedarf über eine Schaltfläche einblenden. Dadurch werden die Nutzer nicht mit einer zu hohen Informationsdichte überfordert. Die Attributwerte lassen sich entweder in einem frei editierbaren Textfeld eintragen oder in einem Drop-Down-Menü auswählen.

Beim Drücken der Escape-Taste werden die Menüs geschlossen. Falls keine Menüs geöffnet sind, wird ein Pausemenü eingeblendet. Hierüber kann die Anwendung beendet werden.

### 4.5.4 Editierfunktionen

Im Editormodus wechselt die Perspektive automatisch zu einer Top-Down-Perspektive. Die Kamera wird hierbei ausschließlich durch das Scrollrad bewegt, indem auf das Gebiet hinein-, bzw. hinausgezoomt werden kann.

Abhängig von der ausgewählten Schaltfläche der Druckknopfleiste im Editormenü verändert sich die Funktionalität der linken Maustaste. Die Anwendungslogik wird im Blueprint BP_PlayerController verwaltet. Die Druckknopfleiste besteht aus insgesamt sechs Schaltflächen (siehe Abbildung 38).

**Abbildung 38:** Schaltflächen: Selektion zurücksetzen, Gebäude selektieren, hinzufügen, entfernen, Grundfläche editieren, Änderungen übernehmen (von links nach rechts aufgezählt) (eigene Darstellung)

Durch die Schaltfläche „Gebäude selektieren“ wird die grafische Selektion aktiviert. Der erste Mausklick erzeugt ein Objekt vom Blueprint BP_SelectorBox, der alle Actors der Klasse BP_Gebäude abfragt, die innerhalb der Box liegen. Beim zweiten Mausklick wird die Box entfernt. Hierbei werden alle Gebäude, die zu diesem Zeitpunkt in der Box enthalten waren, im Hintergrund gespeichert. Die grafische Selektion lässt sich zudem per Knopfdruck aufheben und die gespeicherten Actors zurücksetzen.

Die selektierten Gebäude können über eine Schaltfläche gelöscht werden. Zudem können die Attributwerte innerhalb der Attributtabelle editiert werden. Durch Bestätigung über die Enter-Taste oder einer Schaltfläche, lassen sich die getätigten Eingaben aus der Attributtabelle übernehmen.

Wird genau ein Gebäude selektiert, kann die Grundfläche editiert werden. Beim ersten Mausklick wird der am nächsten liegende Punkt der zugrundeliegenden Spline-Komponente des Gebäudes ausgewählt. Beim zweiten Mausklick wird die Position des Punktes verschoben.

Ist die Schaltfläche „Gebäude hinzufügen“ eingeschaltet, lassen sich per Mausklick neue Objekte der Klasse BP_Gebäude instanziieren. Durch Klicken des Mauszeigers wird an der Zeigerposition ein Gebäude mit zufälliger Rotation und voreingestellter Größe auf der Erdoberfläche erzeugt.

Des Weiteren steht in der Attributtabelle eine Schaltfläche für die Erstellung eines zufälligen Seeds zur Verfügung. Der gewünschte Seed kann aber auch im benachbarten Textfeld eingegeben werden (siehe Abbildung 39). Hierbei wird die PCG-Komponente von BP_Siedlung mit einem neuen Seed generiert, wodurch das vorhandene Gebiet entfernt und durch ein neu generiertes Siedlungsgebiet ersetzt wird.


**Abbildung 39:** Schaltfläche und Textfeld zur Erstellung eines Seeds (eigene Darstellung)

## 4.6 Visualisierung und Rendering

Um größere Datenmengen in der Engine performant und optisch ansprechend zu rendern, stellt die Unreal Engine verschiedene Funktionen zur Leistungsoptimierung zur Verfügung.

### 4.6.1 Nanites

In diesem Projekt wurde die Nanite-Technologie der Unreal Engine verwendet. Dieses Feature ermöglicht es, Static Mesh-Objekte als Dreieck-Meshes mit anpassbarer Detailstufe zu komprimieren und zu verwalten. Die Unreal Engine steuert automatisch, basierend auf den Voreinstellungen, wann und wie zu welcher Detailstufe gewechselt wird. Für weiter entfernte Objekte wird eine grob aufgelöste Darstellung des Dreieck-Meshes verwendet, wodurch die Engine beim Rendern des gesamten Bebauungsplans eine hohe Performance beibehält (siehe Abbildung 40) (Epic Games, o.J. d). Nur für transparente Flächen, darunter Fenster, wurden keine Nanites verwendet da es dort zu Konflikten kommt.

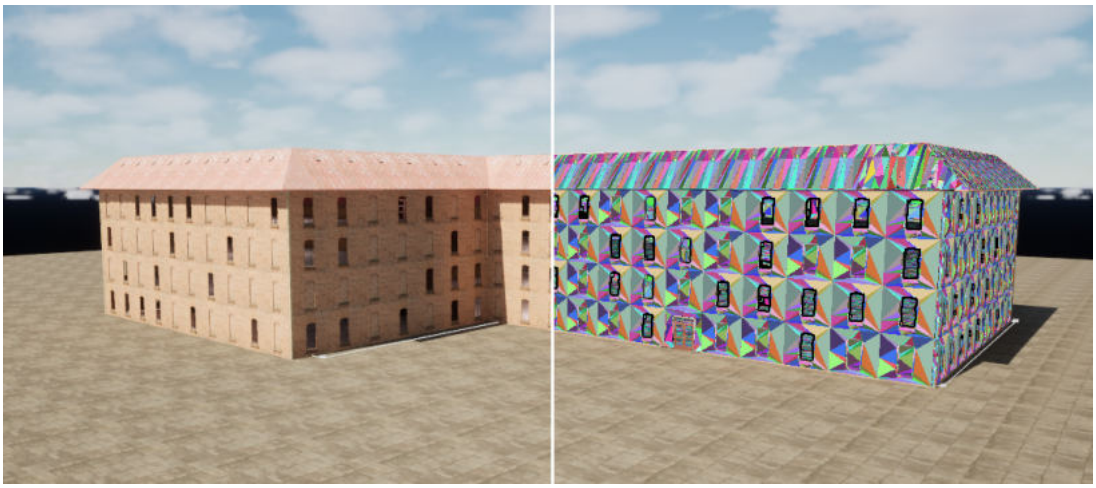

**Abbildung 40:** Texturiertes Gebäude (links) und Nanite-Darstellung durch Dreieck-Meshes (rechts) im Vergleich (eigene Darstellung)

### 4.6.2 Lumen

Um eine realistische Beleuchtungs- und Schattenverhältnisse in virtuellen Umgebungen in Echtzeit zu erzeugen, verwendet die Unreal Engine Lumen als Global Illumination (Epic Games, o.J. b). Mit Global Illumination werden die Interaktionen von Lichtstrahlen mit der Umgebung berechnet und visualisiert. Mittels physikalischer Berechnungen lassen sich unter anderem Reflexionen und Streuungen simulieren (Ritschel et al., 2012). Dadurch sieht die Beleuchtung innerhalb und außerhalb der Gebäude möglichst realistisch aus.

# 5 Evaluation

## 5.1 Projektergebnisse

Als Ergebnis liegt ein interaktives 3D-Modell des Baugebiets „In der Steiniger Heide“ in der Stadt Osnabrück vor. Die Anwendung kann sowohl innerhalb der Entwicklungsumgebung der Unreal Engine, als auch als kompiliertes Spiel ausgeführt werden. Als Datengrundlage für das Modell fließen die über Cesium bereitgestellten amtlichen Geodaten, sowie der georeferenzierte Planausschnitt mit dem prozedural generierten Siedlungsgebiet zusammen. Dabei werden die prozedural modellierten Gebäude so platziert, dass die Festsetzungen des Bebauungsplans annähernd eingehalten werden (siehe Abbildung 41).

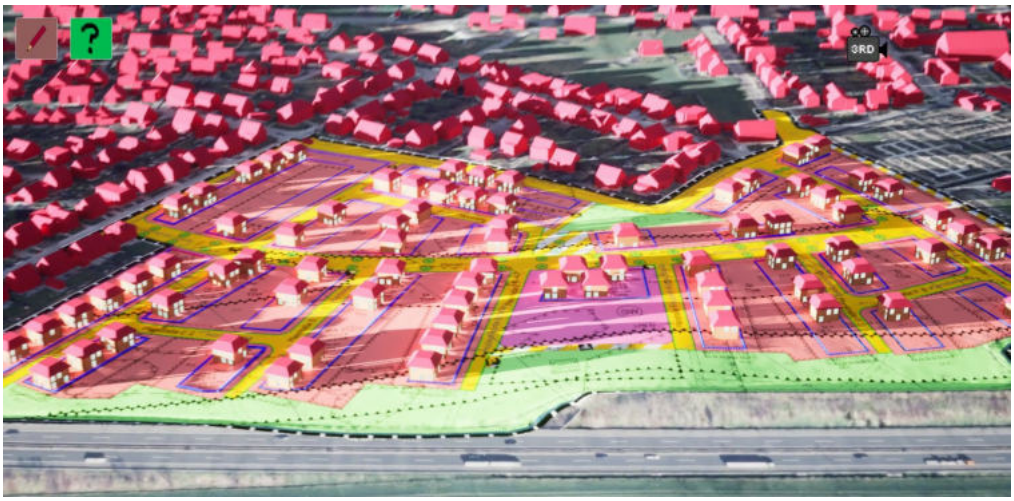

**Abbildung 41:** Projektergebnis zum prozedural generierten Gebiet, nach Vorlage des Bebauungsplans Nr. 629 (eigene Darstellung)

Der Nutzer kann das 3D-Modell aus verschiedenen Perspektiven erkunden. Das Gebiet lässt sich durch einen Avatar sowohl aus der Sicht eines Fußgängers betrachten, als auch über einen Flugmodus aus allen Perspektiven beobachten. Im Editormodus kann das Siedlungsgebiet aus der Vogelperspektive eines Stadtplaners begutachtet werden. Hierbei lassen sich sowohl das Siedlungsgebiet als auch einzelne Gebäude parametergesteuert bearbeiten und anpassen. Die einzelnen Gebäude sind begehbar und verfügen über ausgeleuchtete Innenräume, die durch Treppen miteinander verbunden sind (siehe Abbildung 42).

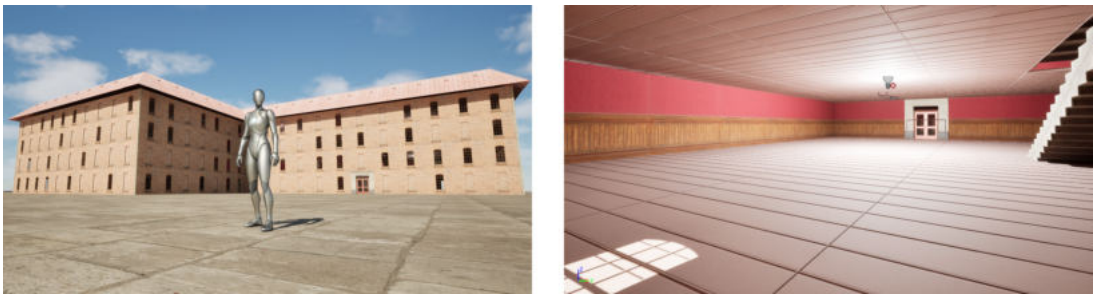

**Abbildung 42:** Projektergebnis zum prozedural modellierten Gebäude (links) mit beleuchteten Innenräumen, einschließlich Treppen (rechts) (eigene Darstellung)

## 5.2 Bewertungskriterien

Um die Gebrauchstauglichkeit des Projektergebnisses einschätzen und langfristig verbessern zu können ist es erforderlich die Nutzererfahrungen durch Evaluation zu erfassen. Zunächst werden in der Regel die Bewertungskriterien aufgestellt welche das Programm erfüllen muss. Im nächsten Schritt werden Leistungsstandards definiert anhand derer gemessen wird ob die

zugehörigen Kriterien erfüllt wurden. Dies kann durch nutzerbasierte Tests und Fragebögen erfolgen. Abschließend werden die Ergebnisse bewertet und beurteilt (Baumgartner, 1999).

### 5.2.1 System Usability Scale

Zur Evaluation des Projektergebnisses wurden Black-Box-Tests mit den Probanden durchgeführt. Dabei besaßen die Probanden keinerlei Vorkenntnisse über das System und erhielten keinen Einblick in den Quellcode (Mohan et al., 2010). Nur eine Benutzerhilfe stand den Probanden zur Verfügung, in der sie über die Grundfunktionen der Anwendung unterrichtet wurden (siehe Anlage 5).

Anschließend wurde den Probanden ein Fragebogen zur Verfügung gestellt. Zur Bewertung der Gebrauchstauglichkeit existieren bereits standardisierte Fragebögen. Im Rahmen dieser Evaluation wurde der Fragebogen zur System Usability Scale (kurz: SUS) verwendet. Mit insgesamt zehn Fragestellungen erfordert dieser Fragebogen wenig Zeit und eignet sich zur Bewertung beliebiger Software (Brooke, 1996). Die Fragestellungen liegen im Original in englischer Sprache vor und wurden nach Rummel ins Deutsche übersetzt (siehe Tabelle 4) (Rummel, 2013).

**Tabelle 4:** Fragen zum System Usability Scale mit deutscher Übersetzung (Rummel, 2013)

| Nr. | Fragestellung | Deutsche Übersetzung nach Rummel |
|---|---|---|
| 1 | I think that I would like to use this system frequently. | Ich denke, dass ich das System gerne häufig benutzen würde. |
| 2 | I found the system unnecessarily complex. | Ich fand das System unnötig komplex. |
| 3 | I thought the system was easy to use. | Ich fand das System einfach zu benutzen. |
| 4 | I think that I would need the support of a technical person to be able to use this system. | Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können. |
| 5 | I found the various functions in this system were well integrated. | Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert. |
| 6 | I thought there was too much inconsistency in this system. | Ich denke, das System enthielt zu viele Inkonsistenzen. |
| 7 | I would imagine that most people would learn to use this system very quickly. | Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System sehr schnell lernen. |
| 8 | I found the system very cumbersome to use. | Ich fand das System sehr umständlich zu nutzen. |
| 9 | I felt very confident using the system. | Ich fühlte mich bei der Benutzung des Systems sehr sicher. |
| 10 | I needed to learn a lot of things before I could get going with this system. | Ich musste eine Menge lernen, bevor ich anfangen konnte das System zu verwenden. |

Aufgrund des sprachlichen Unterschieds ist auf geringfügige Abweichungen bei der Bewertung hinzuweisen (Gao et al., 2020).

Für jede Fragestellung wird, anhand einer Likert-Skala, eine Punktzahl von 1 bis 5 vergeben. Bei der Auswertung des Fragebogens werden alle Fragestellungen in ungerade und gerade Nummern unterteilt. Bei ungeraden Nummern wird der Wert 1 von der Punktzahl abgezogen, während bei geraden Nummern die Punktzahl vom Wert 5 abgezogen wird (Gutiérrez & Rojano-Cáceres, 2020).

$$\text{Score}_{p_1,p_3,p_5,p_7,p_9} = \text{Scale Score} - 1 \tag{3}$$

$$\text{Score}_{p_2,p_4,p_6,p_8,p_{10}} = 5 - \text{Scale Score} \tag{4}$$

Die ausgewertete Punktzahl aller Fragestellungen wird schließlich addiert und mit dem Wert 2,5 multipliziert, sodass eine Gesamtpunktzahl zwischen 0 und 100 erreicht werden kann. Bei der Interpretation der Gesamtpunktzahl ist zu beachten, dass das Programm bereits ab 51 Punkten als „OK“ eingestuft wird (siehe Abbildung 43) (Bangor et al., 2009).

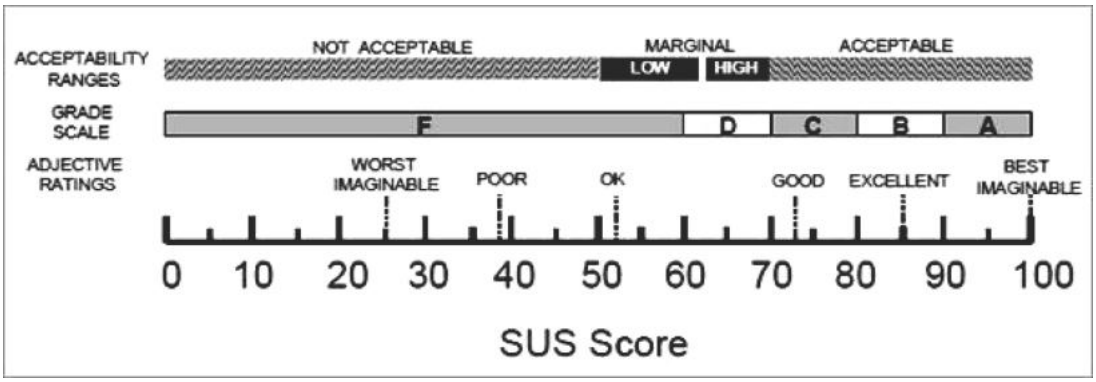


**Abbildung 43:** Einordnung der Gesamtpunktzahl zum SUS Score (Bangor et al., 2009, verändert)

### 5.2.2 Weiteres Feedback

Der Fragebogen wurde zusätzlich durch eigene Fragen erweitert. Um den unterschiedlichen Kenntnisstand der Nutzergruppen im Umgang mit fachbezogener Software zu berücksichtigen, wurden Fragen zu den Vorerfahrungen mit vergleichbarer Software gestellt (siehe Tabelle 5). Dadurch können etwaige Abweichungen in den Ergebnissen zum SUS durch den unterschiedlichen Kenntnisstand der Teilnehmer erklärt werden.

**Tabelle 5:** Fragen zum Erfahrungsstand des Nutzers (eigene Darstellung)

| Nr. | **Fragestellung** (* = Pflichtfeld) | **Antwortmöglichkeiten** |
|---|---|---|
| 1 | Sind Sie bereits mit dem Umgang von GIS- oder CAD-Anwendungen vertraut? (*) | - Ja<br>- Nein |
| 2 | Sind Sie bereits mit der Steuerung von modernen Computerspielen vertraut? (*) | - Ja<br>- Nein |
| 3 | Sind Sie bereits mit Bebauungsplänen und der Darstellung von Planinhalten vertraut? (*) | - Ja<br>- Nein |

Als Testumgebung stand den Probanden sowohl die Entwicklungsumgebung der Unreal Engine als auch das kompilierte Spiel zur Verfügung. Vor der Durchführung der Evaluation war bereits bekannt, dass es im kompilierten Spiel bekannte Fehler bei der Editierung im Siedlungsgebiet gibt. Daher wurde diese Frage bei den Probanden, die das kompilierte Spiel verwendeten, ausgeblendet. Falls dennoch Probleme im Bereich Navigation oder bei der Editierung von Gebäuden und Siedlungen auftraten, hatten die Probanden die Gelegenheit, sich dazu zu äußern (siehe Tabelle 6).

**Tabelle 6:** Fragen zur Testumgebung (eigene Darstellung)

| Nr. | **Fragestellung** (* = Pflichtfeld) | **Antwortmöglichkeiten** |
|---|---|---|
| 1 | In welcher Umgebung wurde die Anwendung getestet? (*) | - Entwicklungsumgebung der Unreal Engine 5<br>- Als kompiliertes Spiel |
| 2 | Konnten Sie sich in der virtuellen Umgebung einwandfrei orientieren und bewegen? | - Ja<br>- Nein |
| 2a | *Falls nein:* Welche Probleme traten bei der Steuerung und Orientierung auf? | (Freitext) |
| 3 | Konnten Sie Änderungen an einzelnen Gebäuden vornehmen? | - Ja<br>- Nein |
| 3a | *Falls nein:* Welche Probleme traten bei der Editierung von einzelnen Gebäuden auf? | (Freitext) |
| *Falls in Frage Nr.1 „Entwicklungsumgebung der Unreal Engine 5“ gewählt wurde:* | | |
| 4 | Konnten Sie Änderungen am Siedlungsgebiet vornehmen? | - Ja<br>- Nein |
| 4a | *Falls nein:* Welche Probleme traten bei der Editierung am Siedlungsgebiet auf? | (Freitext) |

Am Ende der Umfrage durften die Probanden eine Rangliste erstellen um zu bewerten, welche Features in dieser Anwendung besonders gut gelungen sind und welche noch ausbaufähig sind. Zudem konnten sie Ideen zu weiteren Features einbringen oder sonstige Bemerkungen

hinzufügen (siehe Tabelle 7).

**Tabelle 7:** Fragen zu den Features der Anwendung (eigene Darstellung)

| Nr. | Fragestellung | Antwortmöglichkeiten |
|---|---|---|
| 1 | Vergeben Sie eine Platzierung der Features: Platzieren Sie von oben (am besten) nach unten (am schwächsten). Welches Feature hat Ihnen bei dieser Anwendung am besten gefallen? Welches war am schwächsten ausgeprägt? | - Steuerung und Navigation<br>- Benutzeroberfläche<br>- Visualisierung der Gebäude<br>- Editierbarkeit des Gebietes |
| 2 | Welche Features fehlen Ihnen in dieser Anwendung? | (Freitext) |
| 3 | Haben Sie noch weitere Bemerkungen? | (Freitext) |

Die Umfrage wurde über die Plattform Unipark erstellt und über einen Zeitraum von 3 Wochen durchgeführt.

## 5.3 Umfrageergebnisse

Insgesamt nahmen 20 Probanden an den nutzerbasierten Tests und der Beantwortung des Fragebogens teil (siehe Anlage 6). 85% der Teilnehmer waren bereits mit GIS- und CAD-Anwendungen, sowie der Navigation von modernen Computerspielen vertraut. Zudem kannten sich 80% mit der Darstellung von Planinhalten aus. Dadurch bedingt lässt sich erwarten, dass ein Großteil der Teilnehmer über einen ausreichenden Kenntnisstand verfügt, um vergleichbare Anwendungen ohne Einstiegshürden bedienen zu können.

Durch die niedrige Teilnehmerzahl können die Umfrageergebnisse nicht als repräsentativ bezeichnet werden, sie ermöglichen jedoch eine erste Einschätzung zur Gebrauchstauglichkeit der Anwendung. Zudem können bereits fehlerhafte und fehlende Features identifiziert werden, die für die Weiterentwicklung des Prototyps berücksichtigt werden können.

### 5.3.1 Feedback und Bewertung

In der Umfrage gab mehr als die Hälfte der Teilnehmer an, dass die Steuerung und Navigation der Anwendung am besten gelungen wäre. Die Benutzeroberfläche und die Visualisierung der Gebäude erhielt eher durchwachsene Bewertungen. Etwa drei von vier Teilnehmern empfanden die Editierbarkeit des Gebietes als das am schwächsten ausgeprägte Feature.

80% der Probanden konnten sich einwandfrei in der virtuellen Umgebung orientieren und bewegen. In den restlichen Fällen traten Programmfehler auf, bei denen der Avatar durch den

Boden fiel und die Steuerung der Pfeiltasten temporär ausfiel. Zudem waren einige Gebäude zu tief platziert, wodurch diese nicht betretbar waren.

Bei der Editierung traten vermehrt Probleme bei den Teilnehmern auf. 70% gaben an, dass die Editierung einzelner Gebäude einwandfrei funktioniert habe. Hingegen konnten nur 38% der Teilnehmer Änderungen am gesamten Siedlungsgebiet vornehmen, wobei dieses Feature nicht von allen getestet wurde. Dabei kam es vor, dass die getätigten Eingaben nicht erkannt wurden. Außerdem fehlte bei der grafischen Selektion ein visuelles Feedback. Dies führte dazu, dass unklar war, welche Objekte selektiert wurden und ob die vorgenommenen Änderungen korrekt verarbeitet wurden. Auch die Veränderung der Gebäudegrundfläche funktionierte fehlerhaft bis gar nicht.

Trotz der genannten Kritikpunkte wurde die Anwendung im Durchschnitt mit einem SUS-Score von 67,75 Punkten bewertet. Nach Bangor et al. lässt sich dieser Punktestand zwischen „OK“ und „Good“ einordnen (Bangor et al., 2009). Diese Ergebnis lässt sich so interpretieren, dass die Anwendung innerhalb der Teilnehmergruppe grundsätzlich die Anforderungen einer gebrauchstauglichen Software erfüllt. Jedoch besteht hinsichtlich der Editierfunktionen der Software Verbesserungsbedarf.

### 5.3.2 Verbesserungen und neue Features

Die Probanden hatten die Möglichkeit ihre Ideen für die Weiterentwicklung der Anwendung einzubringen. Dabei wurden folgende Features vorgeschlagen, die verbessert oder zukünftig in die Anwendung integriert werden können (siehe Anlage 7):

(1) Steuerung und Navigation: Die Kamerasteuerung sollte sich stärker an modernen Computerspielen orientieren. Dabei könnte die Kamera ohne gedrückte Maustaste gesteuert werden. Die Third-Person-Kamera sollte dabei den Spieler in Laufrichtung verfolgen und sich automatisch mitbewegen. Für den Perspektivwechsel der Kamera könnte zudem eine Schaltfläche zur Verfügung stehen. In der Top-Down-Perspektive vom Editormodus sollte die Postion und Blickrichtung des Avatars visualisiert werden. Der Avatar sollte sich unabhängig von der Blickrichtung in einer angepassten Top-Down-Steuerung bewegen können. Um längere Laufwege zu ersparen, könnte die Position des Avatars per Mausklick verändert werden. Bei der grafischen Selektion könnte zudem das Auswahlrechteck durch die Drag-Funktion der Maustaste erstellt werden.

(2) Editierfunktionen: Der Funktionsumfang des Editormodus sollte zukünftig erweitert werden. Unter anderem könnten Funktionen zum Messen, zur Bildschirmaufnahme und zum Speichern, sowie weiterführende Analysen und Simulationen in die Anwendung integriert werden. Auch sollten getätigte Eingaben rückgängig gemacht werden können. Durch Shortcuts könnten wiederkehrende Eingaben beschleunigt werden. Zudem könnte das Gebiet durch zusätzliche Gebäude- und Dachtypen abwechlungsreicher gestaltet werden. Bei der Platzierung der Gebäude sollten auch Rotation, Größe und Form eingestellt werden können und zugleich die Überbaubarkeit des Gebiets berücksichtigt werden. Die Baugrenzen und -linien sollten sich dabei editieren und verschieben lassen. Darüber hinaus könnten bei der Siedlungsgenerierung weitere Vorgaben, wie etwa Grenzabstände, berücksichtigt werden. Zudem könnte der Prozessfortschritt über einen Ladebalken visualisiert werden.

(3) Benutzeroberfläche: Die Attributtabelle sollte noch intuitiver gestaltet werden. Der dargestellte Text der Benutzeroberfläche sollte in einer einheitlichen Sprache verfasst sein. Für die Schaltflächen könnte eine Hoverfunktion implementiert werden, bei der weiterführende Informationen eingeblendet werden. Zudem sollten die Parameter durch entsprechende Maßeinheiten ergänzt werden. Zusätzlich könnte mehr visuelles Feedback in die Anwendung integriert werden. Dabei sollten sowohl grafisch selektierte Objekte als auch neu platzierte Objekte und vorgenommene Änderungen visuell hervorgehoben werden. Zudem sollten die erlaubten Parameter für die Gebäude und das Siedlungsgebiet eingeblendet werden können.

(4) Datenintegration: Der zugrundeliegende Bebauungsplan mit seinen Festsetzungen sollte stärker in die Anwendung integriert werden. Die Planunterlagen könnten über eine Schaltfläche verlinkt werden, zusätzlich ließe sich für die Planinhalte eine Legende abbilden. Die erlaubten Parameter für die Gebäude und das Siedlungsgebiet könnten dabei visuell hervorgehoben werden. Durch die Integration von Layern, ähnlich wie bei einem GIS, ließen sich verschiedene Kartengrundlagen einfügen, zwischen diesen gewechselt werden kann.

## 5.4 Beurteilung der Ergebnisse

Die festgelegten Ziele dieser Bachelorarbeit werden durch den Prototyp grundsätzlich erfüllt. Zum einen werden die prozeduralen Funktionen der Unreal Engine erfolgreich genutzt, um ein 3D-Modell eines Bebauungsplans zu erstellen. Zum anderen liegt der Prototyp als kompiliertes Spiel vor, das hinreichend getestet wurde.

Um die Evaluation abzuschließen, werden sowohl die in der Literatur vorgestellten Konzepte und Anwendungsbeispiele als auch die daraus resultierenden Anforderungen mit dem Projektergebnis verglichen.

### 5.4.1 Entwicklungsstand

Bei dem Projektergebnis handelt es sich um einen Prototyp, der die Anforderungen der Stakeholder und vergleichbarer Anwendungen nur in Ansätzen erfüllen soll. Nach der Definition von Kritzinger et al. handelt es sich dabei um keinen digitalen Zwilling, sondern vielmehr um ein digitales Modell (Kritzinger et al., 2018). Im Projekt wurden keine dynamischen Daten eingebunden, zudem steht das virtuelle Modell nicht in einem durchgehenden Datenaustausch mit der realen Welt.

Es handelt sich dabei eher um ein geometrisches 3D-Stadtmodell. Durch die Datenintegration mit Cesium fließen das DGM, das DOP und die 3D-Gebäudemodelle als essentielle Datengrundlagen eines 3D-Stadtmodells zusammen. Hierbei werden jedoch die 3D-Gebäudemodelle von CityGML in 3D-Tiles konvertiert und liegen in der Unreal Engine daher nicht mehr semantisch vor. Zum aktuellen Stand fehlt noch ein Großteil der urbanen Objekte, da ausschließlich Gebäude repräsentiert werden. Zwar werden Straßen als Ausschlussflächen für die Siedlungsgenerierung modelliert, jedoch nicht grafisch darstellt.

Derzeit stellt der Prototyp in erster Linie eine Testumgebung für die Integration von Geodaten und die prozedurale Modellierung urbaner Gebiete innerhalb einer Game Engine dar. Zukünftig kann der Prototyp durch die Integration zusätzlicher urbaner Daten oder prozedural modellierter Objekte vervollständigt werden. Auch die Einbindung von IoT in die Unreal Engine ist möglich (Rantanen et al., 2023). Langfristig lässt sich ein urbaner digitaler Zwilling mittels Game Engines realisieren, jedoch stellt die Integration semantischer Daten zurzeit noch eine Herausforderung dar.

### 5.4.2 Geodatenintegration

Die Integration von Geodaten mit Cesium hat grundsätzlich funktioniert. Die amtlichen Daten ließen sich mit nur wenig Aufwand bei der Aufbereitung in die Unreal Engine einbinden. Besonders gut gelungen ist dabei der Zugriff auf extern gelagerte Daten. Die integrierten Geodaten lagen nicht lokal vor, sondern wurden über Cesium in die Engine geladen. Auch können eigene

Geodaten selbst gehostet und in die Engine integriert werden (Cesium, o.J. c). Jedoch kam es zu einigen Programmfehlern, da die Daten während der Laufzeit oft nachgeladen wurden. Dadurch kam es vor, dass teilweise kein vollständiges Siedlungsgebiet prozedural erzeugt werden konnte, da die Bodenflächen aus Cesium außerhalb des Sichtfelds des Avatars lagen.

Zudem gestaltet sich die Geodatenintegration von herkömmlichen GIS-Datenformaten als herausfordernd. Für die prozedurale Modellierung des Baugebiets mussten die Planinhalte händisch digitalisiert werden. Wesentlich sinnvoller wäre eine Integration von Bauleitplänen im Datenstandard XPlanung (Zahid et al., 2024). Außerdem wäre es hilfreich wenn sich etablierte Geodaten, beispielsweise Shape-Dateien, in die Unreal Engine importieren ließen. Auf Grundlage von Splines, die als Polygonflächen formatiert sind, kann die Generierung von Siedlungsgebieten ad hoc vorgenommen werden. Die CityEngine ermöglicht beispielsweise die Anwendung von Produktionsregeln auf zweidimensionale Grundrisse für die prozedurale Modellierung von Siedlungen (Badwi et al., 2022).

Insgesamt funktioniert die Geodatenintegration innerhalb der Unreal Engine nur teilweise. Es lässt sich nur eine begrenzte Auswahl von Geodatenformaten unter Nutzung des Plugins Cesium integrieren.

### 5.4.3 Prozedurale Funktionen

Das Ergebnis zeigt, dass das PCG-Framework der Unreal Engine durchaus Potential für die Modellierung komplexer urbaner Inhalte hat. Durch die Verschachtelung einzelner PCG-Prozesse ist es möglich, prozedural erstellbare Actors in andere PCG-Prozesse zu integrieren. Zudem können PCG-Graphen als Komponente in Actor-Blueprints eingebunden werden und sowohl auf Attribute als auch auf weitere Komponenten des Actors zugreifen.

Die modellierten Gebäude sind sehr detailliert gestaltet. Neben ausmodellierten Wand- und Bodenflächen, sowie unterschiedlichen Dachformen wurden auch Innenräume, Treppen, Beleuchtung und Türen erstellt. Dabei wird durch Structs und Data Assets ein modularer Austausch aus einer Gruppe von Static Mesh-Objekten ermöglicht.

Das generierte Siedlungsgebiet ist noch unvollständig, zeigt aber anschaulich, auf welche Weise die Festsetzungen von Bebauungsplänen als Parameter für die prozedurale Modellierung grundsätzlich einfließen können. Hierbei werden die gesetzlichen Regelungen zur Überbaubarkeit von Grundstücken als Einschränkungsregeln verwendet. Durch die Maße der baulichen

Nutzung und der Bauweise wird die Bebauungsdichte des Gebietes beeinflusst. Hierdurch wird näherungsweise vermittelt, wie die Planinhalte in die Realität umgesetzt werden könnten. Für die Öffentlichkeit reicht die anschauliche Visualisierung eines Bebauungsplans bereits aus, um sich die Realisierung urbaner Planungsszenarien besser vorstellen zu können. Davon profitieren auch Stadtplaner, um die Entscheidungsprozesse nachvollziehbar vermitteln zu können.

Jedoch zeigt das prozedural generierte Siedlungsgebiet einige Grenzen auf. Zum einen wurde sich nur auf eine Teilmenge der Planinhalte und gesetzlichen Festsetzungen beschränkt. Zudem werden die Arten der baulichen Nutzung nicht voneinander differenziert. Außerdem werden als Gebäude ausschließlich Einfamilienhäuser erstellt, wobei es an der Variation von Gebäuden im generierten Gebiet mangelt. Auch wurden textliche Festsetzungen des Bebauungsplans und gesetzliche Regelungen einzelner Bundesländer bei der Modellierung nicht berücksichtigt. Aufgrund größerer Abweichungen zwischen den Festsetzungen verschiedener Bebauungspläne ist es schwierig, einen einheitlichen Algorithmus zur Siedlungsgenerierung auf Grundlage dieser Festsetzungen zu treffen. Dadurch bedingt kann das Projektergebnis nicht den Anforderungen von Stadtplanern und den Trägern öffentlicher Belange vollständig gerecht werden.

Die prozedural generierten Inhalte lassen zurzeit sich weder importieren noch exportieren. Zudem erfolgt die prozedurale Modellierung als node-basierter Workflow im Vergleich zu Shape-basierten Grammatiken auf einem eher umständlichen Wege. Statt einer mengenbezogenen Modellierung von Gebäuden innerhalb des Siedlungsgebietes wird jedes Gebäude individuell mit seinen Bestandteilen berechnet. Dadurch bedingt kostet die Generierung des Siedlungsgebiets viel an Performance. Zudem setzt die Erstellung von PCG-Graphen programmbezogenes Vorwissen beim Nutzer voraus.

Die prozeduralen Funktionen der Unreal Engine sind insgesamt sehr umfangreich gestaltet. Für eine performante, variantenreiche und gesetzeskonforme Modellierung größerer Siedlungsgebiete, fehlt es zum aktuellen Zeitpunkt an Feinschliff.

# 6 Fazit und Ausblick

Die Forschungsfrage dieser Arbeit untersucht, inwiefern Game Engines für die prozedurale Generierung urbaner Planungsszenarien eingesetzt werden können.

Die Beurteilung des Projektergebnisses zeigt, dass Game Engines großes Potential in der pro-

zeduralen Modellierung besitzen. Bereits nach 13 Wochen Entwicklungszeit war es möglich, einen 3D-Bebauungsplan größtenteils nachzumodellieren. Darüber hinaus wird die Unreal Engine regelmäßig durch neue Features erweitert. In der aktuellen Version 5.5 ist erstmals möglich, Grammatiken für die prozedurale Modellierung einzusetzen (siehe Abbildung 44). Darüber hinaus stellt die Unreal Engine seit der Version 5.5, Funktionen zur Erstellung von Höhenlinien und einen A*-Algorithmus zum Pathfinding zur Verfügung (Epic Games, o.J. i).

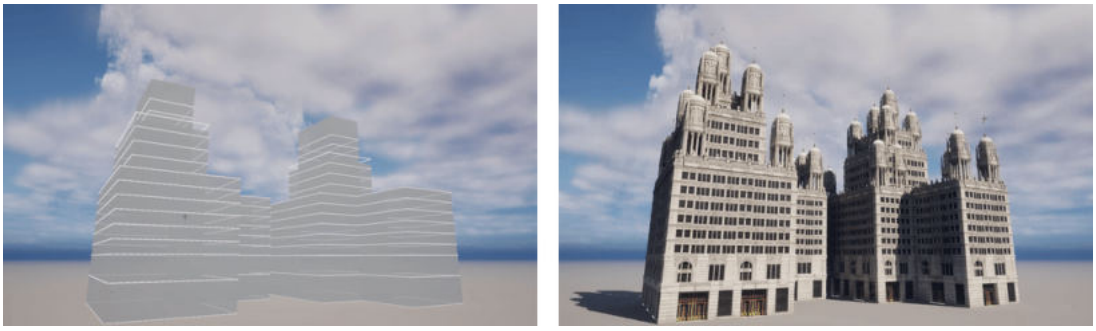

**Abbildung 44:** Grammars in der Unreal Engine 5.5: als Debug-Darstellung (links) und texturiert (rechts) (Epic Games, o.J. i, verändert)

Hingegen steht die Modellierung urbaner Planungsszenarien in einer Game Engine vor vielen Herausforderungen. Zum einen gestaltet sich die Überführung zweidimensionaler Bauleitpläne unter Berücksichtigung gesetzlicher Vorgaben als aufwändig. Zum anderen mangelt es den Game Engines zurzeit an einer umfangreichen Integration von Geodaten. Dadurch bedingt lassen sich Game Engines zum aktuellen Zeitpunkt nur eingeschränkt für die prozedurale Generierung urbaner Planungsszenarien einsetzen.

Zukünftige Arbeiten können dabei auf die Stärken von Game Engines weiter aufbauen. Aus den Umfrageergebnissen gehen bereits konkrete Verbesserungen und Features hervor, mit denen der Prototyp weiterentwickelt werden kann. Auch wenn kein urbaner digitaler Zwilling als Endprodukt entstehen sollte, erfüllt die Anwendung den Zweck, Planungsvorhaben für die Öffentlichkeit anschaulich zu visualisieren. Durch die hohe Grafikleistung von Game Engines, lassen sich komplexe Stadtgebiete wie aus „Marvel's Spider-Man“ darstellen (Santiago, 2019). Auch städtebauliche Entwürfe, wie im Baugebiet Fliegerhorst der Stadt Oldenburg, zeigen durch Visualisierungen das Zusammenleben der dort wohnenden Menschen im zukünftigen Baugebiet (siehe Abbildung 45). Darstellungen wie diese können durch die prozeduralen Funktionen einer Game Engine umgesetzt werden. Zudem können diese Umgebungen aus verschiedenen Perspektiven betrachtet und mit den dargestellten Objekten interagiert werden.


**Abbildung 45:** Visualisierung zum Bauabschnit N-777 E der Stadt Oldenburg (BOMA Architekten, o.J.)

Andererseits können sich weitere Arbeiten auf die Schwächen von Game Engines konzentrieren, mit dem Ziel, einen urbanen digitalen Zwilling zu entwickeln. Die vorgestellten Anwendungsbeispiele zeigen, dass 3D-Stadtmodelle zukünftig durch urbane digitale Zwillinge abgelöst werden können. Gleichzeitig wird auch an internationalen Standards gearbeitet. Im vergangenen Jahr wurde bereits ein Diskussionspapier vom OGC für urbane digitale Zwillinge veröffentlicht (OGC, 2024). Dabei können zukünftige Arbeiten sich mit der Umsetzung solcher Standards auseinandersetzen und semantische Datenmodelle, sowie dynamische Daten in Game Engines implementieren.

Insgesamt stellen Game Engines sinnvolle Open-Source-Alternativen dar, die interdisziplinär für die Erstellung urbaner 3D-Modelle eingesetzt werden können und somit zu den Entscheidungsprozessen der urbanen Planung beitragen.

# Literaturverzeichnis


Al-kazzaz, D. A., & Bridges, A. H. (2012). A framework for adaptation in shape grammars. *Design Studies*, *33*(4), 342–356. https://doi.org/10.1016/j.destud.2011.11.001

Amato, A. (2017). Procedural Content Generation in the Game Industry. In O. Korn & N. Lee (Hrsg.), *Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation* (S. 15–25). Springer International Publishing. https://doi.org/10.1007/978-3-319-53088-8_2

Badwi, I. M., Ellaithy, H. M., & Youssef, H. E. (2022). 3D-GIS Parametric Modelling for Virtual Urban Simulation Using CityEngine. *Annals of GIS*, *28*(3), 325–341. https://doi.org/10.1080/19475683.2022.2037019

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: adding an adjective rating scale. *J. Usability Studies*, *4*(3), 114–123.

Batty, M. (2018). Digital twins. *Environment and Planning B: Urban Analytics and City Science*, *45*(5), 817–820. https://doi.org/10.1177/2399808318796416

BauGB. (2023). *Baugesetzbuch (BauGB)*. Verfügbar 2. Januar 2025 unter https://www.gesetze-im-internet.de/bbaug/BauGB.pdf

Baumgartner, P. (1999). Evaluation mediengestützten Lernens: Theorie – Logik – Modelle. In M. Kindt (Hrsg.), *Projektevaluation in der Lehre - Multimedia an Hochschulen zeigt Profil(e)* (S. 61–97, Bd. 7). Waxmann.

BauNVO. (2023). *Verordnung über die bauliche Nutzung der Grundstücke (Baunutzungsverordnung - BauNVO)*. Verfügbar 2. Januar 2025 unter https://www.gesetze-im-internet.de/baunvo/BauNVO.pdf

Becker, T., Nagel, C., & Kolbe, T. H. (2011). Integrated 3D Modeling of Multi-utility Networks and Their Interdependencies for Critical Infrastructure Analysis. In T. H. Kolbe, G. König & C. Nagel (Hrsg.), *Advances in 3D Geo-Information Sciences* (S. 1–20). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-12670-3_1

Benner, J., Geiger, A., & Häfele, K. H. (2010). Concept for building licensing based on standardized 3D geo information [35.01.01; LK 01]. *5th 3D GeoInfo Conf., Berlin, November 3-4, 2010 International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XXXVIII-4/W15 (2010)*, 9–12.

Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, *59*, 25–37. https://doi.org/10.1016/j.compenvurbsys.2016.04.005

Billen, R., Cutting-Decelle, A.-F., Marina, O., de Almeida, J.-P., Caglioni, M., Falquet, G., Leduc, T., Métral, C., Moreau, G., Perret, J., Rabin, G., San Jose, R., Yatskiv, I., & Zlatanova, S. (2014). 3D City Models and urban information: Current issues and perspectives. *Usage, Usability, and Utility of 3D City Models*, I–118. https://doi.org/10.1051/TU0801/201400001

Blatz, M., & Korn, O. (2017). A Very Short History of Dynamic and Procedural Content Generation. In O. Korn & N. Lee (Hrsg.), *Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation* (S. 1–13). Springer International Publishing. https://doi.org/10.1007/978-3-319-53088-8_1

BOMA Architekten. (o.J.). *Entwurf eines Wohnkomplexes auf einem ehemaligen Fliegerhorst*. Verfügbar 2. Januar 2025 unter https://www.boma-architekten.de/referenz/entwurf-eines-wohnkomplexes-auf-einem-ehemaligen-fliegerhorst

Borchard, K. (2018). Städtebau. In ARL - Akademie für Raumforschung und Landesplanung (Hrsg.), *Handwörterbuch der Stadt- und Raumentwicklung* (S. 2381–2389).

Boschert, S., & Rosen, R. (2016). Digital Twin—The Simulation Aspect. In P. Hehenberger & D. Bradley (Hrsg.), *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers* (S. 59–74). Springer International Publishing. https://doi.org/10.1007/978-3-319-32156-1_5

Brooke, J. (1996). SUS - A Quick and Dirty Usability Scale. In P. W. Jordan, B. Thomas, I. L. McClelland & B. Weerdmeester (Hrsg.), *Usability Evaluation In Industry* (S. 189–194). CRC Press. https://doi.org/10.1201/9781498710411-35

Caprari, G., Castelli, G., Montuori, M., Camardelli, M., & Malvezzi, R. (2022). Digital Twin for Urban Planning in the Green Deal Era: A State of the Art and Future Perspectives. *Sustainability*, *14*(10). https://doi.org/10.3390/su14106263

Cesium. (o.J. a). *3D Tiles*. Verfügbar 5. Januar 2025 unter https://cesium.com/why-cesium/3d-tiles/

Cesium. (o.J. b). *Bring the real world to Unreal Engine*. Verfügbar 2. Januar 2025 unter https://cesium.com/platform/cesium-for-unreal/

Cesium. (o.J. c). *The Cesium Platform*. Verfügbar 5. Januar 2025 unter https://cesium.com/platform/

Cesium. (o.J. d). *Create and host 3D content in the cloud*. Verfügbar 2. Januar 2025 unter https://cesium.com/platform/cesium-ion/

Connected Urban Twins. (o.J.). *Das Projekt*. Verfügbar 2. Januar 2025 unter https://www.connectedurbantwins.de/das-projekt-2/

Demir, C., & Koramaz, T. K. (2018). GIS-based Procedural Modeling in Contemporary Urban Planning Practice. *2018 22nd International Conference Information Visualisation (IV)*, 553–560. https://doi.org/10.1109/iV.2018.00102

DIN. (2018). DIN EN ISO 9241-11:2018-11 Ergonomie der Mensch-System-Interaktion - Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte (ISO 9241-11:2018); Deutsche Fassung EN ISO 9241-11:2018. https://doi.org/10.31030/2757945

DIN. (2024). DIN SPEC 91607:2024-11 Digitale Zwillinge für Städte und Kommunen. https://doi.org/10.31030/3575521

Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., & Worley, S. (2002). *Texturing and Modeling: A Procedural Approach* (3rd). Morgan Kaufmann Publishers Inc.

Epic Games. (2023). *Comparing Blueprints and C++ Use Cases*. Verfügbar 2. Januar 2025 unter https://dev.epicgames.com/community/learning/tutorials/qM2K/unreal-engine-comparing-blueprints-and-c-use-cases

Epic Games. (o.J. a). *Hardware and Software Specifications*. Verfügbar 2. Januar 2025 unter https://dev.epicgames.com/documentation/en-us/unreal-engine/hardware-and-software-specifications-for-unreal-engine

Epic Games. (o.J. b). *Lumen Global Illumination and Reflections*. Verfügbar 3. Januar 2025 unter https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine

Epic Games. (o.J. c). *The most powerful real-time 3D creation tool - Unreal Engine*. Verfügbar 5. Januar 2025 unter https://www.unrealengine.com/

Epic Games. (o.J. d). *Nanite Virtualized Geometry*. Verfügbar 2. Januar 2025 unter https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine

Epic Games. (o.J. e). *Procedural Content Generation Overview*. Verfügbar 2. Januar 2025 unter https://dev.epicgames.com/documentation/en-us/unreal-engine/procedural-content-generation-overview

Epic Games. (o.J. f). *Quixel*. Verfügbar 4. Januar 2025 unter https://www.fab.com/sellers/Quixel

Epic Games. (o.J. g). *Quixel Megascans*. Verfügbar 2. Januar 2025 unter https://quixel.com/megascans/

Epic Games. (o.J. h). *Third Person Template*. Verfügbar 2. Januar 2025 unter https://dev.epicgames.com/documentation/en-us/unreal-engine/third-person-template-in-unreal-engine

Epic Games. (o.J. i). *Unreal Engine Public Roadmap*. Verfügbar 2. Januar 2025 unter https://portal.productboard.com/epicgames/1-unreal-engine-public-roadmap/tabs/109-unreal-engine-5-5

Epic Games. (o.J. j). *Unreal Engine Terminology*. Verfügbar 2. Januar 2025 unter https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-terminology

ESRI. (o.J. a). *Analysis tools*. Verfügbar 5. Januar 2025 unter https://doc.arcgis.com/en/urban/latest/help/help-measure.htm

ESRI. (o.J. b). *Plans*. Verfügbar 2. Januar 2025 unter https://doc.arcgis.com/en/urban/latest/get-started/get-started-plans.htm

ESRI. (o.J. c). *Tutorial 13: Facade Wizard*. Verfügbar 5. Januar 2025 unter https://doc.arcgis.com/en/cityengine/latest/tutorials/tutorial-13-facade-wizard.htm

ESRI. (o.J. d). *Vitruvio - Plugin for Unreal Engine*. Verfügbar 2. Januar 2025 unter https://esri.github.io/cityengine/vitruvio

ESRI. (o.J. e). *What is ArcGIS Urban*. Verfügbar 2. Januar 2025 unter https://doc.arcgis.com/en/urban/latest/get-started/get-started-what-is-urban.htm

Ferré-Bigorra, J., Casals, M., & Gangolells, M. (2022). The adoption of urban digital twins. *Cities*, *131*, 103905. https://doi.org/10.1016/j.cities.2022.103905

Frade, M., de Vega, F. F., & Cotta, C. (2012). Automatic evolution of programs for procedural generation of terrains for video games. *Soft Computing*, *16*(11), 1893–1914. https://doi.org/10.1007/s00500-012-0863-z

Freiknecht, J., & Effelsberg, W. (2017). A Survey on the Procedural Generation of Virtual Worlds. *Multimodal Technologies and Interaction*, *1*(4). https://doi.org/10.3390/mti1040027

Gao, M., Kortum, P., & Oswald, F. L. (2020). Multi-Language Toolkit for the System Usability Scale. *International Journal of Human–Computer Interaction*, *36*(20), 1883–1901. https://doi.org/10.1080/10447318.2020.1801173

GG. (2024). *Grundgesetz für die Bundesrepublik Deutschland*. Verfügbar 2. Januar 2025 unter https://www.gesetze-im-internet.de/gg/GG.pdf

Glaessgen, E., & Stargel, D. (2012). The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. https://doi.org/10.2514/6.2012-1818

Grieves, M. (2022). Intelligent digital twins and the development and management of complex systems. *Digital Twin*, *2*(8). https://doi.org/10.12688/digitaltwin.17574.1

Gröger, G., & Plümer, L. (2012). CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, *71*, 12–33. https://doi.org/10.1016/j.isprsjprs.2012.04.004

Gu, N., & Maher, M. L. (2014). 3 Generative Design Grammars. In *Designing Adaptive Virtual Worlds* (S. 20–38). De Gruyter Open Poland. https://doi.org/10.2478/9783110367669.3

Gutiérrez, M. M., & Rojano-Cáceres, J. R. (2020). Interpretation of the SUS questionnaire in Mexican sign language to evaluate usability an approach. *2020 3rd International Conference of Inclusive Technology and Education (CONTIE)*, 180–183. https://doi.org/10.1109/CONTIE51334.2020.00040

Hendrikx, M., Meijer, S., Van Der Velden, J., & Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, *9*(1). https://doi.org/10.1145/2422956.2422957

Herczeg, M. (2018). *Theorien, Modelle und Kriterien für gebrauchstaugliche interaktive Computersysteme*. De Gruyter Oldenbourg. https://doi.org/10.1515/9783110446869

Hofmann, D., & Heller, A. (2013). Prozedurale 3D-Stadtmodellierung mit der ESRI CityEngine. In J. Strobl, T. Blaschke, G. Griesebner & B. Zagel (Hrsg.), *Angewandte Geoinformatik 2013: Beiträge zum 25. AGIT-Symposium Salzburg* (S. 90–99). Wichmann Verlag.

Hu, H., Feng, B., Xu, B., Zhu, Q., Ge, X., & Chen, M. (2022). Efficient Procedural Modelling of Building Façades Based on Windows from Sketches. *The Photogrammetric Record*, *37*(179), 333–353. https://doi.org/10.1111/phor.12425

Ignatius, M., Wong, N. H., Martin, M., & Chen, S. (2019). Virtual Singapore integration with energy simulation and canopy modelling for climate assessment. *IOP Conference Series: Earth and Environmental Science*, *294*(1), 012018. https://doi.org/10.1088/1755-1315/294/1/012018

Jahnke, M., Berger, T., Donaubauer, A., & Krisp, J. (2011). Nicht fotorealistische Darstellung von 3D-Stadtmodellen. *HMD Praxis der Wirtschaftsinformatik*, *48*(3), 101–112. https://doi.org/10.1007/BF03340592

Jaquemotte, I. (2014). Die Stadt in 3D — Modellierung und Präsentation. *KN - Journal of Cartography and Geographic Information*, *64*(2), 3–9. https://doi.org/10.1007/BF03544089

Jeddoub, I., Nys, G.-A., Hajji, R., & Billen, R. (2023). Digital Twins for cities: Analyzing the gap between concepts and current implementations with a specific focus on data integration. *International Journal of Applied Earth Observation and Geoinformation*, *122*, 103440. https://doi.org/10.1016/j.jag.2023.103440

Jesus, D., Coelho, A., Rebelo, C., & Cardoso, A. (2012). Modeling Urban Environments from Geospatial Data: A Pipeline for Procedural Modeling. *Proceedings of the The Third Workshop on Procedural Content Generation in Games*, 1–8. https://doi.org/10.1145/2538528.2538533

Kelly, G., & McCabe, H. (2006). A Survey of Procedural Techniques for City Generation. *The ITB Journal*, *7*(2), 5. https://doi.org/10.21427/D76M9P

Kim, J.-S., Kavak, H., & Crooks, A. (2018). Procedural city generation beyond game development. *SIGSPATIAL Special*, *10*(2), 34–41. https://doi.org/10.1145/3292390.3292397

Kolbe, T. H., & Donaubauer, A. (2021). Semantic 3D City Modeling and BIM. In W. Shi, M. F. Goodchild, M. Batty, M.-P. Kwan & A. Zhang (Hrsg.), *Urban Informatics* (S. 609–636). Springer Singapore. https://doi.org/10.1007/978-981-15-8983-6_34

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification [16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018]. *IFAC-PapersOnLine*, *51*(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474

Kutzner, T., Chaturvedi, K., & Kolbe, T. H. (2020). CityGML 3.0: New Functions Open Up New Applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, *88*(1), 43–61. https://doi.org/10.1007/s41064-020-00095-z

Lancelle, M., & Fellner, D. W. (2004). Current issues on 3D city models [Poster]. *Image and Vision Computing New Zealand*. https://marcel-lancelle.de/research/publications/

LGLN. (o.J.). *OpenGeoData Niedersachsen*. Verfügbar 2. Januar 2025 unter https://ni-lgln-opengeodata.hub.arcgis.com/

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, *18*(3), 280–299. https://doi.org/10.1016/0022-5193(68)90079-9

Mandelbrot, B. B. (1982). *The fractal geometry of nature*. Freeman.

Matei, A., & Cocoșatu, M. (2024). Artificial Internet of Things, Sensor-Based Digital Twin Urban Computing Vision Algorithms, and Blockchain Cloud Networks in Sustainable Smart City Administration. *Sustainability*, *16*(16). https://doi.org/10.3390/su16166749

Mazzetto, S. (2024). A Review of Urban Digital Twins Integration, Challenges, and Future Directions in Smart City Development. *Sustainability*, *16*(19). https://doi.org/10.3390/su16198337

Merrell, P. (2023). Example-Based Procedural Modeling Using Graph Grammars. *ACM Trans. Graph.*, *42*(4). https://doi.org/10.1145/3592119

Mohan, K. K., Verma, A., & Srividya, A. (2010). Software reliability estimation through black box and white box testing at prototype level. *2010 2nd International Conference on Reliability, Safety and Hazard - Risk-Based Technologies and Physics-of-Failure Methods (ICRESH)*, 517–522. https://doi.org/10.1109/ICRESH.2010.5779604

Mojang. (2023). *Minecraft Controls*. Verfügbar 2. Januar 2025 unter https://www.minecraft.net/de-de/article/minecraft-controls

Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). Procedural modeling of buildings. *ACM Trans. Graph.*, *25*(3), 614–623. https://doi.org/10.1145/1141911.1141931

Müller, P., Zeng, G., Wonka, P., & Van Gool, L. (2007). Image-based procedural modeling of facades. *ACM Trans. Graph.*, *26*(3), 85–es. https://doi.org/10.1145/1276377.1276484

OGC. (2024). *Urban Digital Twins: Integrating Infrastructure, natural environment and people*. Verfügbar 2. Januar 2025 unter http://www.opengis.net/doc/dp/UDT

Omrany, H., Ghaffarianhoseini, A., Ghaffarianhoseini, A., & Clements-Croome, D. J. (2023). The uptake of City Information Modelling (CIM): a comprehensive review of current

implementations, challenges and future outlook. *Smart and Sustainable Built Environment*, *12*(5), 1090–1116. https://doi.org/10.1108/SASBE-06-2022-0116

Pahl-Weber, E., & Schwartze, F. (2018). Stadtplanung. In ARL - Akademie für Raumforschung und Landesplanung (Hrsg.), *Handwörterbuch der Stadt- und Raumentwicklung* (S. 2509–2520).

Parish, Y. I. H., & Müller, P. (2001). Procedural modeling of cities. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 301–308. https://doi.org/10.1145/383259.383292

Partsch, H. (2010). *Requirements-Engineering systematisch: Modellbildung für softwaregestützte Systeme*. https://doi.org/10.1007/978-3-642-05358-0

Patow, G. (2012). User-Friendly Graph Editing for Procedural Modeling of Buildings. *IEEE Computer Graphics and Applications*, *32*(2), 66–75. https://doi.org/10.1109/MCG.2010.104

PlanZV. (2021). *Verordnung über die Ausarbeitung der Bauleitpläne und die Darstellung des Planinhalts (Planzeichenverordnung - PlanZV)*. Verfügbar 2. Januar 2025 unter https://www.gesetze-im-internet.de/planzv_90/PlanZV.pdf

Prusinkiewicz, P., & Lindenmayer, A. (1990). *The algorithmic beauty of plants*. Springer-Verlag. https://doi.org/10.1007/978-1-4613-8476-2

Rantanen, T., Julin, A., Virtanen, J.-P., Hyyppä, H., & Vaaja, M. T. (2023). Open Geospatial Data Integration in Game Engine for Urban Digital Twin Applications. *ISPRS International Journal of Geo-Information*, *12*(8). https://doi.org/10.3390/ijgi12080310

Ranzinger, M., & Gleixner, G. (1997). GIS datasets for 3D urban planning [Urban Data Management Symposium]. *Computers, Environment and Urban Systems*, *21*(2), 159–173. https://doi.org/10.1016/S0198-9715(97)10005-9

Rehkopf, M. (o.J.). *Agile Epics: Definition, Beispiele und Vorlagen*. Verfügbar 2. Januar 2025 unter https://www.atlassian.com/de/agile/project-management/epics

Reicher, C. (2024). Herausforderungen für eine nachhaltige Stadtentwicklung und einen resilienten Städtebau. In S. Beier, P. Hense, C. Klümper, S. Lechtenböhmer & C. Reicher (Hrsg.), *Die UN-Nachhaltigkeitsziele als interdisziplinäre Herausforderung: Aufgaben, Aspekte und Ansätze* (S. 37–46). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-44103-6_5

Ritschel, T., Dachsbacher, C., Grosch, T., & Kautz, J. (2012). The State of the Art in Interactive Global Illumination. *Computer Graphics Forum*, *31*(1), 160–188. https://doi.org/10.1111/j.1467-8659.2012.02093.x

ROG. (2023). *Raumordnungsgesetz (ROG)*. Verfügbar 2. Januar 2025 unter https://www.gesetze-im-internet.de/rog_2008/ROG.pdf

Rummel, B. (2013). *System Usability Scale (Translated into German)*. Verfügbar 2. Januar 2025 unter https://www.researchgate.net/publication/272830038_System_Usability_Scale_Translated_into_German

Sabri, S., Pettit, C. J., Kalantari, M., Rajabifard, A., White, M., Lade, O., & Ngo, T. D. (2015). What are Essential requirements in Planning for Future Cities using Open Data Infrastructures and 3D Data Models. https://api.semanticscholar.org/CorpusID:9992384

Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). *BIM Handbook : A Guide to Building Information Modeling for Owners, Managers, Architects, Engineers, Contractors, and Fabricators* (3rd ed..). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781119287568

Sanier. (2024). *Die Höhe von First und Traufe*. Verfügbar 2. Januar 2025 unter https://www.sanier.de/dach/dach-ratgeber/die-hoehe-von-first-und-traufe

Santiago, D. (2019). *GDC Vault - Procedurally Crafting Manhattan for 'Marvel's Spider-Man'*. Verfügbar 2. Januar 2025 unter https://gdcvault.com/play/1026415/Procedurally-Crafting-Manhattan-for-Marvel

Smelik, R. M., Tutenel, T., Bidarra, R., & Benes, B. (2014). A Survey on Procedural Modelling for Virtual Worlds. *Computer Graphics Forum*, *33*(6), 31–50. https://doi.org/10.1111/cgf.12276

Smith, G., Gan, E., Othenin-Girard, A., & Whitehead, J. (2011). PCG-based game design: enabling new play experiences through procedural content generation. *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. https://doi.org/10.1145/2000919.2000926

Somanath, S., Naserentin, V., Eleftheriou, O., Sjölie, D., Wästberg, B. S., & Logg, A. (2023). On procedural urban digital twin generation and visualization of large scale data. https://doi.org/10.48550/arXiv.2305.02242

Stadt Osnabrück. (2022). *Bebauungsplan Nr. 629 In der Steiniger Heide*. Verfügbar 2. Januar 2025 unter https://bauen.osnabrueck.de/fileadmin/bauen_wohnen/B_629_Stand_25032022.pdf

Stepper, H., & Wietzel, I. (2012). Durchmischung verstehen – neue Einsatzfelder von 3D-Stadtmodellen zur Visualisierung und Simulation urbaner Prozesse. *Proceedings REAL CORP 2012*.

Stiny, G. (1980). Introduction to Shape and Shape Grammars. *Environment and Planning B: Planning and Design*, *7*(3), 343–351. https://doi.org/10.1068/b070343

Stiny, G., & Gips, J. (1971). Shape Grammars and the Generative Specification of Painting and Sculpture. In C. V. Freiman, J. E. Griffith & J. L. Rosenfeld (Hrsg.), *Information Processing, Proceedings of IFIP Congress 1971, Volume 2 - Applications, Ljubljana, Yugoslavia, August 23-28, 1971* (S. 1460–1465). North-Holland.

Stühler, H. U., & Schimpfermann, C. (2023). *Baunutzungsverordnung Kommentar unter besonderer Berücksichtigung des deutschen und gemeinschaftlichen Umweltschutzes*. Kohlhammer. https://doi.org/10.17433/978-3-17-042241-4

Tharma, R., Winter, R., & Eigner, M. (2018). An approach for the implementation of the digital twin in the automotive wiring harness field. In D. Marjanović, M. Štorga, S. Škec, N. Bojčetić & N. Pavković (Hrsg.), *Proceedings of the DESIGN 2018 15th International Design Conference* (S. 3023–3032). https://doi.org/10.21278/idc.2018.0188

Togelius, J., Champandard, A. J., Lanzi, P. L., Mateas, M., Paiva, A., Preuss, M., & Stanley, K. O. (2013). Procedural Content Generation: Goals, Challenges and Actionable Steps. In S. M. Lucas, M. Mateas, M. Preuss, P. Spronck & J. Togelius (Hrsg.), *Artificial and Computational Intelligence in Games* (S. 61–75, Bd. 6). https://doi.org/10.4230/DFU.Vol6.12191.61

Vanegas, C. A., Garcia-Dorado, I., Aliaga, D. G., Benes, B., & Waddell, P. (2012). Inverse design of urban procedural models. *ACM Trans. Graph.*, *31*(6). https://doi.org/10.1145/2366145.2366187

Völkl, S. (o.J.). *Baulinie, Baugrenze, überbaubare Grundstücksfläche? Das müssen Sie wissen*. Verfügbar 2. Januar 2025 unter https://bautipps.almondia.com/bauplanung/hausplanung/ueberbaubare-grundstuecksflaeche/

Wagner, A. (2024). *Wintergarten auf der Grundstücksgrenze*. Verfügbar 2. Januar 2024 unter https://www.fensterbau-ratgeber.de/wintergarten/hintergrund/wintergarten-auf-der-grundstuecksgrenze/

Winter, H. (2023). *Handbuch: 3DProjektplaner*. Verfügbar 2. Januar 2025 unter https://www.connectedurbantwins.de/app/uploads/2024/01/2024-01_Handbuch-3D-Modeller_final.pdf

Winter, H. (o.J.). *3DProjektplaner – Stadtentwicklung mit Perspektive*. Verfügbar 2. Januar 2025 unter https://www.connectedurbantwins.de/praxisbeispiele/3dprojektplaner-stadtentwicklung-mit-perspektive/

Wonka, P., Wimmer, M., Sillion, F., & Ribarsky, W. (2003). Instant architecture. *ACM Trans. Graph.*, *22*(3), 669–677. https://doi.org/10.1145/882262.882324

XLeitstelle. (2023). *Leitfaden XPlanung*. Verfügbar 2. Januar 2024 unter https://xleitstelle.de/sites/default/files/2023-07/Leitfaden_XPlanung_2_Auflage.pdf

Xu, X., Ding, L., Luo, H., & Ma, L. (2014). From building information modeling to city information modeling [Special issue: BIM Cloud-Based Technology in the AEC Sector: Present Status and Future Trends]. *ITcon*, *19*, 292–307. https://www.itcon.org/2014/17

Xu, Z., Qi, M., Wu, Y., Hao, X., & Yang, Y. (2021). City Information Modeling: State of the Art. *Applied Sciences*, *11*(19). https://doi.org/10.3390/app11199333

Yannakakis, G. N., Martínez, H. P., & Jhala, A. (2010). Towards affective camera control in games. *User Modeling and User-Adapted Interaction*, *20*(4), 313–340. https://doi.org/10.1007/s11257-010-9078-0

Zahid, H., Hijazi, I., Donaubauer, A., & Kolbe, T. H. (2024). Enhancing Realism in Urban Simulations: A Mapping Framework for the German National Standard XPlanung and CityGML. In T. H. Kolbe, A. Donaubauer & C. Beil (Hrsg.), *Recent Advances in 3D Geoinformation Science* (S. 437–457). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-43699-4_28

# Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Ausführungen, die anderen veröffentlichten oder nicht veröffentlichten Schriften wörtlich oder sinngemäß entnommen wurden, habe ich kenntlich gemacht.

Die Arbeit hat in gleicher oder ähnlicher Fassung noch keiner anderen Prüfungsbehörde vorgelegen.

Aurich, 6. Januar 2025

Ort, Datum

Unterschrift

# Anlagen

## Anlage 1

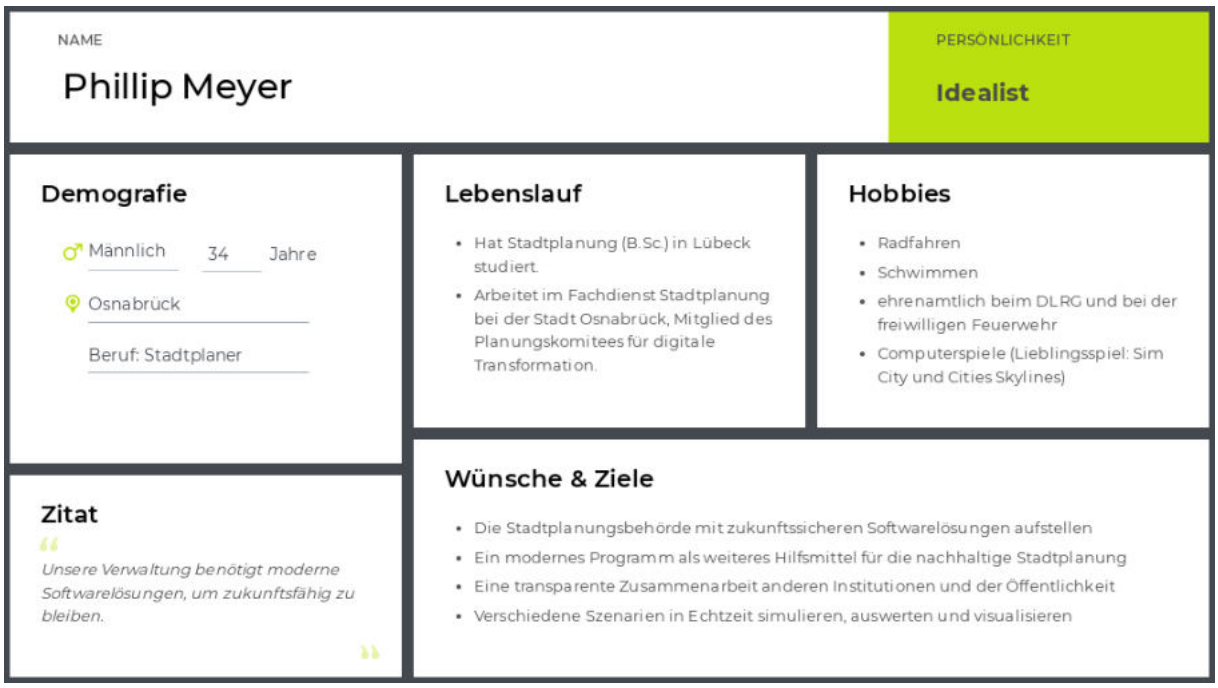


**Abbildung 46:** Persona von Stadtplaner Phillip Meyer (eigene Darstellung, erstellt mit UXPressia)

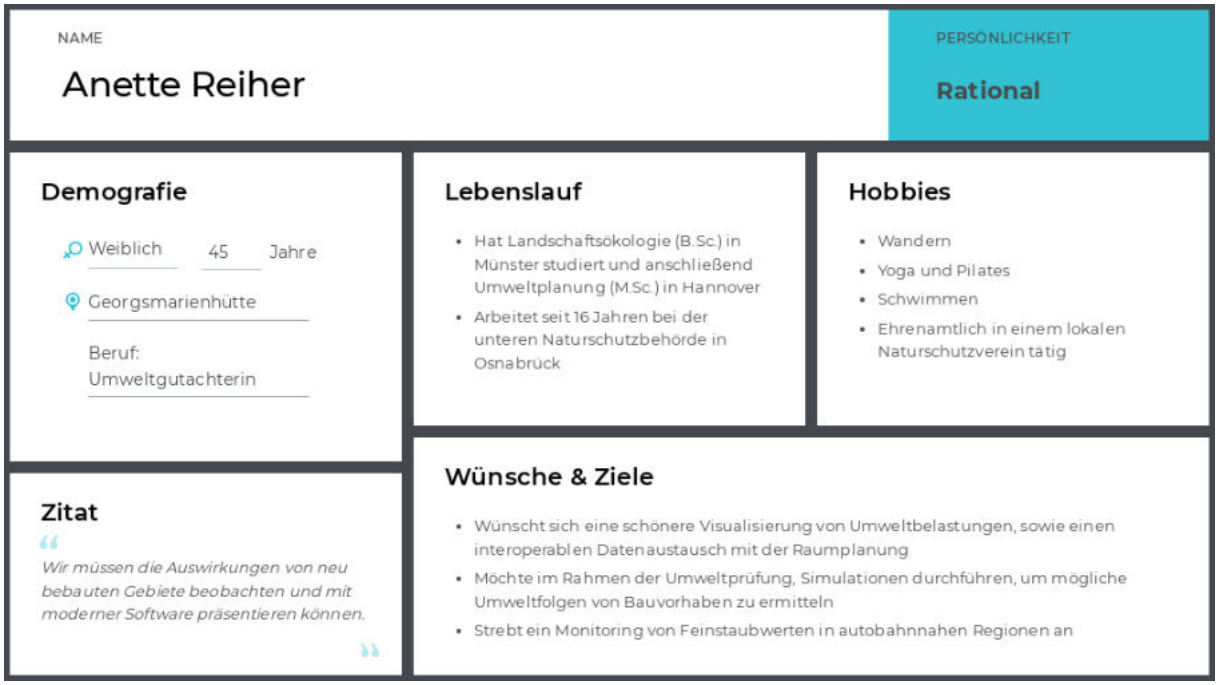


**Abbildung 47:** Persona von Umweltplanerin Anette Reiher (eigene Darstellung, erstellt mit UXPressia)


**Abbildung 48:** Persona von Bauherrin Emily Graf (eigene Darstellung, erstellt mit UXPressia)

# Anlage 2

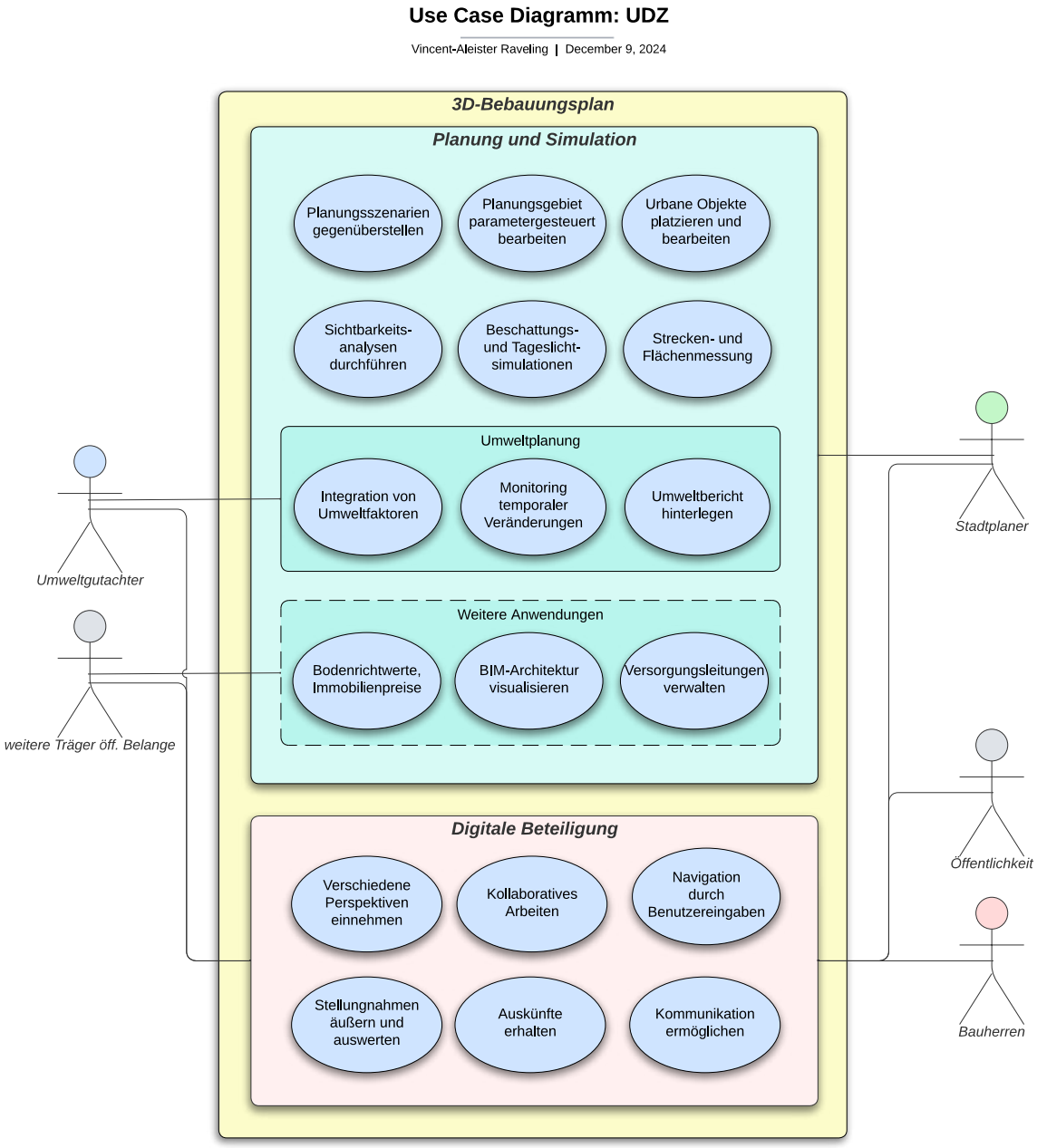


**Abbildung 49:** Use-Case-Diagramm (eigene Darstellung, erstellt mit Lucidchart)

# Anlage 3

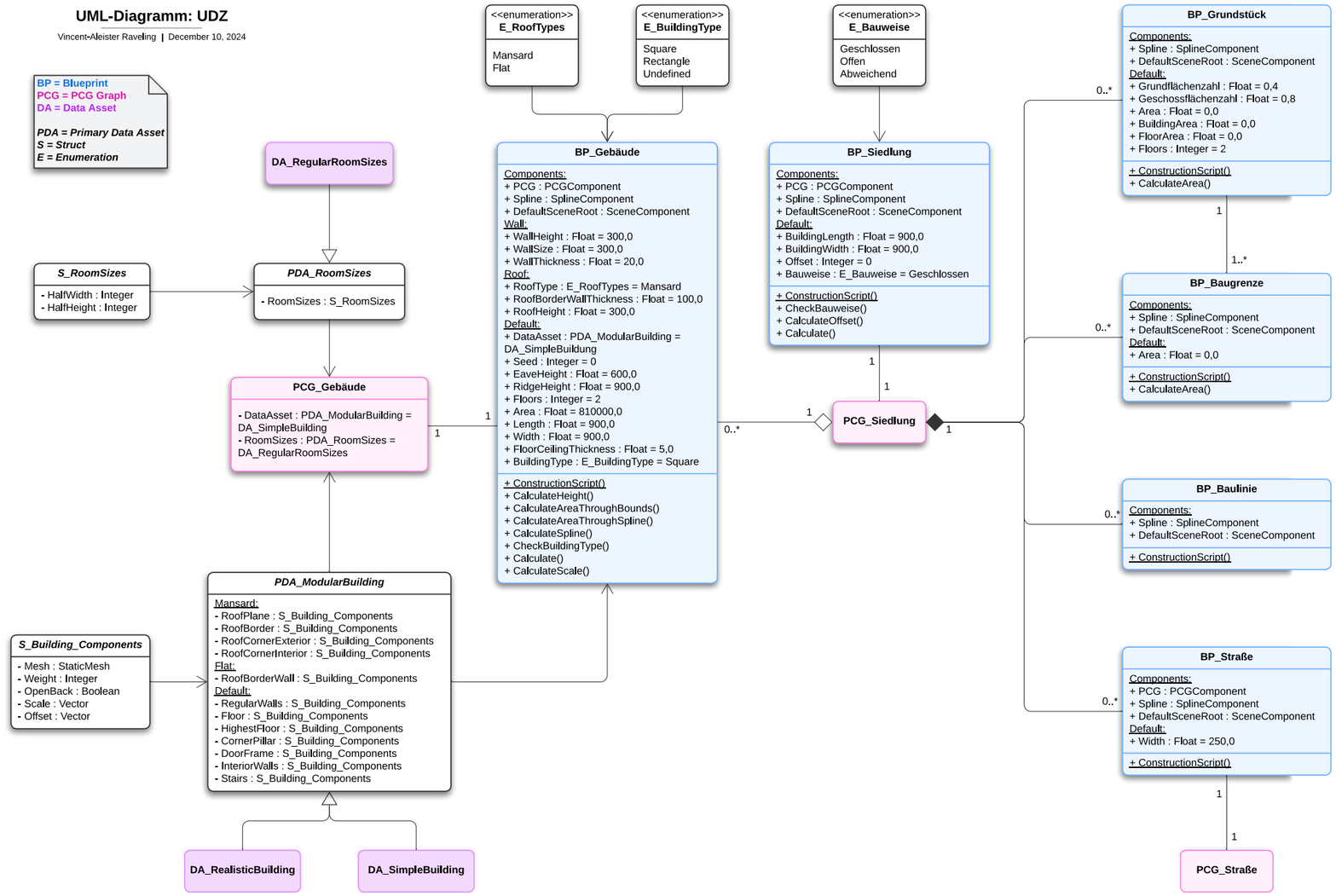


**Abbildung 50:** UML-Klassendiagramm, einschließlich Attribute und Methoden (eigene Darstellung, erstellt mit Lucidchart)

# Anlage 4

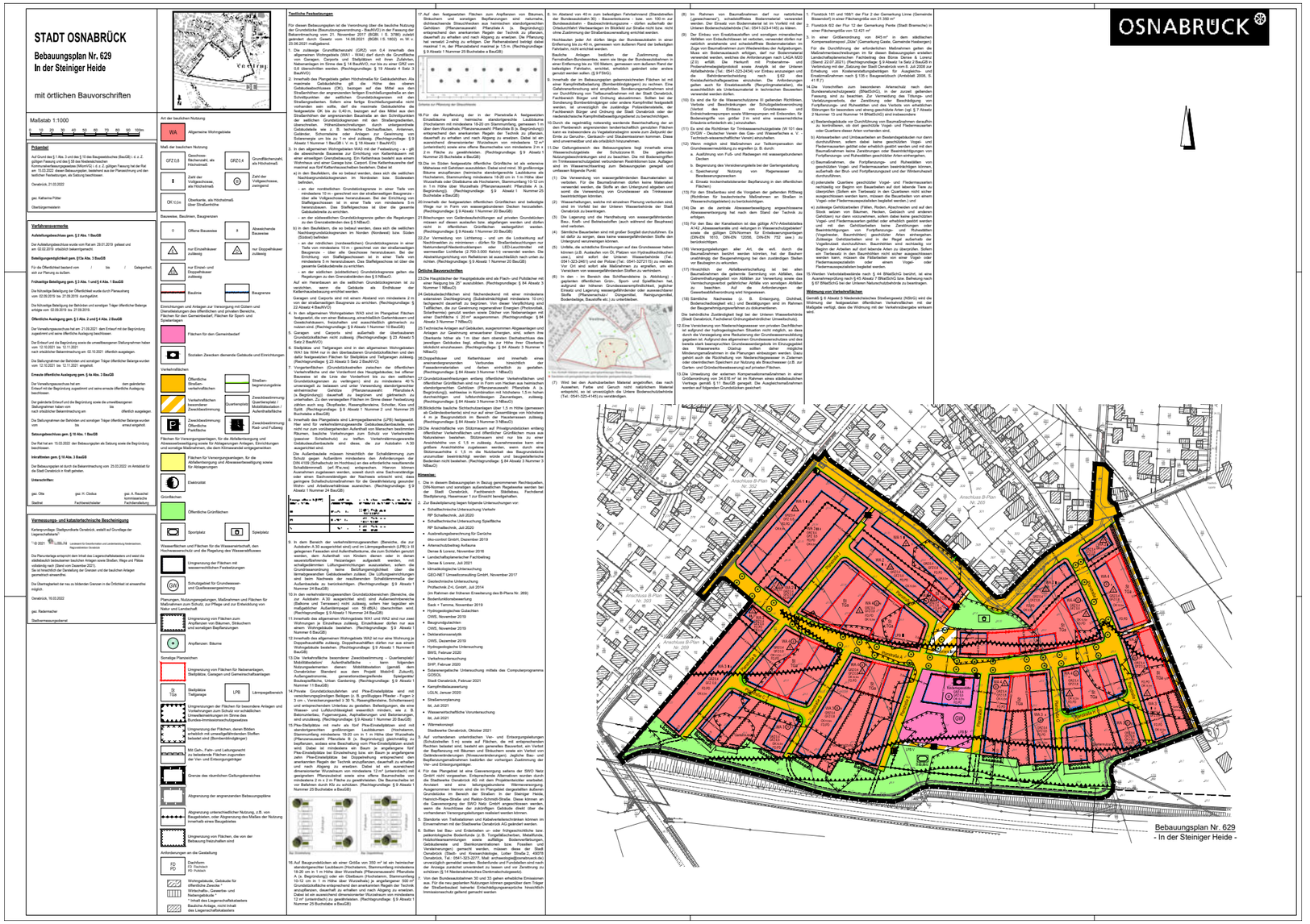


**Abbildung 51:** Bebauungsplan Nr. 629 „In der Steiniger Heide“ (Stadt Osnabrück, 2022)

# Anlage 5

## Im Spiel

Nach dem Start des Spiels können Sie mit einem Avatar durch die Welt navigieren. Für weitere Anweisungen drücken Sie die Taste mit dem Fragezeichensymbol, um das Hilfe-Menü zu öffnen und die Steuerung zu sehen.

Die andere Taste mit dem Pfeil- oder Bleistiftsymbol ermöglicht es Ihnen, zwischen zwei Spielmodi zu wechseln.

### Steuerung

Das Spiel lässt sich wie folgt steuern:

| Taste | Aktion |
|---|---|
| W / Aufwärtspfeil | nach vorne bewegen |
| A / Linker Pfeil | nach links bewegen |
| S / Abwärtspfeil | nach hinten bewegen |
| D / Rechter Pfeil | nach rechts bewegen |
| Q | Flugmodus ein- / ausschalten |
| Leertaste | Springen / nach oben bewegen |
| Strg | nach unten bewegen |
| Umschalt | schneller Rennen |
| E | Türen öffnen / schließen |
| R | Position zurücksetzen |
| TAB | Kameramodus wechseln |
| Esc | alle Menüs schließen / Spiel beenden |
| Mausbewegung | Kamera bewegen (1st / 3rd) |
| Scrollrad | Kamera bewegen (Top-Down) |
| Linke Maustaste | Selektion |

### Spielmodi

Es gibt zwei Spielmodi:

1. **Spielmodus**: In diesem Modus kann der Spieler durch die Szene mit der Third-Person- oder First-Person-Kameraperspektive navigieren.
2. **Editiermodus**: In diesem Modus wird die Attributtabelle aktiviert, die es dem Spieler ermöglicht, die Szene zu bearbeiten. Außerdem wird die Kameraperspektive auf eine Top-Down-Ansicht geändert.

### Attributtabelle

Oben im Menü befinden sich sechs Schaltflächen:

1. **Selektion zurücksetzen** (Roter Pfeil)
2. **Gebäude auswählen** (Gebäude mit Maus-Symbol)
3. **Gebäude hinzufügen** (Gebäude mit + Symbol)
4. **Gebäude entfernen** (Gebäude mit - Symbol)

5. **Grundfläche editieren** (Form mit Maus-Symbol)
6. **Änderungen übernehmen** (Grüne Schaltfläche mit Häkchen)

Nachdem Sie eine Option ausgewählt haben, können Sie die Attribute aller ausgewählten **Gebäude** sowie der gesamten **Siedlung** bearbeiten.

# Anlage 6

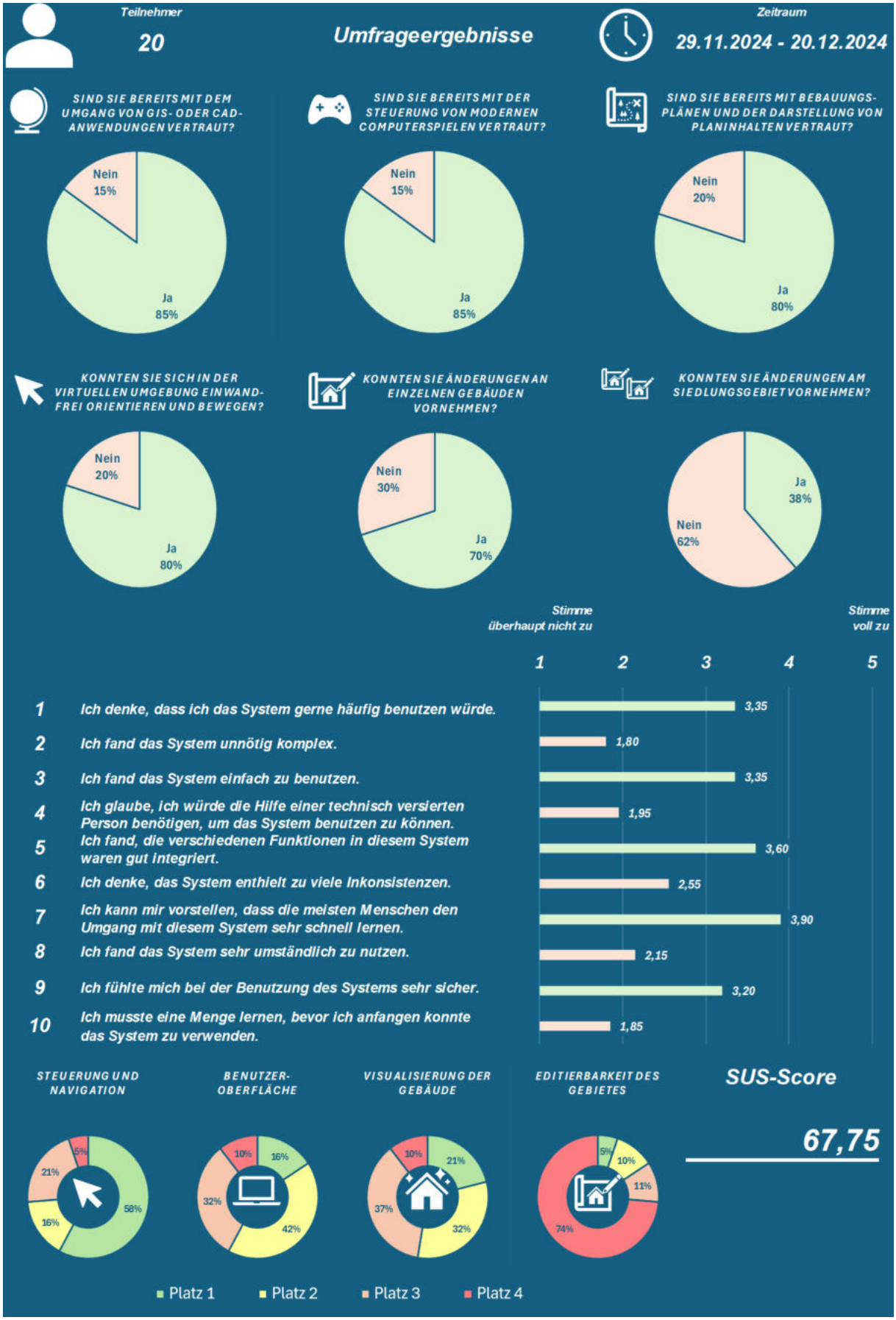


**Abbildung 52:** Ergebnisse des Fragebogens (eigene Darstellung, erstellt mit Excel)

# Anlage 7

1048222 **BA_Raveling**

Die Teilnehmer können nicht teilnehmen.

## Offene Angaben

21.12.2024 16:56:45

Frage:

Welche Probleme traten bei der Steuerung und Orientierung auf?

| Nr. | Grundlagen 1a (v_16) |
|---|---|
| 9 | -bei bestimmten Gebäuden war der Boden höher als die Umgebung, dadurch war ein betreten der Gebäude nicht möglich -ich bin einmal durch den Boden gefallen -Kamerasteuerung durch klicken und bewegen der Maus umständlich -> besser wäre Drehung der Kamera durch Maus bewegen ohne klicken |
| 13 | Nach öffnen der Türen funktionieren die Pfeiltasten nicht mehr. Wenn die Flugmodus Taste im Gebäude gedrückt wurde, konnte die Figur nicht mehr laufen sondern nur noch schweben (auch nach beenden des Flugmodus) |
| 19 | -unschöne Darstellung beim (Wechsel zum) Flugmodus (nicht klar erkennbar, in welchem Modus man ist) -Kamera dreht nicht automatisch in Laufrichtung -Pfeil-Richtungen in Top-Ansicht teilweise verdreht -Pfeil-Tasten funktionieren nicht immer -2x "abgestürzt" im Editier-Modus |
| 26 | Die Bewegung funktionierte einwandfrei, allerdings gab es ein paar Probleme bei der Orientierung. Beim Generieren von neuen Objekten waren diese auf dem ersten Blick nicht ersichtlich. |

Frage:

Welche Probleme traten bei der Editierung von einzelnen Gebäuden auf?

| Nr. | Grundlagen 2a (v_18) |
|---|---|
| 9 | -Gebäude konnten gelöscht und platziert werden, jedoch wurden meine getätigten Eingaben der Gebäudemaße nicht übernommen |
| 13 | Nach löschen des Gebäudes konnte ich das Gebäude nicht mehr ausrichten und auch keine Gebäudemaße verändern (die Eingabe funktionierte aber aber das Gebäude hat seine Größe nicht verändert. |
| 14 | Es ist unklar wann man ein Gebäude ausgewählt hat. Den Grundriss ändern macht gar nichts nach dem Neugenerieren, aber vorher hab ich es auch nicht hinbekommen (Aber ich hab immerhin irgendwelche Spline Points gesehen). |

| | |
|---|---|
| 19 | -Maus reagiert auch hinter der geöffneten Attribut-Anzeige (neue Gebäude erstellen) - bestehende Gebäude ändern: nicht getestet |
| 21 | Die Bedienung des Bau Menü gestaltete sich schwierig. Editierte Gebäude können nicht gedreht oder verschoben werden, Veränderung nur nach Auswahl möglich Gebäude waren nach Auswahl einfach zu löschen. Grundflächeneditor funktionierte nicht. Bei Änderung der Traufhöhe 1200 verschwand das Dach. Die Firsthöhe war nicht zu verändern, allerdings Stockwerke konnten zugefügt werden. Bei der Auswahl Gebäudetyp kamen nur quadratische und keine rechteckigen Gebäude nach Auswahl im Drop down. Länge und Breite der Gebäude, oder die Fläche, konnten nicht editiert werden, bei Änderung verschwand das Gebäude. Bei Änderung der Wandhöhe auf 600 tauchte optisch ein zweites Stockwerk auf. Änderungen der Wand- und Dachhöhe visuell sichtbar. Leider nur zwei Dachformen aus wählbar, Umstellung funktionierte allerdings nach mehreren |
| 26 | Gebäude konnten grundsätzlich erstellt werden. Eine Änderung am Gebäude konnte erst nach mehrmaligen Versuchen umgesetzt werden. Die Kenntnis über die Abfolge zum Editieren war nicht eindeutig. |

**Frage:**

`Welche Probleme traten bei der Editierung am Siedlungsgebiet auf?`

| Nr. | Grundlagen 3a (v_20) |
|---|---|
| 9 | -Gebäude konnten gelöscht und platziert werden, jedoch wurden meine getätigten Eingaben der Gebäudemaße nicht übernommen |
| 10 | Editierkarte war schwierig zu verschieben, zudem war es kaum zu erkennen welches Editierwerkzeug gerade in Benutzung ist und was sich konkret verändert hat. Es dauerte bis man herausgefunden hat, wie man Gebäude löscht, da gerade das weiße Editierfenster kaum auf der Karte zu sehen war. Beim Ändern des Dachtyps musste man zudem das alte Dach nochmal anwählen um es dann erst verändern zu können. Schön wäre es vielleicht auch, wenn man in der Edierkarte sehen kann wo sich die Person befindet (vielleicht auch mit der Ausrichtung wohin diese schaut, um sich besser zurechtzufinden) |
| 13 | Habe ich nicht versucht |
| 14 | Ich konnte es neu generieren aber es hat z.B. die Flachdacheinstellung nicht übernommen. |
| 19 | nicht getestet |
| 21 | Unter dem Menü war keine Funktion zu Verwenden. Keine Speichermöglichkeit. Leider war kein zweiter Test Möglich, da bei Neustart leider das Programm abstürzt. Auch Nach Neuinstallation nicht. |
| 26 | Eine Änderung an Siedlungsgebieten wurde nicht getestet. |

**Frage: Welche Features fehlen Ihnen in dieser Anwendung?**

| Nr. | Feedback 1 (v_21) |
|---|---|

| | |
|---|---|
| 9 | -Man könnte die Auswahl im Menü noch visualisieren. Ich hatte ein Gebäude ausgewählt, aber wusste nicht ob ich es jetzt ausgewählt habe oder nicht da sich die Oberfläche nicht verändert hat. -Beim auswählen fand ich die Steuerung nicht so intuitiv. Bei Windows z.B. muss die linke Maustaste gedrückt gehalten werden beim Ziehen eines Fensters. -Ein Hovertext im Menü, welcher die einzelnen Knöpfe erklärt wäre super. Ohne diese Erklärung fühlt man sich unsicher einen Knopf zu drücken, da man nicht weiß was passiert |
| 10 | Die konkrete Erstellung eines Gebäudes (nicht wahllos durch klicken in der Karte); evtl. Verknüpfung mit planerischen Details (Grenzabstände, etc) |
| 11 | links/rechtsclick mit der maus, um sich in der top down perspektive zu bewegen shortcut eingabefunktion bessere visuelle darstellung von anpassungen am gebäude, zb boden |
| 12 | Weitergehende Analysen und Simulationen und Screenshot-Funktion |
| 13 | Das wechseln der Perspektive könnte durch drücken auf das Kamerasymbols geschehen. |
| 14 | Outlines oder Highlights beim editieren. Soforiges Feedback beim neugenerieren (jetzt macht es ca 10 s lang nix und dann alles auf einmal) |
| 17 | Steuerung vereinfachen ( nicht zwingend die Linke Taste gedrückt halten) Legende einbauen ( Darstellung welche Symbole, welche bedeutung haben zukunfstkonzept: verschiedene Kartengrundlagen beifügen können. Komprimieren mit Liegenschaftskarte um eventeuell weitere Voraussplanungen machen zu können. |
| 19 | Änderung der B-Plan-Grundlage in schwarz-weiß, um mehr zu denken, dass man im Baugebiet steht (vielleicht als Auswahl wie bei GIS Layern) |
| 20 | Neu platzierte Gebäude sollten nur innerhalb der Baugrenzen plaziert werden können. Verschieben von Baugrenzen Größere Vielfalt an Gebäuden |
| 21 | Speichermöglichkeit in der Anwendung |
| 24 | optional: Verlinkung zu Planungsunterlagen (B-Plan -textlicher Teil) Switchen zu verschiedenen Kartengrundlagen (Luftbild, Plan, LiKa?) optional: On/off: bei Texturen/ Detailgrad Gebäude |
| 25 | Wenn man in die Karte2x tipp das man aus der aktuellen Funktionen raus kommt. |
| 27 | Hilfreich wäre es, wenn man in dem Bearbeitungsmodus den Standpunkt des Männchens in dem Kartenausschnitt sehen und dieses aus dem Bearbeitungsmodus heraus an die gewünschte Stelle platzieren könnte, um die Veränderungen direkt zu sehen und das veränderte Gebäude nicht erst wieder suchen zu müssen. |
| 28 | Anzeige der erlaubten Parameter aus dem Bebauungsplan bei der Bearbeitung eines Gebäudes. |

**Frage:**

Haben Sie noch weitere Bemerkungen?

Nr. Feedback 2 (v_34)

| | |
|---|---|
| 9 | -Ich würde mir noch eine Einheit im Menü wünschen. So kann ich die Maße der Gebäude verändern, aber ich habe keine Bezugsgröße zur Realität. Im Bebauungsplan ist die Einheit Meter angegeben. Diese könnte man im Menü ebenfalls nutzen. -Bei der Steuerungshilfe im Menü sind die Tasten (CTRL) auf Englisch. Auf einer deutschen Tastatur sucht man CTRL vergeblich. |
| 10 | Vielleicht beim scrollen über die Symbole einblenden der jeweiligen Funktion; in der Editierfunktion teilweise schwer zu erkennen, was das System jetzt genau macht |
| 12 | Splines nicht so gut sichtbar, mehr visuelles Feedback einbauen |
| 14 | Hab das Ganze in 5.5 getestet. Kann also sein, dass dadurch einiges kaputt gegangen ist. |
| 18 | Anwendung war ein bisschen Träge, das Laden von Features war teilweise verzögert. Ursache beim Endnutzer oder wo anders unklar |
| 19 | Steuerung funktioniert grundsätzlich gut: im Spiel-Modus: sehr intuitiv; im Editier-Modus: Bedienung nicht sehr intuitiv. Automatisches drehen in Lauf-Richtung wäre schön. Auswahl im Editier-Modus ist nicht optimal: man sieht nicht, was ausgewählt ist; Bearbeitungen lassen sich nicht rückgängig machen Endanwendung eher im Planungsbereich bei Kommunen, nicht beim Katasteramt |
| 20 | Intuitivere Bedienung |
| 21 | Leider stürzte das Programm nach Neustart ab und ich kann Sie nicht wieder verwenden. Habe Sie versucht neu zu installieren und es funktioniert nicht! The UE-BPlan3D Game has been crashed and will close Fatal error! |
| 24 | Ein schönes Tool zur Visualisierung und Transparenz von planerischer Gestaltung: ich hoffe, dass die Weiterentwicklung der Anwendung auch nach Ende der BA erfolgt |
| 25 | Ich find das Programm wirklich gut. Natürlich ist es noch ausbaufähig aber es ist ja noch ein Prototyp. Man könnte das Programm gut ans gemeinden und Städte verkaufen. Ich wurde an euer Stelle wenn deine Beta Version fertig ist. Wurde ich es kostenlos für 3 Monate der Stadt oder Kommunen zur Verfügungen stellen damit die ein vernünftiges Feedback geben können. |
| 27 | Man könnte verschiedene Gebäudetypen einbauen. Die Idee finde ich super und mit etwas Feinschliff würde die Software bestimmt sehr hilfreich sein und könnte auch das Interesse von anderen Firmen wecken. |
| 28 | Das Auswählen wäre mit durch das Ziehen mit der Maus intuitiver. Grundfläche editieren ist verwirrend. |