HAW HAMBURG

BACHELORTHESIS
Christian Schulz

Semi-interaktive prozedurale Generierung von Straßennetzen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Christian Schulz

Semi-interaktive prozedurale Generierung von Straßennetzen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Philipp Jenke
Zweitgutachter: Prof. Dr. Stephan Pareigis

Eingereicht am: 23. Dezember 2021

Christian Schulz

Thema der Arbeit

Semi-interaktive prozedurale Generierung von Straßennetzen

Stichworte

Straßennetz, prozedurale Generierung, Graphen, Straßen, Grafik, Generation von Städten, jMonkey

Kurzzusammenfassung

Mit prozeduraler Generierung können in kurzer Zeit ganze (digitale) Städte errichtet werden. Als Herzstück von urbanen Gebieten wird die Generierung von Straßennetzen in dieser Arbeit genauer betrachtet. Aufgrund der Vielfalt der erzeugten Straßenpläne entspricht das Ergebnis nicht immer dem Wunschbild des Anwenders. Hilfreich ist es deshalb, wenn der zufallsgetriebene Entwurf steuerbar ist. Anhand des Forschungsstandes werden Möglichkeiten zur Modellierung von Straßennetzen untersucht. Mit ausgewählten Konzepten wird ein Prototyp erstellt, der die automatisierte Generierung von Straßennetzen in einer bestehenden Simulationsumgebung erlaubt. Die Praktikabilität der Konzepte wird dabei am Ergebnis bewertet. Es zeigt sich, dass die prozedurale Generierung von Straßennetzen eine überzeugende Alternative zur Modellierung per Hand oder durch echte Kartendaten darstellt. Für die Steuerung des Ergebnisses existiert eine Vielzahl an Möglichkeiten mit unterschiedlichen Einwirkungsgraden. Einige Konzepte eignen sich jedoch nur bedingt um glaubwürdige Stadtpläne zu konstruieren. Die Auswahl der Konzepte muss daher auf die jeweiligen Zielvorstellungen zugeschnitten sein.

Christian Schulz

Title of Thesis

Semi-interactive procedural generation of road networks

Keywords

road network, procedural generation, graphs, roads, graphic, generation of cities, jMonkey

Abstract

With procedural generation, entire (digital) cities can be built in a short time. As the heart of urban areas, the generation of street networks is considered in more detail in this work. Due to the variety of street plans generated, the result does not always correspond to the user's desired image. It is therefore helpful to be able to control the random-driven design. The possibilities of modelling road networks are discussed on the basis of the current state of research. With selected concepts, a prototype is created that allows the automated generation of road networks in an existing simulation environment. The practicability of the concepts is evaluated on the basis of the results. The procedural generation of road networks is a convincing alternative to modelling by hand or by real map data. A variety of possibilities with different degrees of influence exist for controlling the result. However, some concepts are only conditionally suitable for constructing credible city maps. The choice of concepts must therefore be tailored to the respective objectives.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Nach [9] ist die prozedurale Generierung[1] die automatische Erstellung von digitalen Gegenständen durch vordefinierte Algorithmen und mit minimalen Nutzereingaben. Neben Straßen können auch Gelände, Vegetation und Gebäude generiert werden. Häufig stehen diese Bestandteile von virtuellen Welten in einem Abhängigkeitsverhältnis zueinander. Die automatisierte Erzeugung der Welten ist zumeist schneller als die Gestaltung durch eine Person. Demgegenüber steht, dass das automatisch erzeugte Ergebnis nicht immer den Vorstellung des Designers entspricht. Deshalb ist es von Vorteil die Generierung richtungsweisend zu steuern und das Ergebnis anzupassen.

Generierte Welten werden sowohl innerhalb von Spielen benötigt, als auch für Simulationen von Städten, für effiziente Transportwege, als Grundlage für digitales Fahrzeugtraining oder zum Testen von Algorithmen der künstlichen Intelligenz. Ein Straßensystem bildet dabei das Hauptmerkmal einer Stadt.

1.2 Ziel der Arbeit

Das Ziel dieser Arbeit ist es einen Straßengenerator zu entwickeln, mit dessen Hilfe lebendige Städte automatisiert erstellt und simuliert werden. Die Basis dieses Generators bildet ein bestehendes Projekt mit Teilfunktionen. Diese Funktionen ermöglichen bereits das manuelle Setzen von Straßenteilen und Kreuzungen. Auch das Simulieren von Fahrzeugen auf Straßen werden durch das Teilprojekt ermöglicht.

[1] https://de.wikipedia.org/wiki/Prozedurale_Synthese , abgerufen am 9.6.2021

1.3 Aufbau der Arbeit

Zu Beginn der Arbeit im Kapitel 2 werden zunächst andere Forschungsergebnisse zum Thema Straßengenerierung beleuchtet. Die damit verbundenen Grundlagen werden im Kapitel 3 erläutert. Für einen Prototypen eines Straßengenerators werden im darauffolgenden Kapitel 4 Anforderungen und eine Lösungsskizze vorgelegt. Ausgewählte Aspekte der Umsetzung des Prototypen werden in Kapitel 5 beschrieben. Im Kapitel 6 folgt eine Analyse der Ergebnisses und in Kapitel 7 werden die Ergebnisse und Erfahrungen zusammengefasst. Zum Schluss folgt ein Ausblick mit möglichen Erweiterungen des Prototyps.

2 Stand der Technik

Nach [11] können Ansätze zur prozeduralen Generierung in regelbasiert, beispielbasiert und lernbasiert unterschieden werden. Vielfach nutzen die vorgeschlagenen Systeme auch mehrere Ansätze in unterschiedlichen Situationen oder für unterschiedliche Aufgaben. Eine scharfe Abgrenzung ist daher nicht immer möglich.

2.1 Straßenlayout

2.1.1 Regelbasierte Systeme

Regelbasiert bedeutet in diesem Kontext, dass Regeln entweder manuell oder automatisch erzeugt werden, nach welchen sich die Erstellung von Objekten richtet. Im Gegensatz zu statistisch orientierten Systemen sind die Straßenparameter häufig nach Heuristiken gewählt.

Gieriges Wachstum

Zu den regelbasierten Systemen zählt die Verwendung von L-Systemen in [20]. Das L-System allein reicht allerdings nicht für die Straßengenerierung aus, da die benötigten Regeln in diesem Fall zu komplex wären. Deshalb werden die Teile, welche die Straßeneigenschaften bestimmen als Funktionsaufrufe außerhalb des L-Systems berechnet. Das überbleibende L-System ist die Verwaltung einer Vorrangwarteschlange von Straßensegmenten bzw. Vorschlägen [16]. Das L-System kann als Algorithmus insofern leichter lesbar dargestellt werden[1].

Die Straßeneigenschaften unterteilen sich in lokale Bedingungen und globale Ziele.

[1] https://nothings.org/gamedev/l_systems.html , abgerufen am 23.08.2021

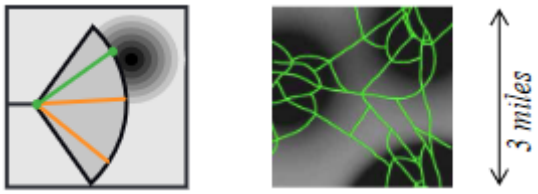

Abbildung 2.1: Links: Wahl eines Strahles anhand der höchsten Bevölkerungsdichte. Rechts: Generiertes Schnellstraßensystem nach dem Prozess aus [20]. Dunklere Gebiete deuten auf eine hohe Bevölkerungsdichte hin.

Die globalen Ziele bestimmen die Eigenschaften eines vorläufigen Vorschlags. Dieser Vorschlag setzt sich aus einer Karte der Bevölkerungsdichte sowie des gewählten Straßenmusters zusammen.

Am Ende einer Schnellstraße werden in einem Bogen mehrere Strahlen gesetzt und anhand der Bevölkerungsdichte entlang dieser Strahlen wird ein Vorschlag ausgewählt. Im Ergebnis werden so durch die Schnellstraßen bevölkerungsreiche Stadtkerne miteinander verbunden. In Bild 2.1 wird der Prozess verdeutlicht.

Normale Straßen können neben der Bevölkerungsdichte auch bestimmten vordefinierten Mustern folgen. So sind alle Straßen eines Gittermusters nach einem global definierten Winkel angelegt. Straßen in einem Kreismuster werden um einen fixierten Punkt herum gelegt.

Die Muster bleiben durch das Folgen der Bevölkerungsdichte eingeschränkt indem die Auswertung nur eine begrenzte Auswahl der Segmentlänge und Abzweigungswinkel zulässt. Verschiedene Muster lassen sich gleichzeitig anwenden. Dafür benötigt jedes Muster eine eigene Eingabekarte mit Grauwerten für die Angabe einer Anwendungsintensität. Die Musterregeln werden separat ausgewertet, die gefundenen Segment-Parameter aufsummiert und mit den Grauwerten ihrer Eingabekarte gewichtet.

Erzeugte Straßen reduzieren die umliegende Bevölkerungsdichte und stoppen ihr Wachstum, wenn die Bevölkerungsdichte in ihrem Endgebiet unter einer bestimmten Grenze liegt.

Weiterhin wird der Vorschlag mit durch die lokalen Bedingungen validiert und sofern möglich auf valide Werte angepasst. Es kann passieren, dass der Vorschlag in ein Parkgebiet, Wassergebiet oder eine andere Straße gerät. Durch Änderung der Länge des Segmentes und Anpassung des Winkels wird die Straße an Ufergebieten entlang oder über

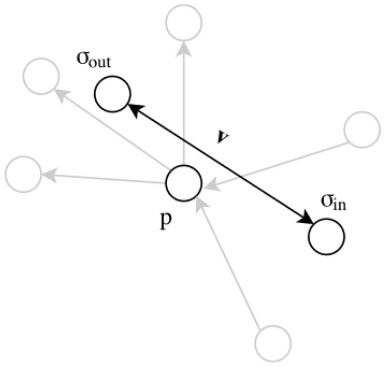

Abbildung 2.2: Berechnung einer Folgestraße mittels ein- (σ_{in}) und ausgehenden (σ_{out}) Verbindungen, aus [30]

Flüsse hinweg geleitet. Bei einem Schnitt mit anderen Straßensegmenten wird in einem bestimmten Radius, versucht eine Kreuzung zu schaffen. Falls keine valide Lösung möglich ist, wird das vorgeschlagene Segment verworfen.

Alle Segmente werden zunächst als mehrere Geraden generiert und erst in einem finalen Schritt müssen Straßen mit starken Biegungen mittels Subdivision [3] geglättet werden.

In [16] wird das Straßennetz als planarer Graph beschrieben. Die Kanten sind Straßensegmente und kennen ihre Start- und Endknotenpunkte. Zudem kennen die Knotenpunkte alle anliegenden Kanten. Die Straßensegmente und Knotenpunkte werden darüber hinaus in einem Quadtree (zu deutsch Quaternärbaum) gehalten. Dies führt zu einer effizienten Eingrenzung von Kandidaten für die Kreuzungsfindung in den lokalen Bedingungen. Schließlich werden die Grauwertkarten zur Eingabe vielmehr automatisch mit Perlin Noise generiert.

Eine Simulation der Straßenerzeugung über Zeit bietet [30]. Für jede Zeitepoche können andere Parameter gewählt werden. So zeigt das generierte System auf Bild 2.3 einen typischen organisch verlaufenden Dorfkern. An den entfernteren Ausläufern, welche in einer späteren Zeitepoche gesetzt werden, ist der Straßenverlauf formaler.

Für die Straßengenerierung wird für jeden Zeitschritt eine zufällige Kreuzung gewählt. An dieser Kreuzung wird eine durchschnittliche Position aller ausgehenden σ_{out} und aller eingehenden σ_{in} Verbindungen berechnet. Mit dem Vektor $\sigma_{in}\vec{\sigma}_{out}$ und einer Abweichung wird die Richtung einer neu entstehenden Verbindung bestimmt (Bild 2.2). Hierdurch wächst das Straßennetz nach außen.

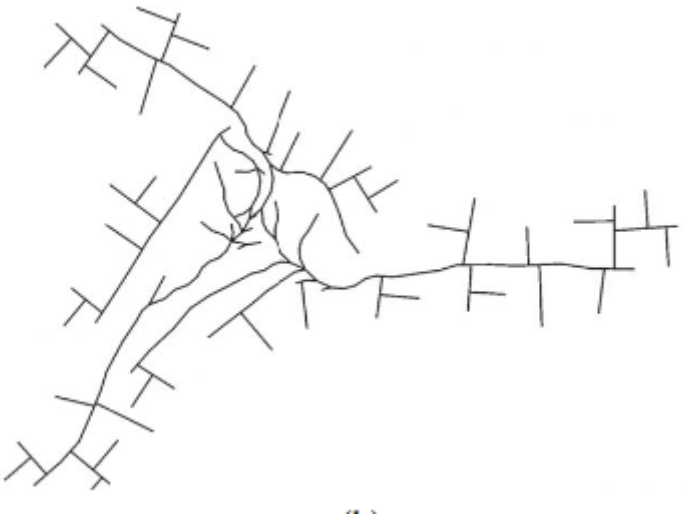

Abbildung 2.3: Über mehrere Zeitepochen generierte Karte, aus [30]

Um nach einem Zeitschritt ein vollständiges Ergebnis zu generieren, prüft [28] bei jedem Schnitt bzw. Einschnappen auf umschlossene Gebiete. Schleifen werden hierbei durch Traversieren über eine Half-Edge-Datenstruktur gefunden. Knoten bestehender Segmente werden für die Erweiterung des Straßennetzes gewählt, wenn ihr Grad nicht mehr als drei beträgt oder es noch nicht genügend fehlgeschlagene Erweiterungsversuche gab. Die Richtung des Folgesegmentes wird durch den Grad des ausgewählten Knotens und die Richtung des Vorgängersegmentes bestimmt. Bei einem Knotengrad von eins wird das Folgesegment mit möglicher Abweichung in die gleiche Richtung des Vorgängers gebaut. Bei einem Knotengrad von zwei wird zufällig entschieden, ob eine Abspaltung nach rechts oder links stattfindet. Bei einem Knotengrad von drei wird die übriggebliebene Richtung gewählt (geradeaus, rechts, links).

Die ausgewählten Segmente nach [28] sind zunächst nur geplant. Eine Verkehrssimulation entscheidet darüber, ob auf der geplanten Straße genügend Verkehrsaufkommen herrscht um diese fest zu implementieren. Die Verkehrssimulation wählt bei einem Zeitschritt neu gebaute, hinsichtlich der Anwohner veränderte und ein paar bestehende Segmente als Startpunkte aus. Zielpunkte sind jeweils z.B. nahe gelegene Populationszentren. Für auf dem Weg besuchte Segmente wird das Verkehrsaufkommen erhöht.

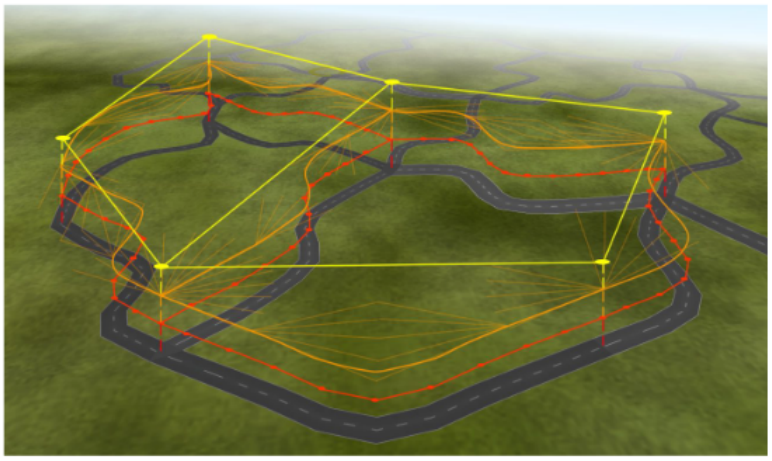

Abbildung 2.4: Verschiedene Graphen für das Plotting von Hauptstraßen aus [13]. Gelb: topologischer Graph, Rot: finale Straßensegmente, Orange: (mögliche) Beispielsegmente und Catmull Rom Spline über ausgewählte Segmente

Seed Systeme

Eine weitere Möglichkeit Straßen zu generieren ist Punkte, *Seeds*, auf einer Karte zu setzen und diese dann miteinander zu verbinden.

In [13] werden die Seeds und ihre Verbindungen manuell gesetzt und in einem topologischen Graphen gehalten. Die tatsächliche Straßenführung muss generiert werden und wird separat in einem weiteren Graphen gehalten. Die ungerichteten, planaren Graphen werden durch Nachbarschaftslisten realisiert. Die Straßenführung wird generiert indem in einem Bogen von 20° zum Zielpunkt mehrere Beispielsegmente gesetzt werden. Von einem dieser Beispielsegmente ausgehend wird der Prozess wiederholt. Darüber hinaus wird das *Sampling* von beiden Endpunkten einer Kante des topologischen Graphen gestartet, sodass sich die Straßenenden in der Mitte treffen. Je nachdem wie die Beispiele für das weitere Propagieren ausgewählt werden kann die Straßenführung an das Gelände angepasst werden. Vorgeschlagene Strategien sind hier möglichst absteigendes Gelände und folglich ein Straßenverlauf ähnlich eines Flusses oder ein möglichst ebener Straßenverlauf. Nach dem Sampling werden die ausgewählten Straßensegmente in einem Catmull Rom Spline zusammengefasst um einen glatteren Straßenverlauf zu generieren. Eine Übersicht über die verschiedenen Stufen in dem Prozess bietet Bild 2.4.

Die so generierten Hauptstraßen bilden in [13] Straßenzellen. In diesen Zellen liegen weitere Straßen, die nach einem gierigen Wachstumskonzept, ähnlich zu [20], generiert

werden. Die Straßenzellen sind kleinst mögliche Schleifen im Graphen und werden mittels eines Minimum Cycle Basis Algorithmus nach [5] generiert.

Neben dem topologischen Graphen und dem Straßengraphen der Hauptstraßen wird jede Straßenzelle mit ihren Hauptstraßen und normalen Straßen in einem separaten Graphen gehalten.

Die Seeds können nach [2] auch auf Grundlage von Geländemodellen automatisch generiert werden. Hier werden die Seeds zudem typisiert. Je nach Populationsdichte kann also ein Stadt-, Ort- oder Dorfseed generiert werden. Das Verbinden der Seeds zu einem topologischen Graphen fängt bei den wichtigsten Stadtseeds an.

Für jeden Seed einer Wichtigkeit wird ein vollständiger Graph betrachtet. Ob eine Verbindung zwischen den Seeds schließlich behalten und als Straße generiert wird, entscheidet ein Optimierungsprozess. Wenn der Nutzen einer Verbindung kleiner als die Kosten sind, wird diese gelöscht. Der Nutzen (B) ergibt sich aus einer Auslastung (U), welche sich aus dem Verhältnis der Populationen (P) in den Endknoten der Verbindung und deren Länge (L) berechnet:

$$U = P_i * P_j / L^2$$

Außerdem beeinflussen die Reisekosten eines Fahrzeugs pro Kilometer (C_{op}) sowie die Differenzlänge zwischen der kürzesten bestehenden alternativen Strecke ($L_{shortest}$) und der zu Prüfenden Verbindung ($L_{connection}$):

$$B = U * (L_{shortest} - L_{connection}) * C_{op}$$

Die dem Nutzen gegenüberstehenden Kosten pro Längeneinheit Straße sind ein Hebel um mehr oder weniger effiziente Verbindungen im Graphen zu erzeugen. Bereits im topologischen Graphen entstehende Schnitte von Verbindungen lassen Knotenpunkte mit geringster Wichtigkeit entstehen. Die jeweils nach den Stadtseeds kommenden Wichtigkeitsstufen von Seeds versuchen eine Verbindung zum bestehenden Straßennetz zu erreichen.

[17] setzt Seeds an den Stellen, an welchen Häuser entstehen sollen, aufgrund von möglichst ebenen Flächen aus einer Geländekarte. Nachdem zwei Punkte durch eine Straße verbunden sind, werden andere Seeds mit der bestehenden Straße durch den A*-Algorithmus verbunden. Hierzu wird die Karte in einen Gittergraph umgewandelt, bei welchem jeder Knoten mit seinen acht nächsten Nachbarn verbunden ist (siehe Bild 2.5). Die Kosten einer Kante ergeben sich aus der Distanz und der Steigung im Startknoten.

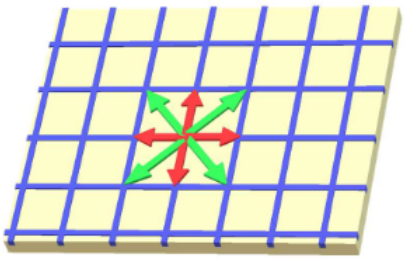

Abbildung 2.5: Gittergraph für A* Wegfindung aus [17].

Die Heuristik ergibt sich aus der Distanz zu dem naheliegendsten Knoten des bestehenden Straßennetzes.

Das Gitternetz wird in [27] noch stärker genutzt. Die Mittelpunkte von benachbarten Zellen eines groben Gitters sind zulässige Knotenpunkte für einen Pfad. Aus den Pfadpunkten wird eine Bezierkurve generiert, welche wieder in belegte und freie Zellen in einem feinerem Gitter übersetzt wird.

Tensorfeld

[4] generiert Tensorfelder auf der Karte, welche durch den Anwender vor Generierung des Straßennetzes angepasst werden können. Durch Überlagerung von Tensorfeldern können Kartengrenzen wie Flüsse und Seen als auch unterschiedliche Nutzeranpassungen beachtet werden. Haupt- und Nebenstraßen haben jeweils eigene Tensorfelder. Die Straßen orientieren sich entlang oder orthogonal zu den zugrundeliegenden Vektoren. Um im resultierenden Straßennetz auch nicht-senkrechte Kreuzungen zu generieren, müssen die Vektoren zusätzlich rotiert werden. Bild 2.6 zeigt die Vektoren, in schwarz/weiß mittels Hyperstreamlines visualisiert und deren Wirkung auf die generierte Straße.

In bestimmten Abständen entlang der Vektoren werden Seed-Punkte in einer Prioritätswarteschlange angelegt, sodass sich eine Straße bildet. Für Querstraßen, welche orthogonal zum zugrundeliegenden Vektorfeld verlaufen werden ebenfalls entlang der Straße Seed-Punkte festgelegt. Eine Straße stoppt ihr Wachstum wenn diese u.a. eine bestimmte Länge erreicht hat, sie eine Kartengrenze erreicht oder nahe einer anderen Straße endet. Somit ähnelt der Wachstumsprozess abgesehen von der Richtungsvorgabe durch die Tensorfelder dem gierigen Wachstum.

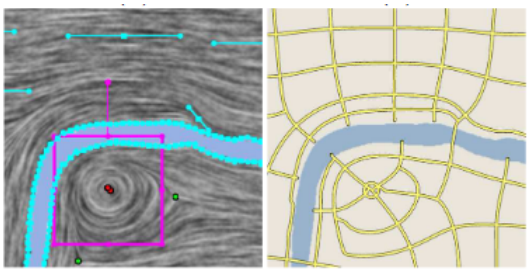

Abbildung 2.6: Links: Verbildlichung der Tensorfelder in Schwarz/Weiß und Farbig von Benutzer gesetzte Designelemente. Rechts: Aus den Tensorfeldern generiertes Straßennetz, aus [4].

Flächennutzung

Die Flächennutzung einer Stadt kann z.B. Wohn-, Gewerbe- und Industriegebiete beinhalten. Die Zuteilung einer Flächennutzung wirkt sich auf das Straßenbild aus, indem in Industriegebiete größere Parzellen entwickelt werden. Wohngebiete hingegen können organischere Straßenverläufe haben. Daneben können den unterschiedlichen Flächentypen die entsprechenden Gebäudetypen zugeordnet werden.

Die Platzierung der Flächen richtet sich in [10] nach Gelände, benachbarte Flächenarten oder Infrastruktur. Nach den Autobahnen werden Seeds um das Stadtzentrum platziert, welche jeweils als mögliche Positionen für ein Distrikt mit einer zugehörigen Flächennutzung in Frage kommen. Jeder Seed wird hinsichtlich der nahegelegenen bestehenden Flächen mit je einem Attraktivitätswert skaliert sowie mit der Distanz bewertet. Hinzukommen weitere geografische Faktoren. Der beste Seed für die zu verwendende Flächennutzung wird ausgewählt. Nachdem alle Flächen platziert sind, entsteht ein Voronoi-Diagramm als Straßennetz mit den ausgewählten Seeds als jeweilige Mittelpunkte.

In [28] können Nutzer eigene Flächentypen implementieren. Neben einem anfangs festgelegtem Verhältnis der Flächentypen werden auch benutzerdefinierte Werteberechnungen einbezogen. So können u.a. nahegelegene Flächentypen, Nähe zu Wasser oder Stadtzentren und das Gelände für die Flächenwahl einbezogen werden. Nach der Erschließung eines Distriktes oder eines Blockes wird seine Flächennutzung bestimmt. Die Parameter der innerhalb des Distriktes gelegenen Straßen richten sich nach dem gewählten Flächentypen. Die Flächennutzung für Blöcke soll die Aufteilung der Bauflächen beeinflussen. Für das frisch umschlossene Gebiet wird für jeden Flächentypen ein Nutzungswert errechnet und der beste Typ ausgewählt.

Eine zufällige Flächennutzung kann durch das Überlagern mehrerer Grauwertkarten, welche jeweils einer bestimmten Flächenart zugeordnet sind, aus Noise-Verfahren erstellt werden. Dieses Vorgehen ist im Ergebnis mit der Überlagerung mehrerer Grauwertkarten in [20] für die Anwendung unterschiedlicher Straßenmuster identisch.

Agenten

[14] verbindet das Konzept der Flächennutzung mit Agenten. Dazu wird die Karte als Raster diskretisiert. Es wird zwischen Straßen- und Zellenagenten unterschieden. Zellenagenten sind verschiedenen Flächentypen zugeordnet. Sie werden auf zufällige, in ihre jeweilige Typen umwandelbare Zellen gesetzt. Valide Zellenverbände für die Umwandlung müssen an Straßen liegen. Dort generieren sie aus den umliegenden Zellen in einem Radius Vorschläge für die Umwandlung in ihren Flächentypen. Wenn ein Vorschlag profitabel ist, wird diese Zelle umgewandelt. Findet der Agent keine profitablen oder umwandelbaren Zellen, wird dieser global umgesetzt. Die Flächennutzung beeinflusst nicht den Straßenverlauf.

Die Parameter für die Straßenentwicklung können auf die Karte gezeichnet werden. Es kann die Größe, Form und Rotation eines Rasters vorgegeben werden. Darüber hinaus gibt es einen Parameter, der bestimmt wie stark das Raster verfolgt werden soll. Durch das Einbringen von unterschiedlichen Parametern auf der Karte entsteht ein heterogenes Straßennetz.

Die Extender-Straßenagenten bewegen sich gemäß der Bedingungen Zelle für Zelle von ihrem Startpunkt weg. Es wird jeweils eine Zelle mit größerem Abstand zum bestehenden Straßennetz gewählt als die vorherige. Die Straßenführung endet, wenn keine anliegende Zelle mehr zu einer Straße umgewandelt werden kann. Der Startpunkt befindet sich auf dem existierenden Straßennetz. Nach Fertigstellung einer Straße oder wenn der Agent einige Zeit keine Straße mehr bauen konnte, wird dieser versetzt.

Verbinder-Straßenagenten bewegen sich und wählen zufällige Zellen in einem Radius entlang bestehender Straßen. Die Verbinder erzeugen eine Straße, wenn der kürzeste Weg zwischen gewählter Zelle und Zelle des Verbinders zu weit ist.

Hauptstraßenagenten sollen nicht an das vorgegebene Raster gebunden sein. Hauptstraßen bilden sich, indem von der aktuellen Agentenposition eine Straße zum nahegelegenen Straßennetz und einem entfernten Ziel gebildet wird.

Das Ergebnis wird vektorisiert[2], um Parzellen und Straßen zu glätten.

Für die Agentensimulation in [25] liegt die Karte in Zellen vor. Jede Zelle kann eine Anzahl von Agenten sowie Landschaftstypen beinhalten. Landschaftstypen können u.a. Wasser, Wald, Gebäude oder auch Straßen sein. Agenten können aus den Landschaftstypen Wasser, Wald, Gebäude und Wiese jeweils unterschiedliche Ressourcen gewinnen. Wasser, Wald und Wiese werden am Anfang nebst einer Start-Straße mit Gebäude zufällig platziert. In jedem Zeitschritt werden alle Agenten in einem *consume-work-trade-rest* Ablauf ausgeführt und die Zellen aktualisiert. Agenten werden auf der Karte verteilt und verbrauchen einen bestimmten Teil der Ressourcen in ihrem Besitz. Den Ressourcentypen, von welchem sie am wenigsten besitzen, versuchen sie auf einer zufälligen Zelle im Radius zu sammeln. Danach wird ein zufälliger Agent in der selben Zelle gewählt und sofern profitabel, werden Ressourcen getauscht. Schließlich bewegt sich der Agent zu einer freien Gebäudezelle für den *rest*-Schritt. Jede Bewegung des Agenten erhöht den Wohlstand der Ziel-Zelle und das Verkehrsaufkommen der mittleren Zelle auf dem Weg. Agenten die ihre Aufgaben nicht erfüllen können werden terminiert. Damit die Zahl der Agenten gleich bleibt, wird ein neuer Agent erschaffen. Neue Agenten entstehen, indem ein bestehender Agent auf der Karte geklont wird. Wenn Wohlstand oder Verkehr einen gewissen Wert auf einer Zelle erreichen, kann die Zelle je nach Typ entsprechend umgewandelt werden.

Freie Zellen können zu Straßenzellen umgewandelt werden, wenn in einem Radius die Summe des Wohlstandes hoch genug ist. Hauptstraßen wandeln mehrere Zellen um sich herum in bebaubare Grundstücke. Auf diesen Grundstücken können normale Straßen entstehen, wenn sie an Gebäudezellen liegen.

Agentensimulationen können realistische Straßennetze generieren. Dem gegenüber steht jeweils eine hohe Laufzeit für die gesamte Simulation [24] [12].

2.1.2 Beispiel-basierte Systeme

Beispiel-basierte Systeme richten das Straßennetz nach einer Vorlage aus. Eine Möglichkeit ist, das Straßennetz nach einem real existierendem Ort zu synthetisieren. Aus dem OpenStreetMap-Projekt[3] lassen sich Kartendaten extrahieren [1]. Diese Daten bedürfen einer Nachbearbeitung, um auch einen Straßengraphen zu generieren. Alternativ kann

[2] https://de.wikipedia.org/wiki/Vektorisierung_(Grafik), abgerufen am 16.07.2021
[3] https://www.openstreetmap.org/about

aus Satellitenbildern ein Straßennetz extrahiert werden [29]. Um auch neue Straßenzüge zu generieren, können Parameter aus der Vorlage für eine zufällige Generierung mittels gierigem Wachstums genutzt werden.

Die Methode aus [19] vereint die Anwendung von statistischen Straßendaten, das zugrundeliegende Gelände und extrahiert interessante Straßenstrukturen aus der Quelle. Aus dem Straßengraphen der Vorlage werden Abstände zwischen Kreuzungen, Gewundenheit der Segmente und Segmentrichtungen analysiert. Diese Parameter bilden die Grundlage für das Straßenwachstum. Daneben werden besondere Elemente wie Kreise, symmetrische Strukturen und nahe beieinander liegende Knoten im Vorlagengraphen extrahiert. Zunächst wird versucht, eines der extrahierten Elemente an den aktuellen Punkt zu setzen. Wenn kein Element passend ist, fällt der Algorithmus auf das gierige Wachstum mit den statistischen Parametern zurück.

In [26] werden in Anlehnung an die beispiel-basierten Systeme auch händisch entworfene Vorlagen in das Straßensystem eingewoben. [31] verformt Vorlagen, sodass sie in polygonale Flächen passen.

2.1.3 Lernen-basierte Systeme

Straßennetze können ebenfalls mit neuronalen Netzen erzeugt werden. In [11] wird einem *generative adversarial network* (GAN) beigebracht, Straßenzüge zu erzeugen, die ähnlich zu einer Vorlage sind. Das GAN besteht aus zwei Spielern, einem Generator und einem Diskriminator. Da zum Training Beispieldaten benötigt werden, ist dieser Ansatz vergleichbar mit den Beispiel basierten-System in [19]. Die Vorlagenzone aus OpenStreetMap wird in Kacheln gerastert. Die Kacheln werden in binäre Pixelkarten umgewandelt. Der Diskriminator wird trainiert, sodass er echte Kacheln aus dem Datensatz von Kacheln, die nicht aus dem Datensatz stammen, unterscheiden kann. Der Generator lernt mit dem Feedback des Diskriminators Kacheln zu erzeugen, die mehr und mehr dem Datensatz ähnlich sind. So kann am Ende des Trainingsprozesses vom Generator eine Pixelkarte erzeugt werden, die dem Stil des Originaldatensatzes ähnelt aber insbesondere durch das Lernen von nur einzelnen Kacheln nicht zu sehr gleicht. Durch das Verfahren werden nicht verbundene Straßenteile und übermäßig viele Sackgassen generiert. Das Ergebnis des Generators ist eine Pixelkarte ohne semantische Informationen. Um einen Straßengraphen und Bauplätze zu finden, bedarf es eines Nachbearbeitungsschritts.

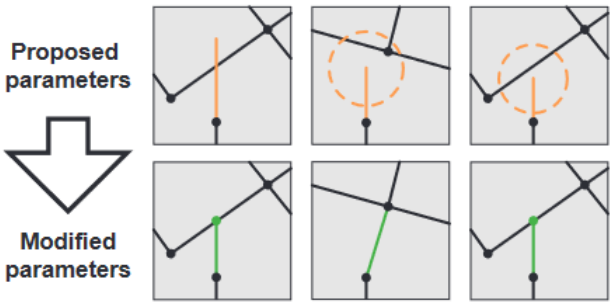

Abbildung 2.7: Beispielhafte Darstellung der Regeln für die Bildung von Kreuzungen aus [20].

2.2 Kreuzungen

In diesem Unterkapitel soll es um Ansätze gehen, die den Schnitt von Straßen behandeln und die damit verbundene Verzweigung des Straßennetzes.

Bei Knotenpunkten mit mehr als zwei Straßensegmenten anliegend wird von einer Kreuzung gesprochen. Eine Kreuzung wird immer entstehen wenn das System bei der Generierung entscheidet, am Endpunkt eines Straßensegmentes mehr als ein Folgesegment zu setzen. Nach [20] werden Kreuzungen zudem geschaffen, wenn das Straßensegment in die Nähe anderer, bereits bestehender Segmente gelangt. Hierzu werden folgende Fälle, wie in Bild 2.7 veranschaulicht, aufgezählt:

1. zwei Straßen schneiden sich direkt und es wird eine neue Kreuzung erschaffen
2. das Segment endet nahe einer bestehenden Kreuzung und wird angepasst, um ebenfalls in dieser Kreuzung zu enden
3. das Segment endet in der Nähe eines bestehenden Segmentes und wird verlängert um eine neue Kreuzung im Schnittpunkt zu bilden

Bei Anwendung der Regeln muss jeweils das nächste Ereignis zum Startpunkt gewählt werden. Nach [16] muss bei Anwendung dieser Regeln letztlich noch eine finale Korrektur unternommen werden. Vor dem finalen Akzeptieren des Segmentes, nach Eintreten einer der o.g. Fälle muss im Endpunkt geprüft werden, ob ein weiteres Segment mit einem zu kleinen Winkel zum aktuellen Segment existiert. In diesem Fall wird das vorgeschlagene Segment verworfen.

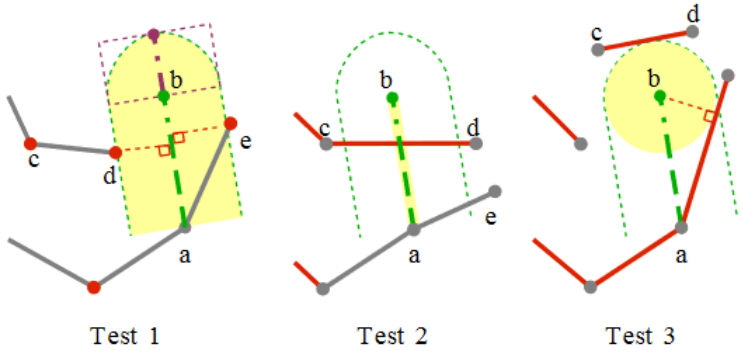

Abbildung 2.8: Darstellung der Regeln für das zusammenschließen von Straßen, aus [13]. Das vorgeschlagene Segment ist grün. Elemente, die getestet werden sind gelb. Im aktuellen Test ignorierte Elemente sind grau.

[13] beschreiben einen effizienten *Snap*-Algorithmus mit naheliegenden Straßen. Wie in Bild 2.8 gezeigt, besteht der Algorithmus aus drei Tests. Test 1 prüft die Entfernung zwischen Knotenpunkten des existierenden Straßennetzes und dem vorgeschlagenen Liniensegment. Zu jedem getesteten Knotenpunkt wird zudem als Variable r berechnet, ob diese relativ näher zum Start- oder Endpunkt liegen. Somit können nur Knotenpunkte getestet werden, die senkrecht zum vorgeschlagenen Segment liegen. Daneben wird ein s-Wert berechnet, der die senkrechte Distanz eines Punktes zum Segment angibt. Ist dieser s-Wert unter einer bestimmten Grenze und liegt der Punkt dem r-Wert nach auch senkrecht zum Segment auf dessen Höhe, so liegt der Punkt im Einschnappbereich. Mit dem kleinsten r-Wert ≥ 0 kann zudem der Schnappunkt identifiziert werden, der am nächsten zum Startpunkt liegt. Der zweite Test prüft auf einen Schnitt mit einem existierendem Segment und nutzt hierzu den s-Wert. Das Vorzeichen des s-Wertes gibt an, auf welcher Seite entlang des vorgeschlagenen Segmentes ein Knotenpunkt liegt. Nur wenn beide Knotenpunkte eines existierenden Segmentes auf einer anderen Seite liegen, ist ein Schnitt möglich. Der dritte Test prüft die Nähe des Endknotens des vorgeschlagenen Segmentes zu bereits existierenden Segmenten. Dieser Test muss nur ausgeführt werden, wenn Test eins und zwei keine Schnappereignisse ergaben. Ein Einschnappen im dritten Test kann dazu führen, dass entweder mit dem nächsten Punkt im gefundenen Segment ein neuer Knotenpunkt entsteht oder falls bereits ein Knotenpunkt in der Nähe besteht, dieser mit dem bestehenden Knotenpunkt verbunden wird. Bild 2.9 zeigt die Berechnung und Wirkung der r/s-Werte.

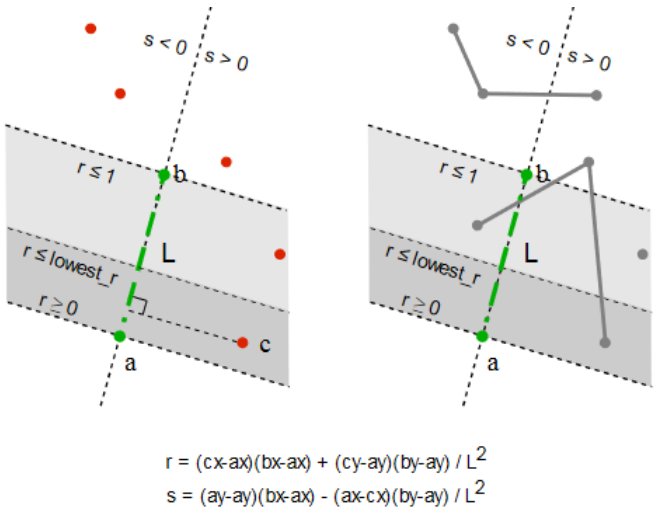

Abbildung 2.9: Berechnung der relativen Positionen von Knotenpunkten (links) und Segmenten (rechts) zum vorgeschlagenen Segment $\vec{AB}$ (grün), aus [13].

2.3 Generieren von Bauflächen

Um an Straßen Gebäude zu generieren müssen Freiflächen gefunden werden. Diese werden typischerweise zudem in Zellen unterteilt auf welchen konkrete Gebäude generiert werden können. Im Folgenden werden Wege beschrieben, Freiflächen und Zellen zu erhalten.

In [20] teilt das Straßennetz das zugrundeliegende Gebiet in mehrere zwischen den Straßen liegende Flächen auf. Weiterhin werden konkave Polygone als Bauflächen ausgeschlossen. Die eigentlichen Bauflächen entstehen durch das Verkleinern von den Straßenknotenpunkten aus, um die Breite der Straße darzustellen. Einzelne Grundstücke entstehen, indem die Polygone wiederholt an den längsten gegenüberliegenden Kanten geteilt werden. Zu kleine Grundstücke oder solche, die nicht an der Straße liegen werden entfernt.

In [13] werden Freiflächen zwischen Straßen mit dem *Minimum Cycle Basis Algorithmus* gefunden, welchen sie auch für das Finden von Stadtzellen nutzen. Weiterhin erlauben sie auch konkave Freiflächen für die weitere Bearbeitung. Die Freiflächen werden auch hier wiederholt mittig, entlang der längsten Kante geteilt bis eine bestimmte Größe erreicht ist. Darüber hinaus priorisieren [13] die Schnitte entlang der Seiten, zu denen das jeweilige Polygon einen Zugang zu einer Straße hat. Polygone ohne Straßenzugang werden schließlich nicht bebaut.

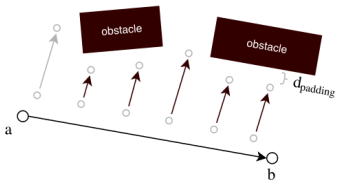

Abbildung 2.10: Strahlen werden von der Straße aus projiziert, aus [30]

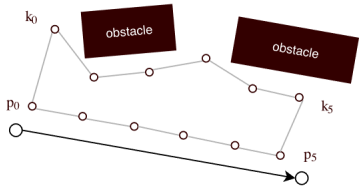

Abbildung 2.11: Aus den Strahlen wird ein Polygon für Bauflächen generiert, aus [30]

Ohne umschlossene Gebiete können nach [30] auch Strahlen im engen Abstand orthogonal zum Straßensegment projiziert werden. Durch Hindernisse werden die Strahlen verkürzt und das finale Polygon für Freiflächen angepasst (Bild 2.11).

2.4 Interaktion

Im Folgenden sollen Ansätze aufgezeigt werden, die den Nutzer das Ergebnis der prozeduralen Generierung von Straßennetzen steuern lassen.

Die Steuerung der Generierung erfolgt in [20] durch vorgegebene Karten mit Grauwerten für Gelände, Wasser, Vegetation, Bevölkerungsdichte, Nutzungsart[4] und Intensitäten von gewünschten Straßenmustern. Daneben kann der Nutzer noch verschiedene Parameter für die Generierung einstellen wie die durchschnittliche Größe von Freiflächen zwischen Straßen und Anzahl von Kreuzungen pro Flächeneinheit. Die Erweiterung der Produktionsregeln für das verwendete L-System ist wohl nicht vorgesehen, da der Einfluss auf Straßeneigenschaften weitestgehend ausgelagert wurde.

[13] lassen den Nutzer durch Kontrollpunkte die Grenzen der entstehenden Stadt sowie deren Hauptstraßen definieren. Durch schlanke Umsetzung der Straßengenerierung, insbesondere die Abfrage ob Straßen sich kreuzen oder verbunden werden sollen, soll eine Verschiebung der Kontrollpunkte eine Neugenerierung des Straßennetzes in Echtzeit liefern. Die verschiebbaren Kontrollpunkte sind im Bild 2.4 gelb dargestellt.

Bei Simulationen wie [14] oder [28] kann ein Benutzer zwischen den Simulationsschritten eingreifen und z.B. durch das Setzen von bestimmten Fokusgebieten die Entwicklung der Stadt steuern.

[4] Wohngebiet, Kommerziell oder gemischt

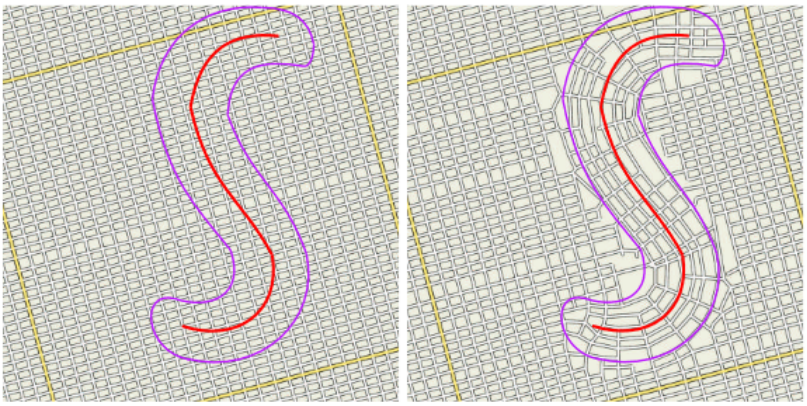

Abbildung 2.12: Manipulation der Straßenorientierung durch das Pinselwerkzeug, aus [4].

Durch die Nutzung von Tensorfeldern kann [4] Höhenkarten einbinden, indem ein globales Tensorfeld an jeder Stelle die Steigung abbildet. Des Weiteren können Elemente wie Kreise oder Geraden mit bestimmter Richtung in die Karte gesetzt werden. In einem Radius werden hierdurch die Vektorrichtungen vorgegeben. Vorgegebene Elemente wie Flüsse ergeben als Polygone zerlegt mehrere gerade Segmente. Durch diese Segmente werden ebenfalls die Vektorrichtungen vorgeben um die Straßenverläufe an den Elementen entlang zu führen. Vom Anwender gesetzte Elemente sowie ein Fluss wird in Bild 2.6 links dargestellt. Zudem erlaubt die Darstellung der Straßenrichtungen vor der eigentlichen Generierung die Nutzung eines Pinselwerkzeuges, um komplexere Straßenmuster einzufügen, wie in Bild 2.12 zu sehen ist.

2.5 Editieren

Die prozedurale Generierung von Straßennetzen ist häufig von zufälligen Initialisierungen, Straßenlängen und Richtungen abhängig. Dadurch ist es für viele Projekte notwendig, das erzeugte Straßenbild manuell anzupassen. Zunächst stehen niedrigschwellige Möglichkeiten zur Verfügung wie löschen, hinzufügen und ändern von einzelnen Straßenknoten sowie das Hinzufügen von Verbindungen zwischen Knoten. Daneben können die Parameter angepasst werden, um die gesamte Karte neu generieren zu lassen. Eine Neugenerierung könnte ebenfalls auch nur auf einen Abschnitt, wie z.B. durch Hauptstraßen abgegrenzte Blöcke, beschränkt sein.

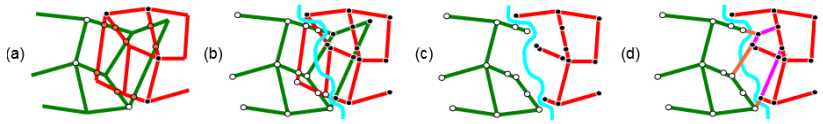

Abbildung 2.13: Graphzusammenführung, aus [15]. a) Zusammengelegte Graphen b) Finden des minimalen Schnittes c) löschen der Kanten in falscher Partition d) Hinzufügen von Straßen zur Verbindung

[15] beschreibt weitergehende Möglichkeiten das Straßennetz zu verändern, dabei aber ein valides und vorhersehbares Ergebnis zu erzeugen. Das Verschieben eines Knotens wird auf neue Kreuzungen überprüft. Sofern keine neuen Schnitte gefunden werden, können alle zur betroffenen Straße benachbarten Freiflächen neu generiert werden.

Änderungen am Straßengraphen werden mittels minimalen Graph-Schnitt [7] realisiert. Das Problem wird als Zusammenlegung von zwei Graphen beschrieben. Die Kantenprioritäten werden manuell festgelegt. Die Graphen werden zusammengelegt und jeder Schnitt ergibt einen neuen Knoten. Sackgassen werden entfernt. Quelle und Senke werden nahe der überlappenden Region gesetzt. Nach dem Schnitt werden alle Knoten in der falschen Partition gelöscht. Um entstandene Löcher wiederherzustellen, werden einzelne Randstraßen nach und nach wieder hinzugefügt. Der Prozess ist in Bild 2.13 verdeutlicht. Bauzellen, die nur von einem Graphen umschlossen sind bleiben erhalten. Zellen, die von Straßen aus beiden Graphen umschlossen werden müssen regeneriert werden.

Daneben wird auch das Konzept von Ebenen genutzt. Die meisten Änderungen können durch den Einsatz von Ebenen und Graphenzusammenführung implementiert werden. Eine Ebene beinhaltet ein Stadtlayout als Straßengraphen. Eine Szene kann eine beschränkte Anzahl von Ebenen haben. Die finale Stadt ist dann das Ergebnis von mehreren Ebenen die in bestimmter Reihenfolge zusammengeführt wurden. Bestimmte Änderungen können so auch vor dem Überschreiben einer Neugenerierung geschützt werden. Teilgraphen können markiert und als neue Ebene an einen anderen Ort gesetzt werden.

2.6 Texturen

[8] beschreibt u.a. die visuelle Darstellung von Straßen und Kreuzungen in einer Übersicht. Straßen werden durch mehrere Punkte definiert. Durch diese Punkte verläuft ein

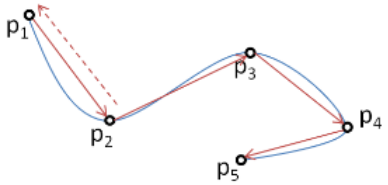

Abbildung 2.14: Spline interpoliert Punkte p1 bis p5, aus [8].

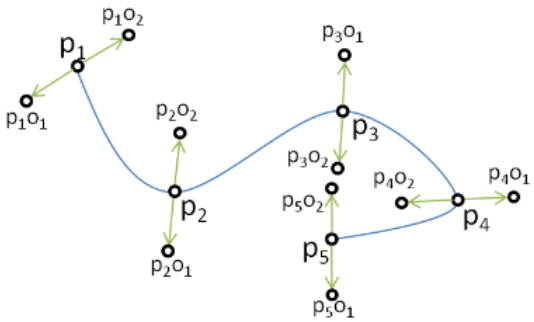

Abbildung 2.15: Splinepunkte in beide Richtungen entlang der Laufrichtung projiziert, aus [8].

Spline (Bild 2.14). Der Spline kann beliebig fein aufgelöst werden um eine Straße zu triangulieren. Von jedem Spline-Punkt P_n wird die Laufrichtung (rote Pfeile in Bild 2.14) errechnet. Anhand der Laufrichtung werden durch orthogonale Vektoren Vp_{n_i} (Bild 2.15) Offset-Punkte errechnet, die links (P_{n_1}) und rechts (P_{n_2}) in Laufrichtung vom Spline-Punkt P_n liegen. Durch die Länge der Vektoren wird die spätere Breite der Straße abgebildet.

Spline-Punkte werden auf eine UV-Karte abgebildet. Die neuen Punkte tragen die UV-Koordinaten (0,d) für P_{n_1} und (1,d) für P_{n_2}. Wobei d die zurückgelegte Entfernung auf dem Spline ist. Um in engen Kurven überlappende Dreiecke zu verhindern, wird für jeden orthogonalen Vektor Vp_{n_i} eine Kollisionsprüfung mit beiden Vektoren Vp_{n-1_i} des vorherigen Spline-Punktes durchgeführt. Wenn eine Kollision ermittelt wird, ist die neue Position des Offset-Punktes an der Stelle des Vorgängers (P_{n-1_i}).

Bei Kreuzungen und Übergängen werden alle Straßenenden einer Verbindung gleichmäßig gekürzt bis sich keine Straßenpolygone mehr schneiden. Die Verbindung erhält gerundete

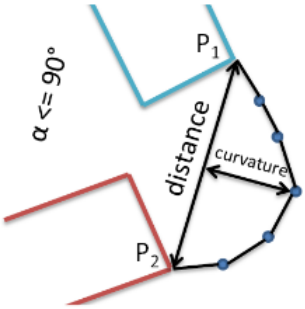

Abbildung 2.16: Erstellung der Rundung für Übergänge, aus [8].

Polygone (Bild 2.16) indem zunächst der Winkel der Verbindung errechnet wird:

$$\alpha = \cos^{-1} * \left(\frac{\vec{d1} * \vec{d2}}{|\vec{d1}| * |\vec{d2}|} \right)$$

Der Winkel wird zudem durch das Kreuzprodukt zwischen $\vec{d1}$ und $\vec{d2}$ auf [0;360]° abgebildet. Übergänge mit einem Winkel zwischen 170° und 190° werden direkt verbunden, da diese nahezu gerade sind. Die Punkte der Kurve werden mit einer quadratischen Funktion errechnet. Diese Kurvatur k ist abhängig von der Distanz d auf $\vec{P_1P_2}$, $d \mapsto [0;1]$ und der maximalen Kurvatur k_{max}. Wobei k_{max} bei d = 0,5 erreicht ist und zu beiden enden der Linie $\vec{P_1P_2}$ abnimmt. k_{max} muss für konkave Rundungen negativ und für konvexe Rundungen positiv gewählt werden. Darüber hinaus wird ein höheres k_{max} für steile Kurven gewählt. Die Kurvatur wird auf die Linie $\vec{P_1P_2}$ der Verbindung projiziert. Im Ergebnis entsteht so eine konkave Kurve für die Innenseite des Übergangs und eine konvexe Kurve für die Außenseite.

Zunächst erhält die Kreuzung eine graue Textur. Das Kreuzungspolygon wird dann entlang seiner Außensegmente abgelaufen. Wenn keine Straße mit dem Segment verbunden ist, wird eine weiße Linie aufgezeichnet. Sonst wird je nach ein- oder ausgehender Straße keine Linie oder eine gestrichelte Linie auf dem Segment eingezeichnet.

3 Grundlagen

Dieses Kapitel behandelt grundlegende Algorithmen und Techniken welche nicht ausschließlich für die Generierung von Straßenzügen genutzt werden können. Die Techniken wurden von den Konzepten aus Kapitel 2 genutzt und spielen zum Teil eine Rolle in der Umsetzung.

3.1 Kurven

Eine Kurve ist eine stetige Funktion, welche in Abhängigkeit des Kurvenparameters t einen Punkt im Raum beschreibt: $f : t \to \mathbb{R}^d, t \in \mathbb{R}$. Durch Anwachsen des Kurvenparameters und Aneinanderhängen der sich ergebenden Punkte entsteht so eine zusammenhängende Kurve.[1] Kurven werden Häufig nur für $t \in [0; 1]$ und durch Polynome definiert. Eine spezielle Kurve ist die kubische Hermite-Kurve. Ein Punkt auf der Hermite-Kurve wird definiert durch $f(t) = (1 + 2t)(1 - t)^2 p_0 + t(1 - t)^2 m_0 + t^2(t - 1)m_1 + t^2(3 - 2t)p_1$ [6]. Die Koeffizienten p_0 und p_1 stellen Start- und Endpunkt der Kurve dar. Dagegen ist der Koeffizient m_0 die Ableitung der Kurve in p_0 und m_1 ist die Ableitung der Kurve in p_1. Der Einfluss der Koeffizienten auf die Kurve ist in Bild 3.1 dargestellt.

Durch das Zusammenfügen von mehreren Kurven an ihren Endpunkten können komplexere geometrische Objekte entstehen. Bild3.2 zeigt eine zusammengesetzte Kurve. Der so entstehende Spline wird an seinen Übergängen geglättet, indem eine Endtangente m_{n+1} einer Kurve mit der Tangente gleichgesetzt wird, welche die benachbarten Kontrollpunkte p_n und p_{n+2} verbindet.[2] Die gewählte Tangente kann ebenfalls skaliert werden, um die Spannung zwischen den Kontrollpunkten zu verändern. Der Cardinal-Spline [23] führt dazu einen Spannungsparameter $T \in [-1; 1]$ ein. Die Tangente m_{n+1} errechnet sich so

[1] https://de.wikipedia.org/wiki/Weg_(Mathematik) , abgerufen am 18.10.2021
[2] https://de.wikipedia.org/wiki/Geometrische_Modellierung#Hermite-Kurven , abgerufen am 18.10.2021

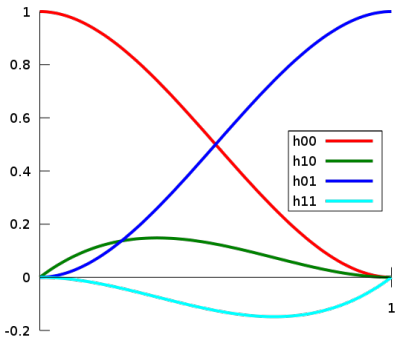

Abbildung 3.1: Einfluss der Koeffizienten auf die Hermite-Kurve. Korrespondierende Koeffizienten: Rot (h00) - p_0; Blau (h01) - p_1; Grün (h10) - m_0; Türkis (h11) - m_1. Aus https://commons.wikimedia.org/wiki/File:HermiteBasis.svg , abgerufen am 18. Oktober 2021.

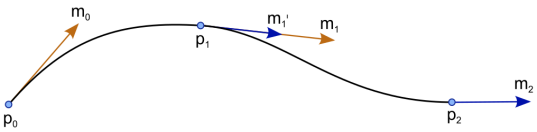

Abbildung 3.2: Spline mit 2 Hermite-Kurven https://commons.wikimedia.org/wiki/File:Hermite_spline_2-segments.svg , abgerufen am 18. Oktober 2021.

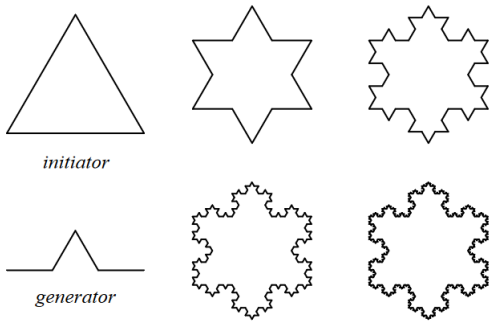

Abbildung 3.3: Erzeugung einer Schneeflocke durch Produktionsregel und Eingabeobjekt, aus [22].

durch $\frac{1-T}{2} * (p_{n+2} - p_n)$. Wobei ein höherer Wert die Tangente kürzt und den Übergang schärfer macht. Ein geringerer Wert verlängert die Tangente und sorgt für einen weicheren Übergang zwischen den Kontrollpunkten.

3.2 L-System

Das Lindenmayer-System [22] ist eine Technik, bei der komplexe Objekte entstehen, indem Teile eines einfachen Eingabeobjektes nach und nach mittels Produktionsregeln ersetzt werden. Im Bild 3.3 werden alle Seiten des initialen Dreiecks durch eine Linie mit einem Zacken ersetzt. In folgenden Iterationen werden auch jeweils alle geraden Linien des Sternes parallel durch eine Zackenlinie ersetzt. L-Systeme sind den Chomsky Grammatiken ähnlich. Sie unterscheiden sich darin, dass im L-System die Produktionsregeln parallel und gleichzeitig alle Buchstaben im Wort ersetzen.

Mit Stapeloperatoren können Verästelungen generiert werden. Die Klammer auf [speichert die momentane Position und Richtung eines Zeigers auf den Stapel während eine geschlossene Klammer] die oberste Position und Orientierung vom Stapel holt. Im Bild 3.4 wird ein Ast durch fünfmalige Anwendung der Ersetzungsregel erzeugt. Die Operatoren + und - verschieben die momentane Zeigerorientierung um +/- 25.7°. Durch die Stapeloperationen bleiben die Zeigerorientierungen lokal. Deterministische L-Systeme erzeugen aus derselben Grammatik identische Objekte. Um die Erzeugnisse variabler zu gestalten, gibt es mehrere Produktionsregeln mit dem gleichen Vorgänger. Die Auswahl der anzuwendenden Produktionsregel richtet sich dann nach einer mitgegebenen Wahr-

Abbildung 3.4: Erzeugung von Verästelungen, aus [22].

scheinlichkeit. Die Erzeugnisse in Bild 3.5 sehen sich noch ähnlich, trotz deutlich unterschiedlicher Ausprägung.

3.3 Perlin Noise

Rauschfunktionen dienen dazu ein Bild zu schaffen, indem die Übergänge zwischen hohen und niedrigen Werten geglättet sind. In einem Rauschen ohne Glättung kann durch wechselnde Extremwerte kein Muster erkannt werden. Geglättetes Rauschen hilft natürliche Phänomene darzustellen. Das Rauschen im zweidimensionalen Bereich kann durch zufällige Werte auf einem quadratischen Gitternetz initialisiert werden. Der Wert eines bestimmten Punktes auf der Fläche errechnet sich durch Interpolation der vier nächstgelegenen Initialwerte zwischen denen er liegt. Perlin Noise[3] [21] skaliert die jeweiligen Gewichtungen der Initialwerte mit einer Polynomfunktion, um die Übergänge zwischen den Extremwerten weicher zu gestalten.[4] Das Ergebnis der Perlinfunktion auf einer Fläche zeigt Bild 3.6.

3.4 Minimum Cycle Base Algorithmus

Der Minimum Cycle Basis Algorithmus aus [5] errechnet diejenigen Kanten eines planaren Graphen, welche ein Gebiet komplett umschließen. Das umschlossene Gebiet wird dabei nicht durch weitere Kanten geteilt.

[3] https://mrl.cs.nyu.edu/ perlin/doc/oscar.html , abgerufen am 23.10.2021
[4] https://de.wikipedia.org/wiki/Perlin-Noise , abgerufen am 23.10.2021

Abbildung 3.5: Erzeugung von nicht deterministischen Verästelungen, aus [22].

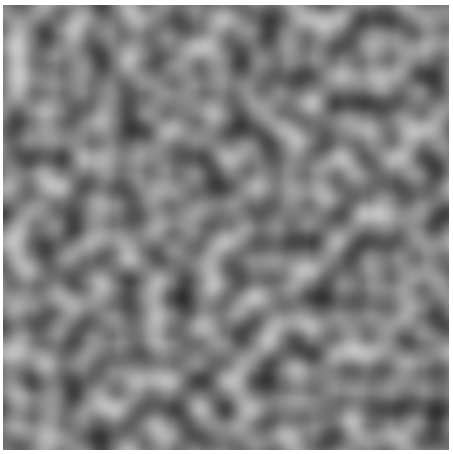

Abbildung 3.6: Perlin Noise auf einer Fläche generiert mit https://github.com/blackears/PerlinNoiseMaker .

Der Algorithmus löscht zunächst alle Sackgassen. Sackgassen sind als Knoten mit nur einem adjazenten Knoten zu erkennen. Zum Erkennen der Kreise wird die Lage der Knoten verwendet. Der Start-Knoten des Algorithmus ist derjenige mit kleinster X-Koordinate im rechtshändigen Koordinatensystem. Von diesem Knoten aus können mehrere Kanten abgehen. Wenn dies der erste Kreis an dem Knoten ist, wird ein Lot in Richtung (0,-1) gesetzt und die Kante zum Ablaufen gewählt, welche den größten Winkel im Uhrzeigersinn zum Lot hat. An allen weiteren besuchten Knoten wird die Kante zum Ablaufen gewählt, welche den größten Winkeln gegen den Uhrzeigersinn hat zur letzten abgelaufenen Kante. Von einem gefundenen Kreis kann die erste Kante aus dem Graphen entfernt werden. Darüber hinaus kann jede auf die erste folgende Kante entfernt werden, wenn sie nur zwei adjazente Knoten hat. Wenn der Startknoten weitere adjazente Knoten hat, die nicht zum gefundenen Kreis gehören, läuft der Algorithmus an diesem Knoten wieder von vorn los. Die Auswahl der ersten Kante richtet sich nunmehr anstelle des Lotes (0,-1) nach der Richtung der letzten gefundenen Kante des letzten Kreises. Andernfalls, wenn der Knoten keine weiteren Nachbarn hat und dieser folglich aus dem Graphen entfernt wird, wird der nächste Knoten nach seinem minimalen X-Koordinaten ausgewählt. In speziellen Fällen können Kreise in Kreisen bestehen. Diese werden durch doppeltes Vorkommen eines Knotens identifiziert. Der Teil zwischen den doppelt vorkommenden Knoten kann aus dem oberen Kreis abgehangen werden und separat mit dem Algorithmus auf eigene Kreise untersucht werden.

Die Kanten der gefundenen Kreise werden in einer Reihenfolge abgelaufen, sodass das Innere stets auf der linken Seite in Laufrichtung ist.

3.5 Quadtree

Der Quadtree oder Quaternärbaum ist eine Baumstruktur, ähnlich dem Binärbaum. Der Quadtree hat an jedem inneren Knoten genau vier Kindknoten.[5] Neben der Datenkompression eignen sich Quadtrees auch zur Bildsegmentierung [18]. Durch die Bildsegmentierung wird ein zweidimensionaler Raum in mehrere Bereiche unterteilt. Wenn ein Blattknoten eine festgelegte Anzahl von Elementen enthält, wird der Blattknoten zu einem inneren Knoten umgewandelt. Der innere Knoten zeigt nunmehr auf seine vier Quadranten, die er beinhaltet. Durch diese Datenstruktur können näherungsweise Elemente aus einer Fläche gefiltert werden, die eine Kollision zueinander haben könnten oder einen

[5] https://en.wikipedia.org/wiki/Quadtree , abgerufen am 24.10.2021

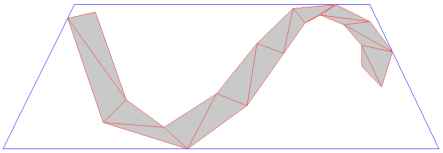

Abbildung 3.7: Dreiecksnetz aus einer kurvigen Straße, aus [8].

Abbildung 3.8: Straßentextur aus dem Vorgabeprojekt.

bestimmten Abstand zu einem Punkt haben. Eine konkrete Kollision muss dann mit den möglichen Kollisionselementen zusätzlich konkret überprüft werden. Im dreidimensionalen Raum wird der Octree analog zur Raumsegmentierung genutzt.

3.6 Texturierung

Das Texturbild eines Objektes wird zunächst erstellt. Die Einteilung der Koordinatenachsen u und v auf der Textur erfolgt üblicherweise im Intervall von [0;1]. Das Polygon, auf welches eine Textur aus Bild 3.8 gelegt werden soll, wird in Dreiecke unterteilt (Bild 3.7). Die Eckpunkte des Dreiecks werden mit Texturkoordinaten (u,v) assoziiert. Die Farbe eines Pixels in dem Dreieck wird durch Baryzentrische Koordinaten errechnet.

Die Baryzentrischen Koordinaten sind drei Skalare, die jeweils eine Gewichtung der Ecken in dem Dreieck für einen Punkt P repräsentieren. Die Repräsentation setzt sich aus der Referenzfläche eines Eckpunktes im Verhältnis zur Gesamtfläche des Dreiecks zusam-

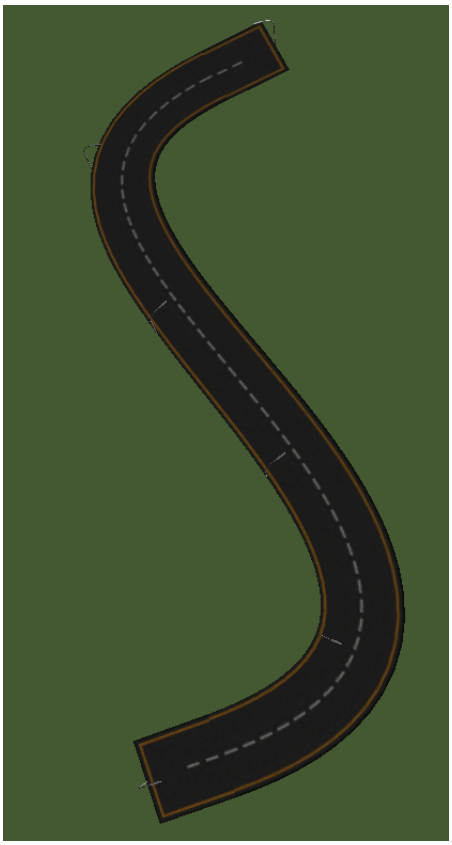

Abbildung 3.9: Gekrümmte Straße mit Textur.

men.[6] Ein Punkt P ergibt sich somit durch $P = \alpha A + \beta B + \gamma C$, wobei A, B, und C die Eckpunkte des Dreiecks sind und α, β, γ die baryzentrischen Koordinaten. Um nunmehr die Texturkoordinaten von P zu erhalten, können die Eckpunkte des Polygondreiecks durch deren Texturassoziationen ersetzt werden.

Das Polygondreieck und das Texturdreieck haben nicht zwangsläufig die gleiche Form. Die Textur wird dann verzerrt oder gestaucht, um auf das Polygon zu passen (Bild 3.9). [6]

[6] Weisstein, Eric W. "Barycentric Coordinates."From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/BarycentricCoordinates.html , abgerufen am 1.11.2012

4 Konzept

Nachdem der Stand der Technik erarbeitet wurde, kann ein Konzept für einen Prototypen geplant werden. Hierzu müssen zunächst Ziele in Form von Anforderungen an ein System festgelegt werden. Daraufhin wird eine Lösungsstrategie zur Erreichung der Anforderungen erörtert. Im Bild 4.1 wird eine Übersicht über den Systemablauf dargestellt.

4.1 Mindestanforderungen

Die Anforderungen werden im Rahmen einer agilen Vorgehensweise erläutert. Zunächst werden Anforderungen behandelt, die an ein minimal existenzfähiges System gestellt werden.

Das erzeugte Straßennetz muss plausibel sein. Insbesondere müssen alle Teile des Straßennetzes zusammenhängen. Außerdem müssen sich die sich überschneidenden Straßenteile eine Kreuzung ergeben. In Kurven und an Kreuzungen soll die Straße eine gleichmäßige Spurbreite aufweisen.

Das Aussehen des Straßennetzes soll durch Parameter steuerbar sein. Zudem soll die Kurvigkeit der Straßenverläufe anpassbar sein und auch die Straßendichte soll sich einstellen lassen.

Darüber hinaus soll der Straßennetzgenerator in das bereits existierende System eingebettet werden. Die bestehenden Funktionen (Abbildung 5.1) dieses Systems sollen dabei nicht eingeschränkt werden. Außerdem ist es das Ziel, dass die Simulation von Fahrzeugen weiterhin möglich bleibt. Zuletzt eine Mindestanforderung, dass die manuelle Erstellung und Verformung von Straßen und Kreuzungen weiterhin möglich bleibt.

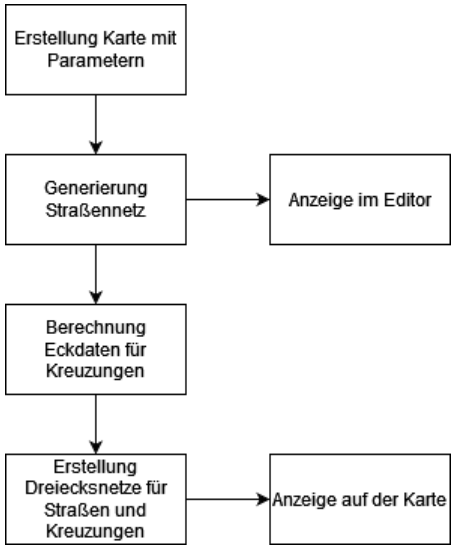

Abbildung 4.1: Übersicht des Systemsablaufes zur Straßengenerierung.

4.2 Weitere Anforderungen

Nach der Sicherstellung der Mindestanforderungen ist die Planung weiterer Funktionen sinnvoll. Es sollten Bauzonen für eine Gebäudegenerierung zur Verfügung gestellt werden. Zudem sollten die Straßen unterschiedliche Hierarchien aufweisen können (z.B. Autobahn, Nebenstraße) und es bedarf Zonen, die unterschiedliche Straßenbilder aufweisen. Die Steuerung der Parameter sollte über die Benutzeroberfläche möglich sein und außerdem sollte bei der Straßengenerierung das Gelände berücksichtigt werden. Das bedeutet, dass Straßen z.B. als Brücke über Wasser oder als Tunnel durch Berge führen. Ähnlich sollte es auch Unter- und Überführungen von Straßen geben.

4.3 Nicht-funktionale Anforderungen

Die Erzeugung des Straßensystems soll um ein vielfaches schneller sein als die Erzeugung durch eine Person. Voraussichtlich werden mehrere Straßensysteme generiert bis der Anwender ein wünschenswertes Ergebnis hat. Das System muss insofern schneller zu einem wünschenswerten Resultat kommen als die manuelle Generierung. Zugleich ist davon auszugehen, dass ein Anwender im prozedural generierten Straßennetz noch Details anpassen muss. Dem gegenüber steht jedoch, dass die prozedurale Generierung dem Anwender die kreative Leistung abnimmt. Auf der kleinsten Karte mit 250 m x 200 m

dauert die manuelle Erstellung eines beliebigen Straßennetzes etwa fünf Minuten. Deshalb soll die prozedurale Generierung auf dieser Karte nicht länger als eine Minute in Anspruch nehmen.

Da das System in einem größeren Projekt entwickelt wird und viele Funktionen sukzessive eingeführt werden, ist es notwendig das System möglichst erweiterbar zu halten. Im Rahmen der agilen Entwicklung ist es nicht möglich alle zukünftigen Funktionen vorher zu sehen. Die Erweiterung des Systems soll insbesondere bei seinen Hauptfunktionen durch Anpassung nur einer Stelle im Bestandscode möglich sein. Hauptbestandteile sind die Behandlung von Kollisionen mit anderen Straßenteilen und Gelände sowie die Ausrichtung neuer Straßenteile.

Die Plausibilität des Straßennetzes soll zudem durch Korrektheit des Systems gesichert sein. Im prozeduralen System entsteht mittels pseudozufälliger Zahlen ein großer Zustandsraum. Die ein- oder mehrmalige Ausführung des Systems wird Fehler nicht zwangsläufig aufdecken. Deshalb wird inkorrekt arbeitenden Teilfunktionen ein erhöhtes Risiko beigemessen.

4.4 Lösungsstrategie

Im Folgendem werden die Herangehensweisen erläutert, die der Erfüllung des vorgenannten Anwendungsfalles und Anforderungen dienen. Als übergreifendes Konzept soll an den Stellen, wo die Literatur unterschiedliche Lösungskonzepte anbietet, möglichst auswechselbare Komponenten genutzt werden.

4.4.1 Datenstruktur Straßennetz

Das Straßennetz ist ein ungerichteter, planarer Graph. Ein einzelner Abschnitt wird mindestens durch seine beiden Endpunkte dargestellt. Weitere Attribute wie Zugehörigkeit zu einem mehrteiligen Straßenzug, Einbahnstraßen und Spurbreite sind auf dieser Datenstruktur ebenfalls denkbar, werden zunächst aber zurückgestellt.

4.4.2 Auswahl Straßenlayout Vorgehensweise

Die regelbasierten Systeme zur Generierung von Straßenlayouts haben den Vorteil, dass sie keine Vorlagen benötigen. Dementsprechend sind sie auch etwas intuitiver steuerbar und implementierbar. Durch die Steuerung mittels Parameter lässt sich mit den regelbasierten Systemen eine breitere Palette an Stilen umsetzen. Ein Nachteil ist die schwierige Umsetzung von komplexen Strukturen wie Kreisverkehren oder Autobahnabfahrten.

Der große Vorteil von Lern- und Beispielansätzen ist die Generierung realistischerer Städte. Komplexe Strukturen aus der Vorlage finden sich häufig im Endergebnis wieder. Darüber hinaus muss sich ein Anwender durch die Extraktion der Daten aus Vorlagen weniger in die Domäne des Straßenbaus einarbeiten. Diese Systeme sind durch Parameter weniger flexibel steuerbar. Schließlich werden auch Datensätze als Vorlagen benötigt. Solche Datensätze sind eventuell nicht immer verfügbar.

Insbesondere wegen der Steuerbarkeit und Unabhängigkeit von Datensätzen der regelbasierten Systeme fällt die Wahl für die Implementierung eines Prototyps auf diese. Unter den regelbasierten Systemen erscheint das gierige Wachstum besonders intuitiv und es wird auch vermehrt umgesetzt. Daneben garantieren sie durch das Wachstum von einem existierenden Straßenpunkt aus, dass der Straßengraph zusammenhängend ist. Deshalb fällt die Wahl des Ansatzes für das Straßenlayout auf die Algorithmus-Gruppe des gierigen Wachstums.

Wie im Kapitel Stand der Technik dargestellt, kann das gierige Wachstum durch L-System oder einer expliziten Nutzung der Vorrangwarteschlange realisiert werden. Das L-System aus [20] hat den Nachteil, dass es durch die Produktionsregeln schwer lesbar ist. Daneben erfolgt auch dort die Generierung nur teilweise im L-System. Die Abfragen zu den globalen und lokalen Zielen, welche schließlich durch Winkel und Straßenlänge das Straßenbild erheblich mitbestimmen, erfolgen außerhalb des L-Systems. Der Vorteil neue Straßennetze durch nur Produktionsregeln der Grammatik zu definieren geht dabei verloren. Durch die Verwaltung der Vorrangwarteschlange lässt sich präzise die Reihenfolge steuern, in welcher Straßen gesetzt werden. So könnten Straßen in der Nähe eines Stadtzentrums bevorzugt werden.

Mangels Metriken wie Stadtzentren, Gelände oder Populationskarten in dieser Entwicklungsphase, kann anstelle einer Vorrangwarteschlange auch eine einfache Warteschlange eingesetzt werden. Neue Straßenpunkte werden am Ende einer gesetzten Straße platziert. Von diesen Straßenpunkten aus werden neue Straßensegmente vorgeschlagen. Die neuen

Straßenpunkte werden jeweils am Kopf der Warteschlange platziert. Durch dieses Last In - First Out Prinzip werden zunächst lange Hauptstraßen gebildet und erst bei Terminierung eines Straßenzuges bilden sich sukzessiv kleinere Verästlungen als Nebenstraßen.

4.4.3 Systemablauf und Komponenten

Aus der Wahl der grundsätzlichen Vorgehensweise ergibt sich ein grundsätzlicher Systemablauf. Initial wird ein Anfangspunkt für das Straßennetz gefunden. Dieser Startpunkt könnte eine bestehende Straße, der Kartenrand oder ein beliebiger Punkt in der Karte sein. Da das Straßennetz hier zunächst nur auf einer leeren, rechteckigen Karte generiert wird, genügt die Kartenmitte als Startpunkt.

Straßenansatzpunkte sind die Enden eines Straßensegmentes, an welches ein neues Straßensegment angelegt werden soll. Der Startpunkt wird zusammen mit den Straßenansatzpunkten in einer Warteschlange verwaltet. Für das Setzen eines neuen Straßensegmentes wird das vorderste Element extrahiert. Die Ausrichtung des Straßensegments richtet sich nach der Straßenrichtung des Vorgängers, sofern das neue Segment Teil eines zusammenhängenden Straßenzuges ist. Andernfalls wird die Straßenrichtung zufällig gewählt. Diese Vorauswahl, bestehend aus Startpunkt und Richtung, wird durch einen Vorschlaggenerator verändert. Die Ausrichtung eines Vorschlages muss einen ausreichend großen Winkel zu allen anderen inzidenten Segmenten haben.

Anschließend wird der Straßensegmentvorschlag auf Kollision mit einem Geländeelement oder einer anderen Straße abgeprüft. Grenzen zählen zu den Geländeelementen. Im Falle einer Kollision mit dem Gelände könnte der Vorschlag verworfen oder angepasst werden, sodass er nicht mehr mit dem Gelände kollidiert. Da hier nur Grenzen als Geländeelemente vorhanden sind, kann der kollidierende Vorschlag verworfen werden. Bei Kollision mit Gewässern oder Bergen könnte eine Anpassung in Betracht gezogen werden. Die Geländekollisionen laufen in einer eigenen Komponente ab, um Erweiterungen leichter zu ermöglichen.

Eine Kollision mit einem anderen Straßensegment führt zu einem Einrasten mit diesem. Ein Straßenkollisionsmodul prüft Kollisionen mit anderen Straßen und gibt das geänderte Straßensegment zurück. Der Schnapp-Algorithmus aus [13] gibt die Möglichkeit die Umgebung des Straßenvorschlags nach anderen Straßenteilen abzuprüfen. Es werden somit nicht nur direkte Schnitte mit anderen Segmenten gefunden. Weiterhin ist die Einschnappumgebung einstellbar und der Algorithmus liefert die erste Kollision vom

Startpunkt aus. Das geänderte Straßensegment kann dadurch keine Kollision mit anderen Straßensegmenten haben.

Durch die Änderung des vorgeschlagenen Segments ist nunmehr wieder eine Kollision mit dem Gelände möglich. Zudem muss in beiden Endpunkten des Vorschlags geprüft werden, ob die Winkel zu allen inzidenten Segmenten ausreichend groß sind. Schließlich muss der Abstand einer eventuell neu erzeugten Kreuzung zu bestehenden Kreuzungen evaluiert werden. Sofern nach einer Änderung des vorgeschlagenen Segments diese Voraussetzungen nicht erfüllt sind, wird es verworfen.

Falls das vorgeschlagene Segment kollisionsfrei gesetzt werden kann, wird über eine Zufallswahl entschieden, ob am Ende des Segmentes eine Verästelung durch 2-4 Straßenansatzpunkte erfolgt. Mindestens ein Straßenansatzpunkt wird schließlich am Kopf der Warteschlange hinzugefügt. Bei Auffinden einer Kollision wird der Straßenzug terminiert. Es werden keine weiteren Straßenansatzpunkte der Warteschlange hinzugefügt.

Aus dem Systemablauf lassen sich Komponenten ableiten. Die Verwaltungskomponente hat die Aufgabe zu entscheiden, wo neue Straßenteile angesetzt werden und wann diese verworfen werden. Die Vorschlagkomponente generiert hinsichtlich des Startpunktes valide Straßenvorschläge. Die Geländekollisionskomponente prüft und behandelt ein Vorschlagsegment bezüglich Kollisionen mit Geländeelementen. Auf derzeitigem Stand gibt es hier lediglich die Kartenränder und das Element wird nicht angepasst. Schließlich hat die Straßenkollisionskomponente die Aufgabe, Kollisionen mit anderen Straßenteilen zu finden und einen kollisionsfreien Vorschlag zu erzeugen. Bild 4.2 zeigt den Zusammenhang der Komponenten in einer Bausteinsicht nach dem *arc42 Template*[1]. Das Laufzeitdiagramm in Bild 4.3 zeigt das Zusammenspiel der Komponenten. Im dargestellten Szenario wird der Straßenvorschlag durch eine Kollision mit einem anderen Straßenteil verändert und bedarf einer zweiter Prüfung durch die Grenzenkomponente.

4.4.4 Visualisierung

Die einfache Verbindung der Segmente führt zu kantigen und unnatürlichen Straßenteilen. Um die Straßen glatter erscheinen zu lassen, werden die Punkte eines Straßenzuges durch eine Kurve interpoliert. Im bestehenden System sind Straßen durch Hermite-Kurven definiert. Durch die Anpassung der Tangenten kann durch mehrere Hermite-Kurven ein interpolierender Spline dargestellt werden. Die Ableitung eines Dreiecksnetzes für die

[1] https://www.arc42.de/overview/ , abgerufen am 17.11.2021

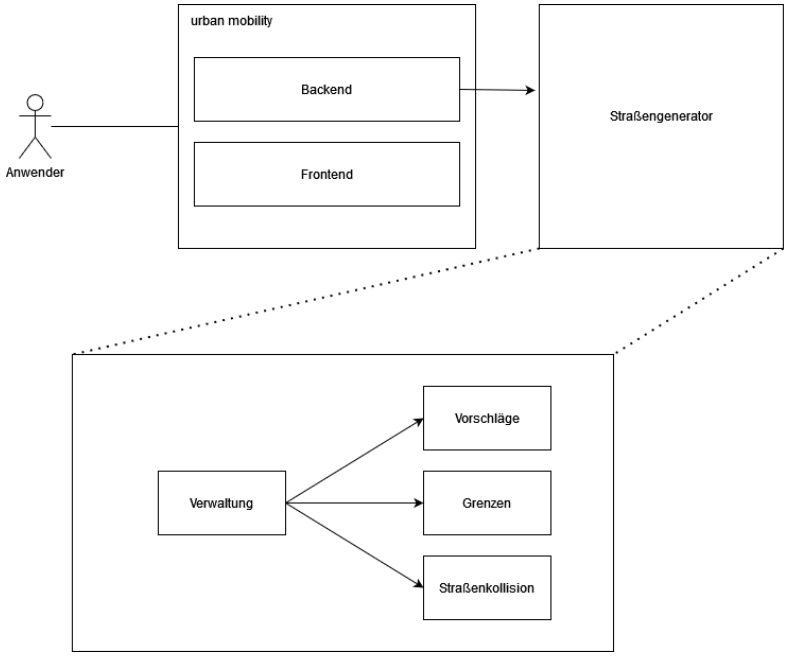

Abbildung 4.2: Bausteinsicht der Systemarchitektur

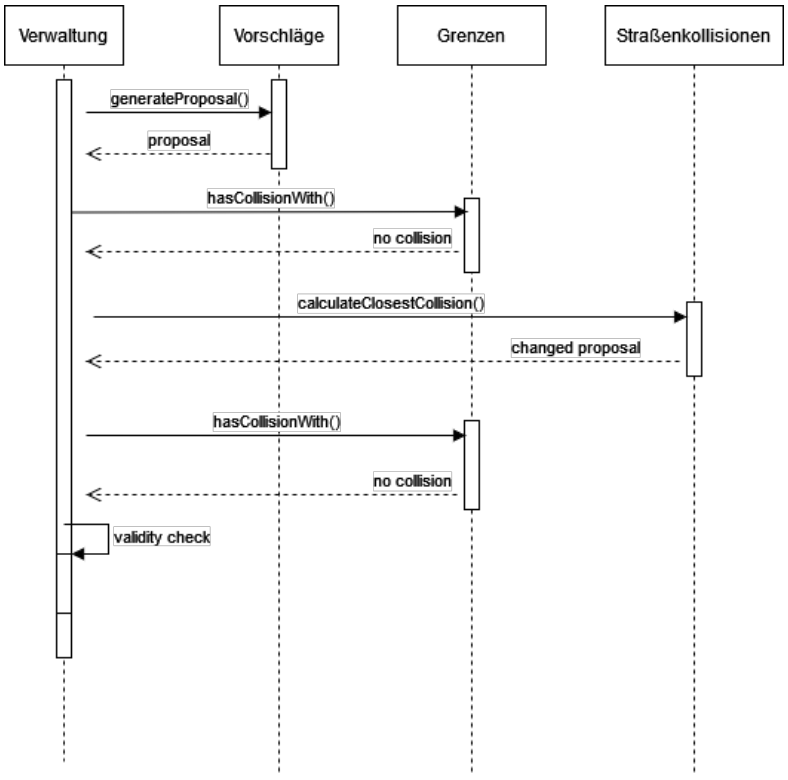

Abbildung 4.3: Laufzeitdiagramm der Komponenten

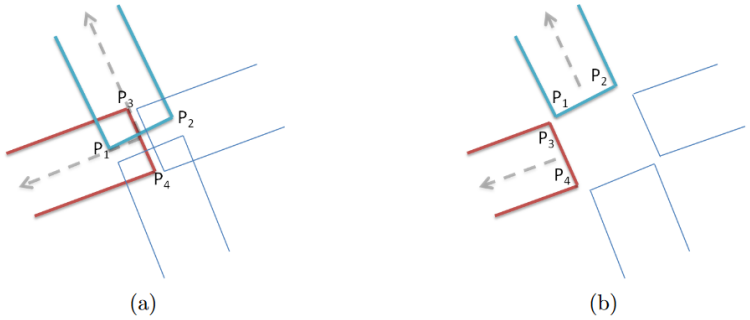

Abbildung 4.4: Vorgehen Einkürzung von Straßenenden an Kreuzungen, aus [8].

Darstellung der Straßen erfolgt nach dem im Kapitel 2.6 und in [8] dargestellten Vorgehen. Die gebildeten Kurven werden stückweise abgelaufen und an jedem Punkt anhand der Kurventangente in den orthogonalen Richtungen die Straßenbreiten abgesteckt.

Kreuzungen und Übergänge zwischen zwei Segmenten werden gleich behandelt. Kreuzungen werden so dimensioniert, dass sich keine Straßenteile mehr überlappen. Hierzu werden nach [8] die inzidenten Straßenenden gleichmäßig gekürzt bis sie sich nicht mehr überschneiden. Das Bild 4.4 (a) zeigt die Situation vor dem Zurückziehen der Straßenenden. In (b) wurden die Straßenenden in Pfeilrichtung eingekürzt und es entstehen Lücken zwischen den Straßenkörpern, die verbunden werden können.

Die Straßenenden der Kreuzung werden im Kreis abgelaufen. Aus der Kreuzung wird ein Dreiecksnetz gebildet. Die Eckpunkte eines Straßenendes bilden gemeinsam mit der Kreuzungsmitte jeweils ein Dreieck. Die Kreuzungsmitte ist der ursprüngliche Knotenpunkt der Kreuzung. Als Textur wird die Straßentextur (Bild 3.8) genutzt. Die Eckpunkte des Straßenendes zeigen dabei auf die unteren oder oberen Eckpunkte der Textur. Die Kreuzungsmitte zeigt auf die Mitte der Textur.

Die Lücken zwischen den Straßenenden werden mit Krümmungen gefüllt. Die direkte Verbindung der Straßenenden dienen als Hilfslinien und sind Fassadensegmente. Der Winkel der Straßenenden zueinander entscheidet, ob ein konvexer oder konkaver Bogen erzeugt wird. Der Verlauf wird durch die Funktion

$$f(x) = -(x - 0.5)^2 \cdot 4 + 1$$

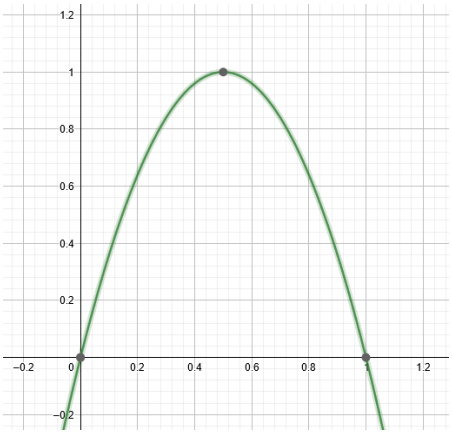

Abbildung 4.5: Verlauf der quadratischen Funktion, erstellt mit https://www.geogebra.org/calculator.

erzeugt. Das Bild der Funktion ist in Bild 4.5 dargestellt. Die Eingabe x ist der relative Punkt auf dem Fassadensegment. Diese Funktion gibt für $x \in [0; 1]$ einen Wert im Intervall von $[0; 1]$ zurück. Es wird ein höherer Wert zurückgegeben, je näher die Eingabe der Hälfte des Fassadensegments ist. Das Ergebnis muss mit einem Wert skaliert werden, der die maximale Wölbung beschreibt.

Das Fassadensegment wird in kurzen Abständen abgelaufen und für jeden Punkt ergibt sich ein Dreieck für das Dreiecksnetz der Kreuzung. Zwei aufeinanderfolgende Punkte der Außenseite der Krümmung zeigen für ein Dreieck auf die linke oder rechte Seite der Textur in Bild 3.8. Der Abstand zwischen den beiden Punkten wird auch auf die Referenzen in der Texturkarte gespiegelt. Der dritte Punkt des Dreiecks ist die Kreuzungsmitte. In der Textur zeigt dieser Punkt auf die Mitte.

Die Visualisierung von Kreuzungen und der Straßen ist bereits teilweise im Bestandsprojekt implementiert. Die zusätzlichen Funktionen werden dort ergänzt.

4.4.5 Sonstige Aufgaben

Straßenkollisionen werden mit dem *Snap*-Algorithmus aus [13] wie im Kapitel 2.2 beschrieben berechnet. Das Einschnappen mit anderen Straßen kann so durch einen Parameter gesteuert werden. Daneben werden zu nahe Straßenteile gleichmäßig über die gesamte Länge des Segmentes erkannt.

Als Vorschlagkomponente wird für den Prototyp zunächst nur ein zufallsgesteuerter Ansatz geplant. Entweder wird die Richtung eines Vorgängers im Straßenzug eingehalten oder eine zufällige Richtung für den Vorschlag generiert.

Für die Grenzenkomponente gibt es im Prototypen nur die Kartengrenzen zu prüfen. Die Kartengrenze wird als Polygon in seine Segmente aufgeteilt und eine Kollision durch Schnitt mit einem Eingabesegment überprüft.

5 Umsetzung

In diesem Kapitel werden Abweichungen vom Konzept und ausgewählte Details der Implementierung dargestellt. Als Übersicht zeigt Bild 5.1 die Funktionen des Frameworks in dem die Straßengenerierung eingebettet wird. Zudem wurde die Generierung von Kreuzungen ersetzt.

5.1 Verwendete Technologien

Das Projekt wird mit Java SE 15 umgesetzt[1]. Wobei die genutzten Funktionalitäten im Projekt höchstens die Java-Version 12 benötigen[2]. Zudem wird das Projekt mittels dem Framework *jMonkeyEngine*[3] Version 3.3.2 umgesetzt. Das Framework wird mit dem Gradle Build-Management-Automatisierungs-Tool eingebunden[4].

[1] https://www.oracle.com/java/technologies/javase/jdk15-archive-downloads.html , abgerufen am 30.11.2021
[2] https://en.wikipedia.org/wiki/Java_version_history#Java_12 , abgerufen am 30.11.2021
[3] https://jmonkeyengine.org/
[4] https://gradle.org/ , abgerufen am 30.11.2021

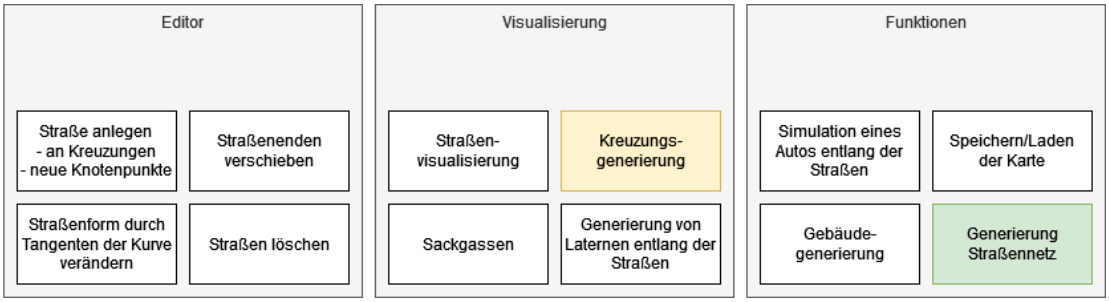

Abbildung 5.1: Übersicht des bestehenden Systems. Der Straßengenerator ist Thema dieser Arbeit. Die Kreuzungsdarstellung wurde überarbeitet.

5.2 Verwaltung

Die Verwaltung des Straßengenerators liegt im *road_ network_ manager* Paket. Insgesamt ergibt sich weitestgehend eine engere Koppelung zwischen den Komponenten. Durch die inhaltliche Nähe der Funktionen zueinander wurde auf ein Abstrahieren der Datentypen durch Datentransferobjekte, die vom urban_mobility Projekt kommen, verzichtet. Die Implementierung nutzt also die Straßen- und Kreuzungsdatentypen. Die Einbindung der Komponenten erfolgt durch Konstruktor Injektion. Dies soll einen späteren Wechsel zu anderen Strategien erleichtern.

Die Initialisierung eines ersten Straßenpunktes erfolgt in der Kartenmitte. Zur Zeit wird die Straßengenerierung nur mit dem Weltkartenmodell aufgerufen.

5.2.1 Straßenzüge

Straßenzüge ergeben sich implizit durch aufeinanderfolgende Straßen mit ähnlicher Richtung. Sie sind im Ergebnis nur durch die Visuelle Einheit erkennbar. Die Hermite-Kurven der Straßendatenstruktur werden nicht zu einem Spline verknüpft. Um visuell Straßenzüge anzudeuten werden Fortführungen eines Straßenzuges mit einem geringen Winkelunterschied zum Vorgänger zugelassen. Eine Fortführung wird daran erkannt, dass das letzte Segment erfolgreich gesetzt wurde. Der oberste Straßenansatzpunkt in der Warteschlange ist dann immer ein Folgepunkt der letzten Straße. Deren Richtung wird dann auch an die Vorschlagkomponente weitergegeben. Alternativ wird der Vorschlagkomponente eine zufällige Richtung mitgegeben. Zusätzlich zur ähnlichen Richtung werden die Tangenten der gelegten Straßen so angepasst, dass die Übergänge zwischen zwei Straßen geglättet und somit unscheinbar werden.

Zur Glättung der Übergänge wird die Tangente im Endpunkt der Vorgängerkurve und die Tangente im Anfangspunkt der Nachfolgekurve gleichgestellt. Als Tangente wird der Vektor vom Anfangspunkt des Vorgängers zum Endpunkt des Nachfolgers gewählt. Ein Parameter skaliert die Länge der Tangente, sodass die Weichheit mit der die Kurve den Kontrollpunkt durchläuft gesteuert werden kann. Dies folgt dem Prinzip der Cardinal Spline[5].

[5] https://de.wikipedia.org/wiki/Kubisch_Hermitescher_Spline#Cardinal_Spline , abgerufen am 25.11.2021

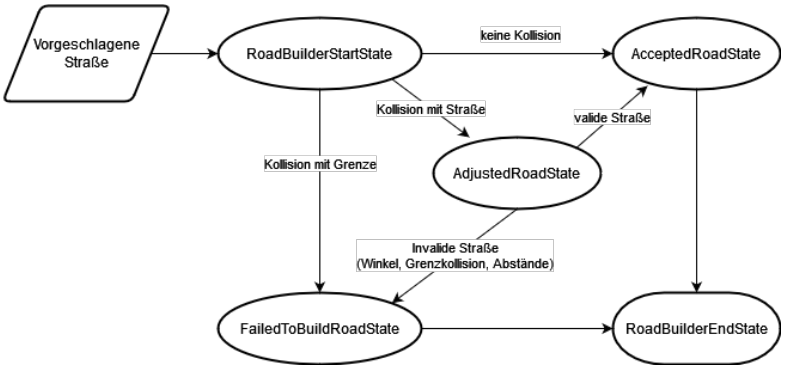

Abbildung 5.2: Zustandsautomat für die Vorschlagevaluation.

Die Tangenten von Straßen ohne Vorgänger richten sich nach dem Vektor vom End- zum Startpunkt. Somit ergibt sich zunächst eine Gerade.

5.2.2 Verhungern der Generierung

Ein Fortlauf eines Straßenzuges wird nicht versucht, wenn das vorhergehende Segment auf eine andere Straße oder Grenze stößt. Auf eine Verästlung besteht während der Bildung des Straßenzuges pro Segment nur eine Chance. Das Prinzip ähnelt so dem L-System aus [20]. Es besteht die Möglichkeit, dass zu einem frühen Zeitpunkt in der Generierung keine neuen Straßenansätze mehr in der Warteschlange gehalten werden und die Generierung deshalb vorzeitig stoppt. Auf der Karte finden sich dann nur wenige Straßen wieder. Als Lösung werden bei leerer Warteschlange solange Straßenansatzpunkte hinzugefügt bis wiederholt kein Segment erfolgreich gesetzt werden konnte. Das wiederholte Fehlschlagen ist dabei ein Indikator für eine Sättigung der Karte mit Straßen.

5.2.3 Prototypen Zustandsautomat

Der Ablauf von einem Vorschlag bis zur akzeptierten oder abgelehnten Straße hat mehrere Fallunterscheidungen. In einer *If-Else*-Verkettung ist die Lesbarkeit und Erweiterbarkeit des Ablaufes stark eingeschränkt. Als Lösung wurde hier ein Zustandsautomat implementiert. Die expliziten Zustände und deren Übergänge erleichtern die Lesbarkeit zumindest unter Zuhilfenahme der Dokumentation (Bild 5.2).

Der Zustand ist abgebildet in der *RoadBuilderContext*-Klasse. Im Start-Zustand werden Straßenkollisionen direkt behandelt. Der *AdjustedRoadState* überprüft die Validität

des geänderten Straßenteils, da dieser nunmehr wieder mit Grenzen kollidieren oder unerwünschte Winkel und Abstände zu anderen Kreuzungen haben könnte. Im *AcceptedRoadState* werden die Tangenten des einzubindenden Straßenteils angepasst.

5.3 Straßenvorschläge

Die implementierte Vorschlagkomponente richtet die neue Straße nach der mitgegebenen Richtung aus. Die vorgegebene Richtung wird durch einen kleinen Winkel zufällig gedreht, um zu gerade Straßenzüge zu vermeiden. Falls der vorgeschlagene Straßenteil im Startpunkt den Mindestwinkel gegenüber den anderen anliegenden Straßenteilen nicht einhalten kann, wird der Vorschlag um 5° rotiert. Die Rotation des Vorschlages wird wiederholt bis das vorgeschlagene Element passt oder 360° rotiert wurde.

5.4 Kollisionsbehandlung

Der Kollisionserkennung zwischen Straßen ist nach dem Schnappalgorithmus aus [13] implementiert. Der Schnappradius ist nach dem Algorithmus ein Wert relativ zur Länge des zu überprüfenden Segmentes. Zur Übersetzung eines absoluten Wertes in den relativen Wert wird der absolute Schnappradius durch die Segmentlänge geteilt. Da beim dritten Test die am Startpunkt anliegenden Segmente nicht ignoriert werden, kann das Vorschlagsegment in bestimmten Fällen am eigenen Startpunkt einrasten. Dies ist insbesondere dann der Fall, wenn der Schnappradius größer als die Länge des Segmentes ist. In diesem Fall ist die Neugenerierung von Straßen komplett blockiert. Deshalb wird der Schnappradius, sofern er die Länge des Segmentes übersteigt, auf 90% der Segmentlänge gekürzt. Invalide Segmentanpassungen durch das Einrasten bleiben aber weiterhin möglich. Die Validität des geändertes Segmentes wird von der Verwaltungskomponente abschließend überprüft.

Im Stand der Technik Kapitel wurde der Algorithmus bereits grob beschrieben. In der groben Beschreibung ist die Komplexität des Algorithmus nicht vollständig ersichtlich. Die konkrete Implementierung wird deshalb hier beschrieben.

Als Ergebnis liefert die Kollisionskomponente die Kollisionsposition, die Kollisionsart und das Element mit dem der Vorschlag kollidiert. Als Kollisionselement wird entweder das Straßensegment oder der Straßenknotenpunkt zurückgegeben. Demnach leitet sich

der Kollisionstyp ab - es ist entweder eine Knoten- oder Segment-Kollision. Die Eingabe besteht aus dem Graphen des bestehenden Straßennetzes, dem Vorschlagsegment und dem Startpunkt des Vorschlagsegmentes.

Zunächst wird eine *HashMap* gefüllt, in welcher jedes Element ein Knotenpunkt des Straßengraphen ist, das auf seine assoziierten r/s-Werte zum Vorschlagsegment zeigt. Die r/s-Werte werden nach Bild 2.9 ermittelt, wobei die Rechnung für die s-Werte eigentlich $s = ((a \cdot y - c \cdot y) \cdot (b \cdot x - a \cdot x) - (a \cdot x - c \cdot x) \cdot (b \cdot y - a \cdot y))/L^2$ lauten sollte. Da jeder Test nach Bild 2.8 den Startknoten ignoriert, wird dieser mit seinen Werten aus der *HashMap* entfernt.

Für den ersten Test werden aus dem Assoziativ-Speicher alle Elemente ausgewählt, die einen r-Wert zwischen 0 und 1+Schnappradius haben. Darüber hinaus müssen die ausgewählten Elemente im s-Wert absolut kleiner als der Schnappradius sein. Von den ausgewählten Elementen wird das Element mit dem niedrigsten r-Wert als mögliche Knotenkollision identifiziert.

Der zweite Test findet Segmentkollisionen. Dazu werden aus allen Segmenten des Straßengraphen diejenigen ignoriert, die am gleichen Startpunkt wie der Vorschlag anliegen. Aus den übrigen Segmenten werden jene ausgewählt, deren Endpunkte im s-Wert unterschiedliche Vorzeichen haben sowie einer der Endpunkte einen r-Wert im Intervall [0;1] hat oder die Endpunkte unterschiedliche Vorzeichen im r-Wert haben. Von den verbleibenden Segmenten wird ein möglicher Schnittpunkt mit dem Vorschlagsegment errechnet und der dem Startpunkt naheliegendste Schnittpunkt wird als mögliche Segmentkollision identifiziert.

Nach den ersten beiden Tests werden die gefundenen Kollisionen verglichen. Es wird die Kollision ausgewählt, die dem Startpunkt am nächsten liegt. Im Fall von Segmentkollisionen wird weiterhin geprüft, ob diese zu einer Knotenkollision umgewandelt werden kann. Hierzu wird auf dem Kollisionssegment der Endpunkt ausgewählt, der dem Schnittpunkt mit dem Vorschlagsegment am nächsten liegt. Zudem wird geprüft, ob die Distanz zwischen dem Schnittpunkt und dem ausgewählten Endpunkt kleiner als der Schnappradius ist. Das Vorschlagsegment soll dann mit dem jeweiligen Endpunkt einrasten anstatt auf dem Segment einen neuen Knotenpunkt zu bilden. Es wird also eine Knotenkollision zurückgegeben.

Sofern in den ersten Tests keine Kollision gefunden wurde, wird die Nähe von Segmenten zum Endpunkt des Vorschlagsegmentes überprüft. Dazu wird ein neuer Assoziativ-

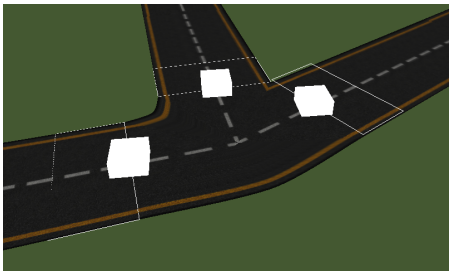

Abbildung 5.3: Hüllkörper der Straßenenden in weiß zur Erkennung von Überlappungen.

Speicher aufgebaut, der alle Segmente des Straßennetzes mit ihren r/s-Werten bezüglich des Endpunktes des Vorschlagsegmentes hält. Im Bild 2.9 besteht das jeweilige Straßensegment nun aus den Endpunkten a und b, während der Endpunkt des Vorschlagsegmentes c ist. Aus den Straßensegmenten werden diejenigen ausgewählt, zu denen der Vorschlagpunkt einen r-Wert im Intervall [-Schnappradius; 1+Schnappradius] hat und der absolute s-Wert kleiner als der Schnappradius ist. Das Segment mit dem kleinsten s-Wert ist das naheliegendste Einrasten mit dem Vorschlagendpunkt. Da es sich hier um eine Segmentkollision handelt, wird wieder die Möglichkeit einer Umwandlung in eine Knotenkollision überprüft.

5.5 Visualisierung von Kreuzungen

Im bestehenden Projekt waren Kreuzungen durch statische Texturen realisiert. Die möglichen Kreuzungstypen waren Übergänge mit zwei Straßen bis zu Kreuzungen mit vier anliegenden Straßen. Die Winkel der Straßen zueinander waren durch die statischen Texturen festgesetzt auf 90° bzw. 45°.

Für die neuen Kreuzungen müssen zunächst für alle Knotenpunkte die Positionen gefunden werden, an welchen sich die anliegenden Straßenenden nicht mehr überlappen. Als Referenz für die Position wird der Kurvenparameter der jeweiligen Straße genommen. Aus jedem Knotenpunkt wird ein das Straßenende repräsentierender Hüllkörper (Bild 5.3) berechnet. Initial liegen die Hüllkörper am Kurvenparameter 0 bzw. 1. Bei Feststellung einer Überscheidung wird der Kurvenparameter um einen kleinen Wert vom Straßenende weggeschoben. Die Hüllkörper werden neu berechnet und wieder auf Kollisionen geprüft. Es werden die Kurvenparameter von allen, am Knotenpunkt anliegenden, Straßenteilen gleichmäßig verschoben.

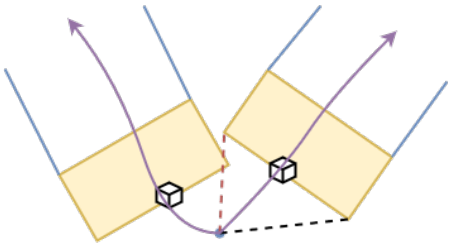

Abbildung 5.4: Hüllkörper der Straßenenden in Gelb nach Kollisionsbehandlung. Violett: Richtung der Straßenkurven; Gestrichelt: Ränder eines Dreiecks für die Texturierung der Kreuzung; Blau: angedeutete Straße

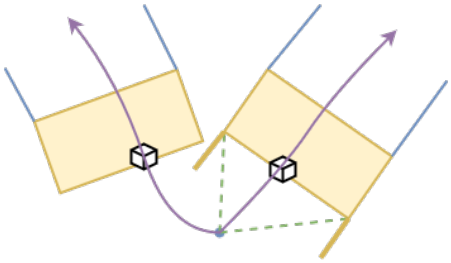

Abbildung 5.5: Erweiterung der Straßenenden-Hüllkörper.

Die alleinige Verschiebung nach Hüllkörperkollision reicht nicht aus. Im Ergebnis überschneiden sich die Straßenteile zwar nicht mehr, jedoch liegen sie in einigen Fällen noch so, dass die Texturierung deutlich erschwert wird. Das Bild 5.4 zeigt, dass ein Dreieck von einem Hüllkörper noch immer schneiden würde. Ursächlich ist, dass die Hüllkörper entlang der Straßenkurve verschoben werden. Eine Verschiebung des Kurvenparameters bewirkt keine gleichmäßige Überbrückung einer Entfernung. Daneben führen Krümmungen in der Kurve nach dem Verschieben zu unterschiedlichen Distanzen zum Kreuzungsmittelpunkt.

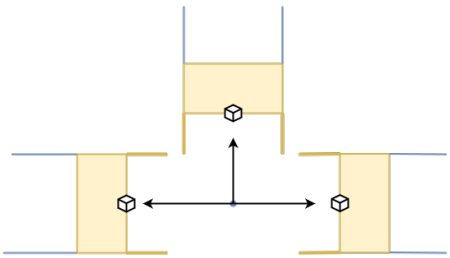

Abbildung 5.6: Hohe Distanzen zur Kreuzungsmitte durch Hüllkörpererweiterung.

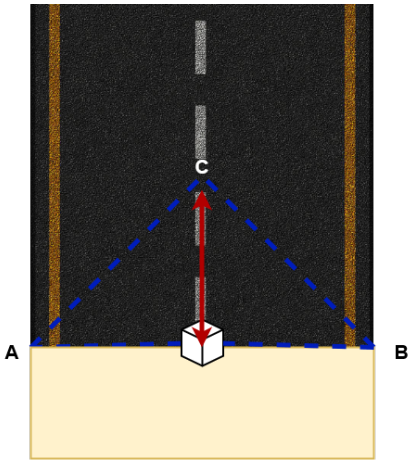

Abbildung 5.7: Bildung des Texturdreiecks für Straßenmündungen.

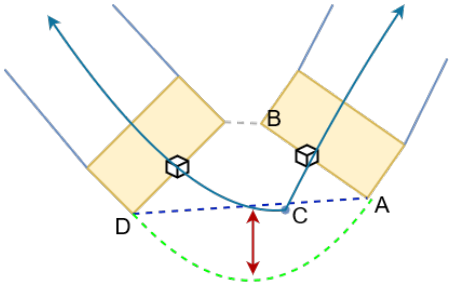

Abbildung 5.8: Bildung der Kurve zwischen Straßenenden mittels quadratischer Funktion.

Als Lösung werden die Hüllkörper durch jeweils ein Segment an den vorderen Eckpunkten in Richtung Kreuzungsmitte erweitert (Bild 5.5). Diese Erweiterung der Hüllkörper wird nicht mit der ersten Kollisionsprüfung erfasst, da dies zu unerwünscht großen Abständen zur Kreuzungsmitte selbst bei geraden T-Kreuzungen führt (Bild 5.6). Für jedes Straßenende wird daher mit der Hüllkörpererweiterung eine Kollision mit den anderen Straßenenden ohne die Erweiterung geprüft. Wenn es eine Kollision gibt, wird nur der von der Erweiterung getroffene Hüllkörper zurückgeschoben.

Nachdem die jeweiligen Positionen der anliegenden Straßenenden für die Kreuzungsbildung gefunden wurden, werden die Straßenenden gegen den Uhrzeigersinn abgelaufen. Die Bildung des Texturdreiecks für die Straßenöffnung zeigt Bild 5.7. Die seitlichen Punkte der Straßenmündung bilden dabei A und B. Der Punkt C ist in der Mitte der Textur, wobei der Abstand zu $\vec{AB}$ (roter Doppelpfeil) der reale Abstand zum ursprünglichen Knotenpunkt ist.

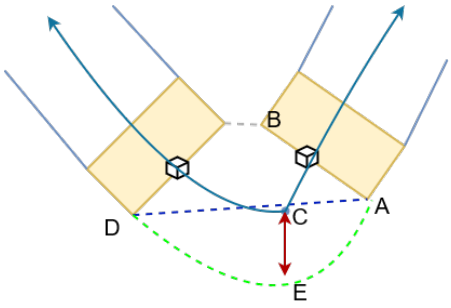

Abbildung 5.9: Bildung der Kurve zwischen Straßenenden mittels Hermite-Spline.

Die Verbindungen zwischen den Straßenmündungen müssen extra geformt werden. Der Ansatz aus [8] sieht hierzu eine quadratische Funktion und maximale Krümmungen in Abhängigkeit vom Winkel der Straßenteile zueinander vor. Das Bild 5.8 verdeutlicht die Situation. Die maximale Krümmung wird so gewählt, dass die Spurbreite stets zur Linienführung der Straßenkurve passt. Die Kreuzungsmitte C kann links oder rechts vom Segment $\vec{DA}$ liegen. Deshalb wird der Abstand zwischen $\vec{DA}$ und C negiert, falls C auf der linken Seite von $\vec{DA}$ liegt. Im Bild 5.8 ist der Abstand demnach positiv. Die maximale Kurvatur ergibt sich aus dem errechneten Abstand und einer Spurbreite der Straße.

Das Segment $\vec{DA}$ kann stückweise abgelaufen werden. Ein Punkt in der Fassaden-Kurve (grüne Linie) errechnet sich durch den prozentualen Fortschritt auf dem Segment $\vec{DA} \in [0;1]$. Der Fortschritt wird in die Kurvenfunktion $f(x) = -(x-0.5)^2 * 4 + 1$ eingesetzt und mit der maximalen Kurvatur skaliert. Problematisch hierbei ist, dass die maximale Auswölbung auf der Hälfte des Segmentes $\vec{DA}$ liegt (roter Doppelpfeil im Bild 5.8). Da die Kreuzungsmitte C nicht auf der Mitte des Segmentes liegt, ergeben sich unrealistische Straßenverengungen. Die Kurvatur kann durch die quadratische Funktion auch nicht sinnvoll verschoben werden, um die Auswölbung näher zur C zu bringen.

Als Lösung wurde die Fassadenkurve mittels eines interpolierenden Hermite-Spline implementiert. Die Wirkung zeigt Bild 5.9. Der Spline besteht aus zwei den Hermite-Kurven über $\vec{DE}$ und $\vec{EA}$. Die Starttangente von $\vec{DE}$ und die Endtangente von $\vec{EA}$ zeigen vom jeweiligen Kontrollpunkt aus in Richtung E, wobei die Endtangente von $\vec{EA}$ invertiert wird. Am Kontrollpunkt E zeigen die Tangenten in die Richtung des Vektors $\vec{DA}$. Der Spline hat somit einen ähnlichen Aufbau wie die Splines über die Straßensegmente.

Schließlich muss der Kontrollpunkt E berechnet werden. Die Orthogonale von $\vec{DA}$ gibt die Richtung vor. In diese Richtung wird von C aus eine Spurbreite entfernt der Kon-

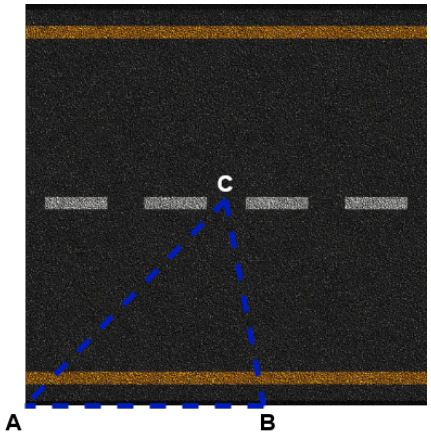

Abbildung 5.10: Bildung des Texturdreiecks für Kurvenpunkte zwischen Straßenenden.

trollpunkt E gesetzt: $\vec{E} = \vec{C} + (\text{orthogonal}(\frac{\vec{DA}}{|\vec{DA}|}) * \text{Spurbreite})$. Der gebildete Spline kann nunmehr stückweise abgelaufen werden, um die Punkte der Kurve zu errechnen. Jeder Punkt B auf dem Spline ergibt mit seinem Vorgänger A und der Kreuzungsmitte C ein Texturdreieck (Bild 5.10). Der reale Abstand zwischen A und B wird auch auf die Texturkoordinaten übertragen. Dagegen bleibt die Texturkoordinate von C statisch auf der Texturmitte.

Die Bildung von konkaven Krümmungen zwischen den Straßenenden muss abgewandelt werden, da sich die gebildeten Dreiecke sonst mit den Dreiecken der Straßenmündungen überlappen. Der mittige Kontrollpunkt des Spline ergibt sich aus der Mitte des Fassadensegmentes $\vec{F} = \frac{\vec{D}+\vec{A}}{2}$ und der Richtung von dort zur Kreuzungsmitte. Die Entfernung von der Kreuzungsmitte in die Richtung $\vec{FC}$ gibt die Krümmung durch den Kontrollpunkt des Spline an. Besonders wenn die Distanz der benachbarten Straßenenden sehr kurz ist, wirkt die so gewonnene Einbuchtung unregelmäßig tief. Deshalb wird die Kurvatur auf maximal ein Viertel der Distanz zwischen den Straßenmündungen begrenzt.

6 Evaluation

Nachdem im vorangegangenen Kapitel der Prototyp zur Straßengenerierung erstellt wurde, wird dieser im nachfolgenden Abschnitt detailliert analysiert. Dazu erfolgt eine nähere Betrachtung der Parameter und auch Herausforderungen bei der Erstellung des Prototyps sowie dessen Laufzeit werden näher beleuchtet.

6.1 Parameter des Straßennetzes

Der folgende Teil behandelt die möglichen Einstellparameter des generierten Straßennetzes und deren Auswirkung auf das Layout. Für alle Parameter gibt es Grenzen, außerhalb derer ein unrealistisches Ergebnis wahrscheinlich ist.

Die Länge eines Segmentes kann durch Minimal- und Maximallängen gesteuert werden. Im Falle von Einrastungen wird von der eingestellten Länge aber auch abgewichen. Die Unterscheidung von Minimal- und Maximallängen kann dabei zu mehr Variabilität im erzeugten Straßennetz führen. Kurze Segmente erzeugen Verwerfungen durch zu lange Tangenten. Bei langen Segmenten tritt das Problem hingegen nicht auf, obwohl die Tangenten in Abhängigkeit von der Segmentlänge gewählt werden. Bei kurzen Segmenten treten zudem Probleme auf, wenn es darum geht an Kreuzungen den Startpunkt des Segmentes an der Kreuzung festzulegen. Der Startpunkt muss bei kurzen Segmenten so weit verschoben werden, dass dieser nicht mehr auf dem Segment liegt. Zu lange Segmente können häufig auf einer kleineren Karte nicht mehr gesetzt werden.

Zudem kann beobachtet werden, dass der Schnappradius keinen Effekt hat, wenn er größer als die Segmentlänge selbst ist. Darüber hinaus sollte der Schnappradius nicht kleiner als die Breite einer Straße sein. Da die Kollisionen aufgrund der geraden Segmente geprüft werden, sollte der Schnappradius außerdem eine Größe über der Straßenbreite hinaus haben. Durch den Zusammenhang zwischen Straßenlänge und Schnappradius sollten diese Parameter nur zusammen skaliert werden. Als guter Wert eignet sich etwa ein

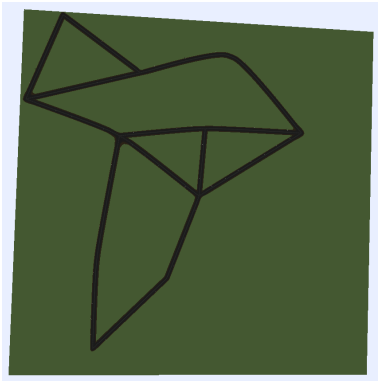

Abbildung 6.1: Längenskalierung 5

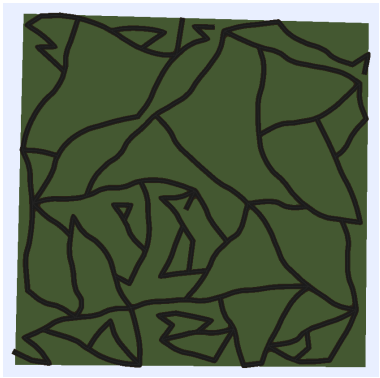

Abbildung 6.2: Längenskalierung 1

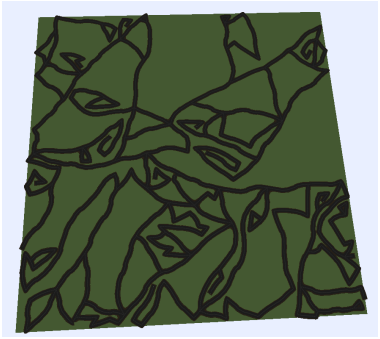

Abbildung 6.3: Längenskalierung 0.5

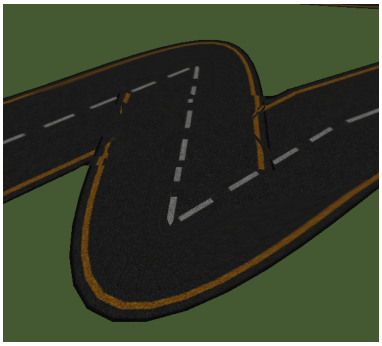

Abbildung 6.4: Längenskalierung 0.5 Artefakte durch zu kurze Segmente

Abbildung 6.5: Darstellung der Wirkung des Parameterkonglomerates Längenskalierung auf einer 500m x 500m Karte.

Schnappradius von fünfzig Prozent der Straßenlänge. Für die Straßenlänge eignen sich Werte zwischen dem siebenfachen und dem fünfunddreißigfachen der Straßenbreite. Eine große Differenz zwischen Minimal- und Maximallänge ist durch das Konglomerat mit dem Schnappradius nicht mehr angezeigt. Die Abbildung 6.5 stellt die Wirkung der Längenskalierung dar. Dabei zeigen die langen Segmente in Bild 6.1, dass nur noch wenige Straßen insgesamt gesetzt werden. Eine längere Längenskalierung führt auch zu geraden Strukturen, da die eigentliche Kurvatur von Straßen nur durch die Fortführung von Straßenzügen entsteht. Straßenzüge entstehen aufgrund der Kartengröße kaum noch. Eine zu kurze Längenskalierung führt durch den kleinen Schnappradius zu leichten Überlappungen und durch die kurzen Segmente auch zu Artefakten an Kreuzungen (Bild 6.4), weshalb die Längenskalierung zwischen eins und fünf liegen sollte.

Abbildung 6.6: Straßennetz generiert mit möglichst strengen Parametern. Abweichwinkel: 0°; Minimalwinkel: 85°

Ein minimaler Winkel an Kreuzungen beeinflusst, wie viele Straßen an einer Kreuzung anliegen könnten. Ein zu hoher minimaler Winkel bewirkt dabei, dass keine Verästlungen mehr möglich sind. Schon bei 90° werden Viererkreuzungen äußerst selten generiert. Als Höchstwert eignet sich daher ein Winkel von 85°. Zu kleine Winkel zwischen zwei nebeneinanderliegenden Straßen können für weitläufige Überlappungen an den Kreuzungsenden führen. Als Mindestwert sind 30° festgelegt.

Für Straßenzüge ist ein Abweichwinkel definiert, in welchem das nächste Element zur Richtung des Vorgängers abweichen kann. Geringe Abweichungen sorgen für lineare Straßenstrukturen, während hohe Abweichwinkel für organische Straßenzüge sorgen. Zu hohe Abweichwinkel führen zu starken Verwerfungen der zugrundeliegenden Straßenkurven durch die Tangenten. Als maximaler Wert wird daher 45° in beide Richtungen festgelegt. Das Minimum ist bei 0°.

Trotz der begradigenden Wirkung durch einen Abweichwinkel von 0° und einem Minimalwinkel von 90° ist es nicht möglich ein Schachbrettmuster zu generieren. Dies liegt daran, dass der Winkel in beim Verhungern und bei Verästelungen zufällig bzw. unabhängig von bestehenden Straßenteilen gewählt wird. Das Bild 6.6 zeigt, dass durch strenge Minimalwinkel lange Sackgassenstraßen entstehen. Eine möglichst offen generiertes Straßennetz (Bild 6.7) wirkt organisch, aber entwickelt teilweise unrealistische Kreuzungen. Auch die Wahl von moderaten Parametern in Bild 6.8 wirkt noch organisch.

Zugleich ist die Neugenerierung von Straßenansätzen bei Verhungern der Generierung steuerbar. Der Parameter regelt die zugelassene Anzahl von erfolglosen Straßenlegungsversuchen hintereinander. Andernfalls werden fortlaufend neue Straßenbauversuche an

Abbildung 6.7: Straßennetz generiert mit möglichst offenen Parametern. Abweichwinkel: 50°; Minimalwinkel: 30°

Abbildung 6.8: Straßennetz generiert mit möglichst mittleren Parametern. Abweichwinkel: 20°; Minimalwinkel: 55°

Abbildung 6.9: Straßennetz generiert mit 5 Wiederholungsversuchen.

Abbildung 6.10: Straßennetz generiert mit 40 Wiederholungsversuchen.

Sackgassenenden vorgenommen. Das Resultat sind unter bestimmten Umständen lange Sackgassen wie in Bild 6.6. Eine hohe Wiederholungszahl wirkt sich besonders auf die Laufzeit aus, da die erfolglosen Versuche auf Null gesetzt sind, sobald eine Straße platziert werden konnte. Die Bilder 6.9 und 6.10 zeigen den Unterschied zwischen den Füllgraden. Sobald eine Fläche ohne hineinragende Sackgassen von Straßen umschlossen wird, können diese Flächen nicht weiter mit neuen Straßen befüllt werden.

Schließlich ist es möglich den minimalen Abstand von Kreuzungen zu steuern. Eine hohe Distanz zwischen Kreuzungen sorgt für eine geringe Konnektivität im Straßennetz. Sie verhindert aber nicht, dass sich Straßenteile zu nahe kommen und so der Raum zwischen

Straßen zu klein zum Bebauen wird. Hierzu führt nur eine höhere Längenskalierung mit einem entsprechend höheren Schnappradius.

Nach der Generierung des Straßennetzes bleibt jedes Segment für sich mit seinen Tangenten editierbar. Gleichzeitig können weitere Straßenteile hinzugefügt oder gelöscht werden, was als positiv zu bewerten gilt, da es somit eine der gestellten Mindestanforderungen erfüllt.

Das Straßensystem ähnelt je nach gewählten Parametern dem eines realen Straßennetzes. Wobei es vereinzelt Straßenführungen gibt, die in der Realität gegebenenfalls nicht den Anforderungen an einen effizienten Verkehr entsprächen. So gibt es teils lange Wege ohne Verästlung oder große Freiflächen. Für noch effizientere Verbindungen und eine bessere Platznutzung innerhalb des Straßensystems sollten deswegen alternative Umsetzungsstrategien erwogen werden.

6.2 Bestehende Probleme

Mit der Umsetzung des Prototyps ergeben sich Punkte, die noch nicht optimal gelöst werden konnten. Dies betrifft unter anderen Problemstellen, für die sich theoretisch einfache Lösungswege finden lassen würden. Im begrenzten Rahmen dieser Arbeit wurde allerdings versucht vor allem komplexe Probleme bei der Straßengenerierung zu lösen. Gleichzeitig traten auch Probleme bei den konkaven Kreuzungsteilen auf, die trotz mehrerer Ansätze nicht abschließend gelöst werden konnten, aber in weiterführenden Arbeiten sicherlich zu beheben wären.

6.2.1 Kartengrenzen

Die Kollisionsprüfung mit der Kartengrenze wird zur Zeit nur mit den Segmenten des Straßenskeletts durchgeführt. Das hat zur Folge, dass Straßenelemente über den Kartenrand hinaus liegen. Als Lösung könnte die Kollisionserkennung des Straßensystems mit einem Radius um die Straße herum genutzt werden. Alternativ könnte um das zu testende Segment ein Rechteck gespannt werden, um alle Segmente des Rechtecks mit jedem Segment der Grenze auf Kollision zu prüfen.

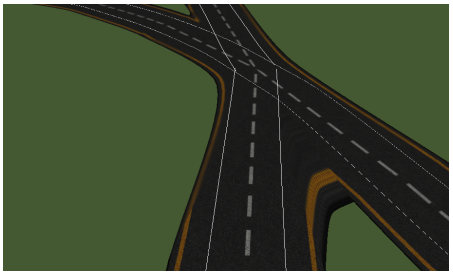

Abbildung 6.11: Kreuzung mit problematischer Visualisierung.

6.2.2 Kreuzungen

Bei näherer Betrachtung der Straßensysteme fällt auf, dass der konkave Teil einiger Kreuzungen nicht immer stimmig aussieht. Die Einbuchtung ist mit der Berechnung des konvexen Teils häufig zu tief. Die Heuristik, die Einbuchtung maximal ein viertel so tief zu machen, wie die Entfernung zwischen den Kreuzungsteilen ist, führt zu sehr flachen Einbuchtungen. Im Bild 6.11 sind beide Probleme zu sehen. Die weißen Linien deuten den Pfad von Fahrzeugen an. An den flachen Winkeln links und rechts ragt der Rand der Kreuzung ein wenig zu weit in die Straße hinein. Am spitzen Winkel unten greift hingegen die Heuristik für eine zu geringe Ausbeulung.

Bei der Kreuzung im oben gezeigten Bild 6.11 ist am unteren Teil der Gabelung eine Verzerrung des Gelbstreifens zu beobachten. Wünschenswert wäre an dieser Stelle eine Verbesserung der Texturen. Insbesondere wird hier eine Textur benötigt, die auf einer Seite den Gelbstreifen hat und auf der gegenüberliegenden Seite offen die graue Straße zeigt ohne Begrenzung oder Mittelstreifen. Die Textur kann entlang des Gelbstreifens wiederholt werden. In der Breite jedoch müsste die Textur weit genug sein, um die reale Entfernung zur Kreuzungsmitte abzubilden. Die gestreckten Texturen bleiben sonst sehr sichtbar.

Artefakte, wie sie in Bild 6.4 zu sehen sind, erzeugen zur Zeit lediglich eine Warnung im Programm. Sie können behandelt werden, indem bei Anlegen eines Segmentes an eine Kreuzung das Prinzip des Zurückschiebens der Kreuzung direkt ausgeführt wird und bei Überschreiten des Kurvenparameters von 0.5 das Segment zurückgewiesen wird. Dadurch wird die Prüfung auf Straßenendenkollision bei jeder neu angelegten Straße an eine Kreuzung vollführt. Dieses Vorgehen ist voraussichtlich durch die vielen Teilschritte laufzeitintensiv. Alternativ könnte nach Erstellung des Straßennetzes einmalig für jede Kreuzung der Algorithmus die Straßenenden finden und zu weit zurückgezogene Segmente

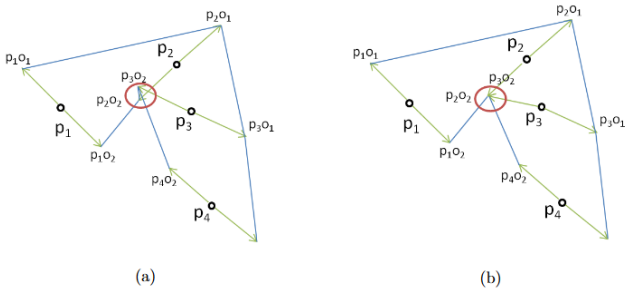

Abbildung 6.12: (a) Dreiecke einer Straße überlappen. (b) Behandlung durch Verschiebung des Eckpunktes auf die Position des vorhergehenden Eckpunktes. Aus [8]

an einer Kreuzung löschen. Der Nachteil wäre hier allerdings, dass dann eventuell kein zusammenhängender Graph mehr existieren würde und entsprechend behandelt werden müsste mittels Löschung oder Verbindung. Außerdem können vermehrt Sackgassen kurz vor Kreuzungen generiert werden, was in einem realen Straßennetz selten vorteilhaft für die Verkehrsführung wäre. In jedem Fall müsste die Berechnung der Kreuzungsenden im Backend erfolgen. Ein Lösungsansatz könnte darin bestehen, dass zwei Straßenübergänge nicht wie Kreuzungen behandelt werden. Als Spline würden die glatten Übergänge und die Kurve selbst an den Übergängen die Straßenform bestimmen.

6.2.3 Straßensegmente

Bei zu stark abweichenden Tangenten kann es dazu kommen, dass sich die Straßenkurven selbst schneiden. Um sich überlappende Dreiecke zu vermeiden hat [8] einen Weg entwickelt dieses Problem zu lösen. Der Eckpunkt des Dreiecks, der in einem vorhergehendem Dreieck liegen würde, wird auf den Eckpunkt des Vorgängers versetzt (siehe Bild 6.12). Dieses Problem wird derzeitig noch nicht behandelt und es bilden sich Artefakte.

6.2.4 Vorschlagsystem

Das aktuelle Vorschlagsystem beruht auf einer zufälligen Richtungswahl, wenn ein Straßenzug nicht fortgeführt wird. Das Prinzip des Straßenzuges in dieser Form berücksichtigen andere Vorschlagstrategien bisher nicht oder setzen sie stattdessen komplett in einer Iteration um. Teile des zufälligen Vorschlagsystems arbeiten zum jetzigen Stand in der Verwaltungskomponente und müssten in die Vorschlagkomponente eingearbeitet werden.

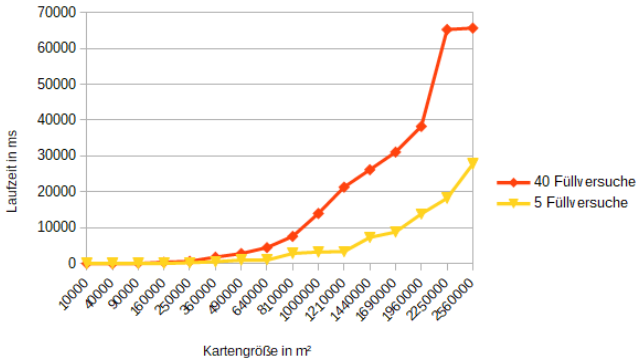

Abbildung 6.13: Laufzeitverhalten der Straßengenerierung ohne Visualisierung. X-Achse ist nicht linear eingeteilt.

Als Lösung müsste die Vorschlagkomponente die Kollisionsprüfungen selbst aufrufen, um etwa zwei Seeds in einem Seedsystem miteinander zu verbinden.

6.3 Laufzeit

Für die Laufzeit kann durch die Zufallskomponente der Verästlung oder des Mechanismus gegen Verhungern und zufälliger Richtungswahl keine konsistente Aussage getroffen werden, z.B. in Abhängigkeit zur Kartengröße. Der kleinste Mechanismus ohne Zufallskomponente ist die Prüfung eines Vorschlages selbst, der gegen jedes bestehende Straßen- und Kartensegment auf Schnitt geprüft wird. Die Laufzeit der Prüfung ist insofern linear zur Anzahl zu den bestehenden Segmenten. Bemerkbar macht sich das auch, da kurze Längenskalierungen eine höhere Laufzeit bedeuten als lange Längenskalierungen. Die Generierung ist auf kleineren Karten um ein vielfaches schneller als der händische Aufbau des Straßennetzes. Auf größeren Karten dauert die Generierung länger, vermutlich aufgrund der wiederholten Straßenlegungsversuche bei Fehlschlägen. In der Grafik 6.13 wird das Laufzeitverhalten mit unterschiedlichen Straßenlegungsversuchen deutlich. Der Sprung im vorletzten Datenpunkt zeigt das unvorhersehbare Verhalten der Zufallskomponenten. Die Laufzeitentwicklung ist vergleichbar. Im Bild 6.14 zeigt sich eine etwa lineare Laufzeitentwicklung zur Kartengröße.

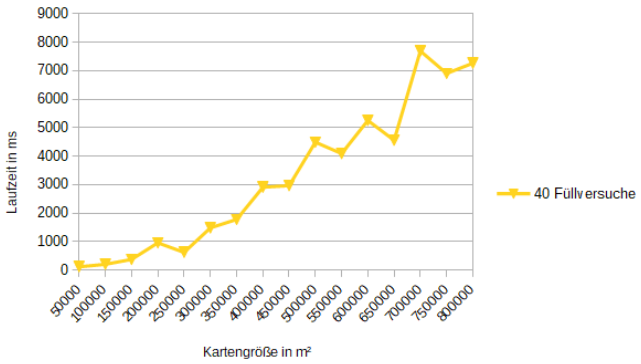

Abbildung 6.14: Laufzeitverhalten der Straßengenerierung ohne Visualisierung. X-Achse ist linear eingeteilt.

7 Schluss

Die übergeordnete Zielsetzung dieser Arbeit bestand darin einen Straßengenerator zu entwickeln, mit dessen Hilfe lebendige Städte automatisiert erstellt und simuliert werden können. Dazu wurde der Stand der Technik ermittelt und darin verschiedene Konzepte der prozeduralen Generierung von Straßensystemen vorgestellt. Es folgte die Definition von Anforderungen an den Prototypen eines Straßengenerators, auf deren Basis die Auswahl der dafür geeigneten Konzepte folgte. Als Ergebnis dieser Arbeit entstand so ein System, dass auf Grundlage von Parametern und Zufallskomponenten Straßennetze verschiedener Stile erzeugen kann. Der Ansatz des gierigen Wachstums erweist sich dabei als geeignet, um mit wenig Interaktion ein noch steuerbares Ergebnis zu erzeugen.

Die Umsetzung hat gezeigt, dass die Erstellung von dynamischen Kreuzungen sehr umfangreich ist. Zugleich waren die in dieser Arbeit gewählten Techniken, die bereits bei ähnlichen Projekten Anwendung fanden, nicht ohne weiteres auf den eignen Prototypen übertragbar, was zu Herausforderungen innerhalb der Umsetzung führte. Der Schnappalgorithmus stellt sich durch seine Koppelung zur Segmentlänge als weniger flexibel heraus als eingangs erhofft. Zu beachten ist, dass Splines und Kurven bei abgerundeten Straßenverläufen und Kreuzungen ein besonders hilfreiches Mittel sind.

7.1 Ausblick

Der in dieser Arbeit entwickelte Generator kann für die automatisierte Generierung von Straßennetzen genutzt werden. Als weitere Funktion empfiehlt sich das Auffinden von Bauflächen zwischen den Straßen. Auf den gefundenen Flächen können dann Gebäude generiert werden, damit das Gesamtbild dem einer realen Stadt ähnelt. Hilfreich dafür wäre zudem die Anpassung der Straßen auf verschiedene Spurzahlen, die Einführung weiterer Straßenlayoutstrategien sowie eine Anpassung, mit deren Hilfe auf verschiedenen Höhen Straßen generiert werden können für Tunnel und Brücken. Zusätzlich könnten

Straßen durch Parks oder Flüsse geleitet werden, für ein noch realistischeres Erlebnis. Abschließend könnte eine Editierfunktion eingeführt werden, mit der nur lokale Straßennetzteile (im Gegensatz zum gesamten Straßennetz) neu generiert werden können, falls diese Ausschnitte dem Anwender nicht gefallen.

Literaturverzeichnis

[1] Burch, Michael ; Wallner, Günter ; Arends, Sven T. ; Beri, Puneet: Procedural City Modeling for AR Applications. In: *2020 24th International Conference Information Visualisation (IV)* IEEE (Veranst.), 2020, S. 581–586

[2] Campos, Carlos ; Leitão, Joã. M. ; Pereira, Joao P. ; Ribas, António ; Coelho, António F.: Procedural generation of topologic road networks for driving simulation. In: *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)* IEEE (Veranst.), 2015, S. 1–6

[3] Catmull, Edwin ; Clark, James: Recursively generated B-spline surfaces on arbitrary topological meshes. In: *Computer-aided design* 10 (1978), Nr. 6, S. 350–355

[4] Chen, Guoning ; Esch, Gregory ; Wonka, Peter ; Müller, Pascal ; Zhang, Eugene: Interactive procedural street modeling. In: *ACM SIGGRAPH 2008 papers*. 2008, S. 1–10

[5] Eberly, David: *The minimal cycle basis for a planar graph*. 2005

[6] Foley, James D. ; Van, Foley D. ; Van Dam, Andries ; Feiner, Steven K. ; Hughes, John F. ; Hughes, J: *Computer graphics: principles and practice*. Bd. 12110. Addison-Wesley Professional, 1996

[7] Ford, Lester R. ; Fulkerson, Delbert R.: *Flows in networks*. Princeton university press, 2015

[8] Freiknecht, Jonas: Procedural content generation for games. (2021)

[9] Freiknecht, Jonas ; Effelsberg, Wolfgang: A Survey on the Procedural Generation of Virtual Worlds. In: *Multimodal Technologies and Interaction* 1 (2017), Nr. 4. – URL https://www.mdpi.com/2414-4088/1/4/27. – ISSN 2414-4088

[10] Groenewegen, Saskia ; Smelik, Ruben M. ; Kraker, Klaas J. de ; Bidarra, Rafael: Procedural City Layout Generation Based on Urban Land Use Models. In: *Eurographics (Short Papers)*, 2009, S. 45–48

[11] Hartmann, Stefan ; Weinmann, Michael ; Wessel, Raoul ; Klein, Reinhard: Streetgan: Towards road network synthesis with generative adversarial networks. (2017)

[12] Kelly, George ; McCabe, Hugh: A survey of procedural techniques for city generation. In: *ITB Journal* 14 (2006), Nr. 3, S. 342–351

[13] Kelly, George ; McCabe, Hugh: Citygen: An interactive system for procedural city generation. In: *Fifth International Conference on Game Design and Technology*, 2007, S. 8–16

[14] Lechner, Tom ; Watson, Benjamin ; Wilensky, Uri ; Tisue, Seth ; Felsen, Martin ; Moddrell, Andy ; Ren, Pin ; Brozefsky, Craig: Procedural modeling of urban land use / North Carolina State University. Dept. of Computer Science. 2007. – Forschungsbericht

[15] Lipp, Markus ; Scherzer, Daniel ; Wonka, Peter ; Wimmer, Michael: Interactive modeling of city layouts using layers of procedural content. In: *Computer Graphics Forum* Bd. 30 Wiley Online Library (Veranst.), 2011, S. 345–354

[16] Mei, Yandong: Procedural Generation of Urban Landscape with Extended L-System for Virtual Worlds. In: *Bulletin of Hosei University Graduate School. Graduate School of Informatics* (2017), Nr. 13, S. 1–6

[17] Mizdal, Tiago B. ; Pozzer, Cesar T.: Procedural content generation of villages and road system on arbitrary terrains. In: *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)* IEEE (Veranst.), 2018, S. 205–2056

[18] Nischwitz, Alfred ; Fischer, Max ; Haberäcker, Peter ; Socher, Gudrun: *Computergrafik und Bildverarbeitung: Band II: Bildverarbeitung.* Springer-Verlag, 2012

[19] Nishida, Gen ; Garcia-Dorado, Ignacio ; Aliaga, Daniel G.: Example-driven procedural urban roads. In: *Computer Graphics Forum* Bd. 35 Wiley Online Library (Veranst.), 2016, S. 5–17

[20] Parish, Yoav I. H. ; Müller, Pascal: Procedural Modeling of Cities. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA : Association for Computing Machinery, 2001 (SIGGRAPH '01), S. 301–308. – URL https://doi.org/10.1145/383259.383292. – ISBN 158113374X

[21] Perlin, Ken: An image synthesizer. In: *ACM Siggraph Computer Graphics* 19 (1985), Nr. 3, S. 287–296

[22] Prusinkiewicz, Przemyslaw ; Lindenmayer, Aristid: *The algorithmic beauty of plants*. Springer Science & Business Media, 2012

[23] Salomon, David: *Curves and surfaces for computer graphics*. Springer Science & Business Media, 2007

[24] Smelik, Ruben M. ; Tutenel, Tim ; Bidarra, Rafael ; Benes, Bedrich: A survey on procedural modelling for virtual worlds. In: *Computer Graphics Forum* Bd. 33 Wiley Online Library (Veranst.), 2014, S. 31–50

[25] Song, Asiiah ; Whitehead, Jim: TownSim: Agent-based city evolution for naturalistic road network generation. In: *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 2019, S. 1–9

[26] Teng, Edward ; Bidarra, Rafael: A semantic approach to patch-based procedural generation of urban road networks. In: *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 2017, S. 1–10

[27] Teoh, Soon T.: Algorithms for the Automatic Generation of Urban Streets and Buildings. In: *CGVR* Citeseer (Veranst.), 2008, S. 122–128

[28] Weber, Basil ; Müller, Pascal ; Wonka, Peter ; Gross, Markus: Interactive geometric simulation of 4d cities. In: *Computer Graphics Forum* Bd. 28 Wiley Online Library (Veranst.), 2009, S. 481–492

[29] Wilkie, David ; Sewall, Jason ; Lin, Ming C.: Transforming GIS data into functional road models for large-scale traffic simulation. In: *IEEE transactions on visualization and computer graphics* 18 (2011), Nr. 6, S. 890–901

[30] Williams, Benjamin ; Headleand, Christopher J.: A time-line approach for the generation of simulated settlements. In: *2017 International Conference on Cyberworlds (CW)* IEEE (Veranst.), 2017, S. 134–141

[31] Yang, Yong-Liang ; Wang, Jun ; Vouga, Etienne ; Wonka, Peter: Urban pattern: Layout design by hierarchical domain splitting. In: *ACM Transactions on Graphics (TOG)* 32 (2013), Nr. 6, S. 1–12

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original