Technische Universität München
Ingenieurfakultät Bau Geo Umwelt
Lehrstuhl für Geoinformatik
Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe

Automatisierte Generierung eines Stadtmodells für Straßenbahnsimulatoren

Julia Offer

Masterarbeit

Bearbeitung:	21.03.2016 - 20.09.2016
Studiengang:	Geodäsie und Geoinformation (Master)
Betreuer:	Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe (Technische Universität München)
	Caroline Marx, M.Sc. (Technische Universität München)
	Klaus R. Müller, Dipl.-Phys. (Müller Systemtechnik GmbH)

2016

Selbstständigkeitserklärung

Erklärung gemäß § 18 Absatz 9 APSO der Technischen Universität München:

„Ich versichere, dass ich diese Masterarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.“

München, den 20.09.2016

_____

Julia Offer

Kurzfassung

Die Verwendung von Trainingssimulatoren ist heutzutage in vielen Branchen gang und gäbe. Diese haben den Vorteil, dass beliebige Szenarien - insbesondere solche, deren Darstellung in der Realität mit hohem Aufwand verbunden ist - jederzeit erzeugt und trainiert werden können. Auch Straßenbahnführer, sei es zu Ausbildungs-, Weiterbildungs- oder Prüfungszwecken, werden mithilfe von Trainingssimulatoren geschult. Die Realitätsnähe des 3D-Modells spielt eine entscheidende Rolle für die Effektivität des Trainings. So ist eine Fahrt besonders lehrreich, wenn die Gegebenheiten der Realität so getreu wie möglich abgebildet werden, um z.B. Gefahrenstellen identifizieren zu können. Die Firma Müller Systemtechnik GmbH hat einen solchen Simulationstrainer für Straßenbahnführer entwickelt. Die Erzeugung des 3D-Modells erfolgt derzeit im Wesentlichen manuell.
Gegenstand dieser Masterarbeit ist es, die vorwiegend manuelle Generierung des 3D-Modells für einen Straßenbahnsimulator mithilfe prozeduraler Modellierung soweit wie möglich zu automatisieren und zu untersuchen, inwieweit eine Automatisierung sinnvoll ist. Als Datengrundlage dienen im Rahmen dieser Arbeit OpenSteetMap-Rohdaten. Die OSM-Daten werden mithilfe der Software FME aufbereitet. Zusätzlich erfolgt in FME eine Zerlegung der Schienentrasse in folgende Bauelemente: normale Schiene, Bahnübergang und Fußbahnübergang, Weiche und Gleiskreuzung sowie Bahnsteig und Bahnsteigpolygon. Im Anschluss werden in der Software CityEngine Modellierungskonzepte in Form von Regeln erstellt, durch deren Anwendung das 3D-Modell entsteht. Abschließend erfolgt ein Vergleich der in dieser Arbeit entwickelten Methode und dem daraus resultierenden 3D-Modell mit der Methodik der Müller Systemtechnik GmbH und dem manuell erstellten 3D-Modell.

Abstract

The use of training simulators is common practice in many industries. Simulators have the advantage that any scenario can be created and trained at any time, especially those which cannot be easily modeled. For example, tram drivers are trained with simulators, whether in use of education, qualification, or exam. The realistic representation of the 3D model is crucial for the quality of the training. A ride is especially instructive, if the simulation is as realistic as possible, e.g., to identify danger zones. The company Müller Systemtechnik GmbH developed such a training simulator for tram drivers. Currently, the generation of the 3D model is primarily conducted manually.
This master thesis aims to develop and evaluate an automated 3D model generation system for a tram simulator by means of procedural modeling. For this work, OpenStreetMap serves as a data source. The OSM data is processed with the software FME. Furthermore, the rail track is split into components, consisting of normal rail tracks, level crossings, switches, railway crossings, platforms, and platform polygons. Following the modelling step, concepts are created in the form of rules in the software CityEngine. By the use of the rules the 3D model is generated. Finally, a comparison is drawn between the resulting 3D model produced by the developed method of this work, the method of Müller Systemtechnik GmbH, and the manually created 3D model.

Inhaltsverzeichnis

1 Motivation

1.1 Zielsetzung

Sogenannte Trainingssimulatoren sind heutzutage in vielen Branchen gang und gäbe. So absolvieren angehende Piloten eine Vielzahl ihrer Flugstunden in Flugsimulatoren, das gleiche gilt für Lokführer. Die heutige Technik ermöglicht eine täuschend echte Simulation der Realität. Doch dienen die Simulatoren nicht nur zur Ausbildung, sondern auch zur Weiterbildung und dem regelmäßigen Training. Ein besonderer Vorteil ist die Möglichkeit, gewünschte Situationen auf Knopfdruck zu erzeugen. Szenarien mit schlechten Sichtverhältnissen aufgrund von starkem Regen oder Schneefall können in einem Simulator mit Leichtigkeit trainiert werden. Der Einsatzbereich von Simulatoren wächst stetig, auch Straßenbahnführer werden mit diesen trainiert.
Die Müller Systemtechnik GmbH hat einen Simulationstrainer für Lok- und Triebwagenführer sowie für Straßenbahnfahrer entwickelt. Mithilfe dieser Simulatoren können Lernende bestimmte Strecken abfahren und so die in ihrem Berufsleben alltäglichen Szenarien üben und verinnerlichen. Gegenstand dieser Arbeit ist der Simulationstrainer für Straßenbahnfahrer, welcher für drei verschiedene Einsatzbereiche konzipiert ist. Einerseits für Fahrten gemeinsam mit dem Ausbilder, um Szenarien zu trainieren, welche in der Realität nicht oder nur schwer nachzustellen sind. Weiterhin dient der Simulator dazu, dem Lernenden die Möglichkeit zu geben, zusätzliche Übungsfahrten ohne Ausbilder durchzuführen. Den dritten Einsatzbereich bilden sogenannte Prüfungsfahrten, mit deren Hilfe der Ausbilder die Kenntnisse und Fähigkeiten des Lernenden anhand der protokollierten Handlungen und gefahrenen Geschwindigkeiten bewerten kann. Beim Abfahren einer simulierten Strecke muss der Lernende Verschiedenes beachten und die jeweils erforderlichen Handlungen ausführen. Wichtig sind hier unter anderem Straßenbahnsignale, Verkehrsampeln oder Höchstgeschwindigkeiten. Die gleiche Aufmerksamkeit muss den übrigen Straßenverkehrsteilnehmern, wie Autofahrern, Fahrradfahrern oder Fußgängern zuteilwerden. Ein Simulator muss all diese Aspekte in entsprechendem Maß berücksichtigen. Zudem ist es erforderlich, dass die Umgebung in der Nähe der Straßenbahnschienen einen möglichst hohen Wiedererkennungswert für den Straßenbahnführer aufweist. Hierbei ist es von Bedeutung, umstehende Gebäude oder sogenannte Landmarks, Gebäude mit einem hohen Wiedererkennungswert wie z.B. Kirchen oder andere markante Bauwerke, so realitätsgetreu wie möglich darzustellen [Müller, 2015b].
Derzeit erfolgt die Generierung des computergestützten dreidimensionalen (3D) Modells im Wesentlichen manuell. Eine automatisierte Generierung ist angestrebt. Diese würde einige Vorteile, beispielsweise die Verringerung des Arbeitsaufwandes oder die Möglichkeit schneller Anpassungen, mit sich

bringen. In dieser Arbeit wird untersucht, inwieweit die Möglichkeit besteht, mittels prozeduraler Modellierung eine automatisierte Generierung des 3D-Modells zu erreichen.

1.2 Problemstellung

Die 3D-Modelle für den Straßenbahnsimulator der Firma Müller Systemtechnik GmbH werden weitgehend manuell erzeugt. Diese Arbeit untersucht, inwieweit eine Automatisierung der Generierung des 3D-Modells möglich und sinnvoll ist. Am Ende der Arbeit werden die im Rahmen dieser Arbeit gewonnenen Ergebnisse sowie die zugrundeliegende Methodik mit der bisherigen Generierung des Modells verglichen. Die Untersuchungen finden am Beispiel einer Teststrecke statt, die in dieser Arbeit durch die Tramlinie 5 zwischen den Haltestellen *Auestadion* und *Großenritte* der Stadt Kassel repräsentiert wird. Für diese Strecke existiert bereits ein 3D-Modell der Firma Müller Systemtechnik GmbH, somit besteht eine gute Basis für den Vergleich beider Methoden.
Ein digitales Geländemodell (DGM), welches in der Regel wichtiger Bestandteil von 3D-Stadtmodellen ist, wird in dieser Arbeit außer Acht gelassen, weil auch in dem bisherigen 3D-Modell für den Straßenbahnsimulator auf ein DGM verzichtet wurde und damit dementsprechend bessere Vergleichsmöglichkeiten geschaffen werden. Die 3D-Modelle für den Simulator, wie die Straßenbahntrasse, die umgebenden Straßen oder die Gebäude, werden auf einer flachen Ebene platziert.
Das Training mit einem Straßenbahnsimulator ist dann besonders sinnvoll und lehrreich, wenn dieser die Gegebenheiten der Realität so getreu wie möglich wiedergibt. Hierfür müssen zunächst die Elemente bestimmt werden, die für eine realistische Wiedergabe der Realität in einem Modell zwingend notwendig sind. Die für einen Straßenbahnsimulator wichtigen Faktoren werden im Folgenden aufgeführt:

- Straßenbahntrasse mit
  - Skelettlinie(n)
  - korrekter Spurzahl
  - korrektem Spurverlauf
  - Beschaffenheit (z.B. Holz-/Betonschwellen, Schotter-/Asphaltunterlage)
- Umgebende Straßen mit
  - korrekter Zahl der Spuren
  - Spurbreite
  - erlaubter Richtung der Befahrung
    - Einbahnstraße in Fahrt-/Gegenrichtung
    - Links-/Rechtsabbiegerspur, Geradeausspur

    - Parkspur
- Fußgänger und Radwege (mit erlaubter Befahrungsrichtung)
- Kreuzungen zwischen allen Verkehrswegen und der Straßenbahntrasse
- Bahnübergänge
- Unter- und Überführungen
- Straßenbahnsignale
- Verkehrsampeln
- Gebäude, wobei die markanten Gebäude der Realität entsprechen sollen
- Vegetation
- Strommasten für die Oberleitungen

[Müller, 2015a]

Als Datengrundlage dienen OpenStreetMap (OSM)-Daten. Um die OSM-Daten für eine 3D-Modellierung verwenden zu können, müssen diese zuvor aufbereitet werden. Die Aufbereitung erfolgt mithilfe der Software Feature Manipulation Engine (FME). Um die Schienentrasse erfolgreich modellieren zu können, soll diese in Bauelemente zerlegt werden. Hierfür gilt es Lösungskonzepte zu entwickeln. Die anschließende Generierung des 3D-Modells erfolgt mithilfe der Software CityEngine des Environmental Systems Research Institute (ESRI Inc.) mittels prozeduraler Modellierung. Für die prozedurale Modellierung in der CityEngine müssen Modellierungskonzepte in Form von Regeln erarbeitet werden, wobei die 3D-Modelle möglichst realitätsnah sein sollen.

1.3 Methode der Müller Systemtechnik GmbH

Dieses Kapitel beschäftigt sich mit der Methode der Müller Systemtechnik GmbH zur Erstellung eines 3D-Modells für einen Straßenbahnsimulator. Die Firma Müller Systemtechnik GmbH erstellt das computergestützte 3D-Modell semi-manuell [Müller, 2015a]. Nachfolgend wird auf die Methodik der Müller Systemtechnik GmbH näher eingegangen. Es werden die Objektarten, die Datengrundlage, die verwendeten Programme und der zeitliche Rahmen erläutert.
Die im 3D-Modell existierenden Objektarten entsprechen den in Kapitel 1.2 genannten Elementen, die ein 3D-Modell eines Straßenbahnsimulators beinhalten soll.
Die Tabelle 1.1 zeigt, wie die Erfassung der Rohdaten erfolgt. Im Folgenden wird nur auf ausgewählte Tabellenpunkte eingegangen.
Für die Erzeugung der Trasse dienen Orthophotos als Grundlage, die eine Auflösung von bis zu 20 cm besitzen. Über der Grundlage der Orthophotos werden Skelettlinien erzeugt, welche die Mittelachse der Straßenbahntrasse oder der Straße repräsentieren. Die Erzeugung der Skelettlinien

erfolgt durch das Zeichnen eines Splines. Falls die Straßen parallel zur Straßenbahntrasse verlaufen, ist die Erzeugung einer Skelettlinie ausreichend, andernfalls werden für die Straßen separate Skelettlinien erstellt. Für den Fall, dass Teilbereiche eines Orthophotos aufgrund von Verdeckungen, beispielsweise durch Bäume, unbrauchbar sind, wird auf eine topografische Karte zurückgegriffen. Zusätzlich werden die zweidimensionalen (2D) Profile der Straßenbahntrasse bzw. der Straßen erzeugt. Dies erfolgt mit einer 3D-Modellierungssoftware, z.B. mit dem Creator der Firma Presagis.

Objekt	**Quelle**
Trasse	Orthophoto + topografische Karte
Straßen samt Kreuzungen	Orthophoto + topografische Karte + Stadtplan
Beschaffenheit	Begehung
zulaufende und abgehende Gleise	Stadtplan, Begehung
Fußwege und Trampelpfade	zusammen mit den Straßen + Begehung
Straßenbahnsignale	Begehung oder Angaben des Auftraggebers
Verkehrsampeln	Begehung oder Angaben des Auftraggebers
Art der Kreuzungen und Bahnübergänge (Schranken, Ampeln etc.)	Begehung oder Angaben des Auftraggebers
Bordstein zur Trasse	Begehung oder Angaben des Auftraggebers
Gebäudegeometrien	3D-Stadtmodell oder Begehung
Weiter entfernte (auch außerhalb der Stadt liegende) Gebäude	Orthophoto + Begehung
Bäume, Hecken und wesentliche Sträucher	Orthophoto + Begehung
Strommasten der Oberleitung	Begehung oder Angaben des Auftraggebers
Gebäudefassaden	Begehung

Tabelle 1.1 – Erfassung der Rohdaten [Müller, 2015a]

Anschließend werden die 2D-Profile entlang der Splines extrudiert, wodurch texturierte 3D-Geometrien entstehen. Hierfür setzt die Müller Systemtechnik GmbH eigene Software ein. Die Texturen sind generisch und zeigen beispielsweise die Oberfläche der Schienen- oder Straßentrasse.
Wie bereits erwähnt, wird bei dem 3D-Modell für die Tramlinie 5 in Kassel auf ein DGM verzichtet. Daher wird in diesem Kapitel eine Beschreibung der Methode mit DGM außer Acht gelassen.
Die Gebäudegeometrien liefert entweder ein 3D-Stadtmodell; alternativ werden die Gebäudegeometrien auf Basis von Orthophotos und terrestrischen Photos manuell erzeugt. Die Fassaden von markanten Gebäuden werden mit manuell erzeugten Fassadenfotos texturiert. Wenn die Gebäude keinen besonderen Wiedererkennungswert aufweisen, erhalten sie generische Fassadentexturen.

Die Texturierung erfolgt wiederum mit einer 3D-Modellierungssoftware.
Die Vegetation sowie die Straßenbahnsignale und Verkehrsampeln werden durch Modelle dargestellt, die von der Müller Systemtechnik GmbH erstellt werden.
Der zeitliche Rahmen für die Erstellung des zu einer Strecke von 10 km gehörenden 3D-Modells liegt bei drei bis neun Monaten [Müller, 2016].
Die Abbildung 1.1 zeigt einige beispielhafte Bilder aus dem Simulator der Müller Systemtechnik GmbH.

Abbildung 1.1 – Bilder aus dem Simulator der Müller Systemtechnik GmbH [Müller, 2015a]

1.4 Aufbau der Arbeit

Zu Beginn wird ein kurzer Überblick über verwandte Arbeiten und Simulatoren gegeben.
Danach werden theoretische Grundlagen genannt und erläutert. So werden die OSM-Daten, welche die Datengrundlage bilden, näher erklärt und das Konzept der prozeduralen Modellierung dargelegt. Zudem wird die Funktionsweise der in dieser Arbeit verwendeten Softwareprodukte FME und CityEngine in groben Zügen aufgezeigt.
Anschließend erfolgt eine Erläuterung zum Testgebiet und der Datengrundlage.
Einen großen Teil der Arbeit nimmt die praktische Umsetzung in Anspruch. In diesem werden die Datenaufbereitung und die Zerlegung der Schienentrasse in Bauelemente mittels FME behandelt, ebenso die Erstellung der Regeln und die Erzeugung der 3D-Modelle in der CityEngine.
Im nächsten Schritt werden die Ergebnisse visualisiert und evaluiert. Es erfolgt die Gegenüberstellung der Methode der Müller Systemtechnik GmbH und der im Rahmen dieser Arbeit erstellten Methodik. Ergänzend werden Verbesserungsmöglichkeiten aufgezeigt.
Den Abschluss bilden das Fazit und ein Ausblick auf mögliche weiterführende Schritte.

2 Verwandte Simulatoren und Arbeiten

Der Einsatz von Trainingssimulatoren ist in vielen verschiedenen Bereichen üblich. In dieser Arbeit wird geprüft, inwieweit eine automatisierte Generierung eines 3D-Modells für einen Straßenbahnsimulator möglich und sinnvoll ist.
Die Besonderheit stellt dabei der Ansatz dar, prozedurale Modellierung und Realitätsnähe zu kombinieren. Prozedurale Modellierung wird benutzt, um auf eine schnelle Weise virtuelle Umgebungen zu erschaffen, und zeigt sich somit als interessanter Ansatz für die Modellierung von 3D-Modellen, die in Simulatoren verwendet werden.
Nachfolgend werden für diese Arbeit interessante Simulatoren und Literatur vorgestellt.

Fahrsimulationssoftware SILAB
Die Fahrsimulationssoftware SILAB der Würzburger Institut für Verkehrswissenschaften GmbH (WIVW) kann für verschiedene Anwendungsbereiche genutzt werden, beispielsweise in der Forschung und Entwicklung, zum Testen von Fahrzeugkomponenten in einer realistischen und reproduzierbaren Fahrzeug- und Verkehrsumgebung oder für Training und Rehabilitation von Fahranfängern oder Einsatzfahrern [Würzburger Institut für Verkehrswissenschaften GmbH, 2014a]. Der Nutzer kann hierbei auf ein für unterschiedliche Fragestellungen bereitgestelltes Szenarienpaket zurückgreifen, hat jedoch auch die Möglichkeit, selbst auf persönliche Fragestellungen zugeschnittene Szenarien zu entwerfen. Bei der Erstellung solcher Szenarien kann der Nutzer bis in die Details gehen und Fahrbahnmarkierungen sowie Beschilderungen und Ampeln erstellen [Würzburger Institut für Verkehrswissenschaften GmbH, 2014b].

Simulationssoftware Zusi
Die Simulationssoftware Zusi ist eine Eisenbahnsimulation, bei der der Schwerpunkt auf dem Fahren eine Zuges liegt. Die Bedienung des Triebfahrzeugs sowie die betrieblichen Abläufe sind möglichst realitätsnah nachgebildet. Das Streckennetz sowie die Landschaften werden mithilfe von originalen Gleisplänen und digitalen Geländemodellen im UTM-Koordinatennetz gebaut. Durch die Hinterlegung von Trassierungsvorschriften und Weichenbauformen ist es dem Streckenbauer möglich, Abbilder von Originalstrecken zu entwerfen. Weiterhin lassen sich alle in Deutschland gebräuchlichen Signalsysteme nachbilden. Eine Strecke setzt sich aus zwei Datenpaketen zusammen: eines für die Funktionsstruktur der Strecke und eines für die Landschaftsinformationen. So sind mithilfe des ersten Datenpakets Nachbildungen aller erdenklichen Streckennetze durch beliebig lange, beliebig verzweigbare und beliebig positionierbare Vektoren realisierbar. Die Strecken können hierbei wunschgemäß mit Signalen, Ereignissen oder Weiterem versehen werden. Mittels des zweiten Da-

tenpakets lassen sich beliebige Landschaften erstellen und gestalten [Hölscher [2015b]]. Mithilfe des Gebäudeeditors ist zusätzlich das Erstellen von Loks, Waggons und Gebäuden möglich [Hölscher, 2015a].

Aufbau eines modularen Straßenbahnführerstandes für simulationsbasierte Arbeitsplatzuntersuchungen

Im Rahmen der Entwicklungen von Bedien- und Assistenzsystemen für eine optimale Unterstützung des Straßenbahnfahrers baut das Deutsche Zentrum für Luft- und Raumfahrt e.V. (DLR) einen modularen Straßenbahnführerstand. Ein Testbereich besteht darin, perzeptive und kognitive Prozesse auf Seiten des Fahrers zu untersuchen und mithilfe von simulationsgestützten Probandenstudien neuartige Assistenzsysteme und Automationskonzepte in Hinsicht auf Akzeptanz und Wirkung evaluieren zu können, wofür simulationsbasierte Untersuchungen nötig sind. Der Straßenbahnsimulator verfügt zu diesem Zweck über eine flexible Simulationssoftware. So können mithilfe von vordefinierten Kacheln verschiedene Verkehrsszenarien durch unterschiedliche Streckenbereiche gebildet werden. Jede Kachel stellt einen individuellen Streckenbereich mit einer Schnittstelle für weitere Kacheln dar. Überdies kann jede Kachel in Bezug auf ihre Fahrumgebung, sprich Gebäude, Natur und Verkehrsinfrastruktur editiert werden. So ist es möglich, mittels weniger Handgriffe unterschiedliche Szenarien zu schaffen [Grippenkoven u. a., 2014].

3D City Models for Simulation and Training Requirements on next Generation 3D City Models

Frank Bildstein untersucht in *3D City Models for Simulation and Training Requirements on next Generation 3D City Models*, inwiefern der Einsatz von 3D-Stadtmodellen als virtuelle Szenarien zu Ausbildungszwecken in Simulatoren geeignet ist. Die Untersuchung findet für verschiedene Anwendungsfelder von Simulatoren statt, unter anderem für Flugsimulatoren, Fahrsimulatoren oder nautische Simulatoren. Neben dem Aufbau und den typischen Komponenten der Simulatoren, werden die jeweiligen spezifischen Besonderheiten der virtuellen Datenbasen (= virtuellen Trainingsszenarien) samt Quelldaten und Erzeugungsverfahren beschrieben. Zudem wird dargestellt, inwieweit 3D-Stadtmodelle für virtuelle Szenarien von Nutzen sind. Beispielsweise könnte sich der Zeitaufwand für die Modellierung von städtischen Datenbasen wesentlich verringern und die Qualität verbessern. Des Weiteren werden Anforderungen an zukünftige 3D-Stadtmodelle genannt, unter anderem die Zugänglichkeit der Stadtmodelle durch das Anlegen eines Web-Katalogs zu verbessern oder die regelmäßige Aktualisierung von 3D-Stadtmodellen zu gewährleisten. Im Fazit hält der Autor fest, dass 3D-Stadtmodelle in einer Vielzahl von Simulatoren eingesetzt werden können, um die virtuellen Szenarien und somit die Qualität des Trainings in Bezug auf die Realitätsnähe zu verbessern. Weiterhin ist zu erwarten, dass der Gebrauch von 3D-Stadtmodellen den Aufwand für die Beschaffung und die Kosten für die Datengrundlage signifikant reduziert [Bildstein, 2005].

Procedural Modeling of Cities

In *Procedural Modeling of Cities* von Parish und Müller wird die Modellierung von Städten mithilfe der Software CityEngine erstmals vorgestellt. Die Erstellung einer Stadt wurde hier auf die Generierung

eines Verkehrsnetzwerkes und die Erzeugung von Gebäuden reduziert. Parish und Müller schlagen hierfür den Ansatz der prozeduralen Modellierung basierend auf L-Systemen vor. Als Eingangsdaten dienen Informationen aus geographischen Karten, z.B. Gewässergrenzen und statistischen Karten beispielsweise über die Bevölkerungsdichte. CityEngine generiert daraus ein System von Straßen, unterteilt das Land in Parzellen und erzeugt die Geometrie für die Gebäude sowie die Textur der Fassaden. Für die Erzeugung einer gesamten Stadt werden zwei verschiedene L-Systeme angewandt, eines für die Straßen- , ein anderes für die Gebäudegenerierung. Für die Erstellung des Straßennetzes einer Stadt wurden die L-Systeme mit Methoden erweitert, welche die Beachtung von globalen Zielen und lokalen Gegebenheiten erlauben und die Komplexität der Produktionsregeln reduzieren. Die Gebäude werden von einem stochastischen, parametrischem L-System generiert. Das System basiert auf einer hierarchischen Menge verständlicher Regeln, die den Bedürfnissen des Nutzers entsprechend angepasst werden können [Parish u. Müller, 2001].

Procedural Modeling of Buildings

Procedural Modeling of Buildings von Müller u. a. beschreibt, wie durch die *CGA Shape Grammatik*, einer *Shape Grammatik* für prozedurale Modellierung von Architekturmodellen, Gebäudemodelle von hoher visueller Qualität und hohem Detailreichtum erzeugt werden können. Mithilfe der Regeln der CGA Shape Grammatik wird zuerst das Grundmodell des Gebäudes erzeugt. Anschließend wird die Struktur der Fassade definiert und zuletzt die Details von Fenster, Türen und anderen Bauteilen festgelegt. So ist es möglich, aus einem Grundmodell viele verschiedene Gebäudevarianten zu schaffen. Im Vergleich zur manuellen Generierung ist diese Methode sehr viel zeitsparender [Müller u. a., 2006].

3 Theoretische Grundlagen

3.1 OpenStreetMap

Das OpenStreetMap (OSM)-Projekt wurde 2004 in London von dem Informatikstudenten Steve Coast ins Leben gerufen und hat das Ziel, eine „freie“ Weltkarte zu schaffen. „Frei“ bedeutet hier, dass alle Daten für jedermann frei zugänglich und kostenlos sind. OSM verwendet die Open Database Licence (ODbL); hierbei handelt es sich um eine freie Datenbank-Lizenz, die das Kopieren, Weitergeben, Benutzen und Ableiten von Werken aus der Datenbank legitimiert. Die Voraussetzung dafür ist, dass bei einer Weiterverwendung der Daten der Besitzer der Datenbank genannt wird und die Weitergabe unter den selben Bedingungen erfolgt, also wieder unter den ODbL Lizenzbedingungen [Immler, 2014].
Die OSM-Karte wird von unzähligen Freiwilligen, Mapper genannt, erstellt. Die Mapper erfassen die Daten zu Straßen, Gebäuden und vielem Weiterem mit Hilfe von GPS-Geräten oder verwenden z.B. zur Verfügung gestellte Luftbilder als Datengrundlage und pflegen diese in die OSM-Datenbank ein [Ramm u. Topf, 2010].
Neben der sogenannten OSM-Standardkarte gibt es Spezialkarten (vgl. Abbildung 3.1). Alle Spezialkarten gründen auf der OSM-Datenbank; je nach Erfordernis können gewünschte Details ausgeblendet, verstärkt oder anders dargestellt werden. Es existiert eine Vielzahl von Spezialkarten, darunter die Wander- und Fahrradkarte, die Seekarte oder die Feuerwehrkarte [Immler, 2014].

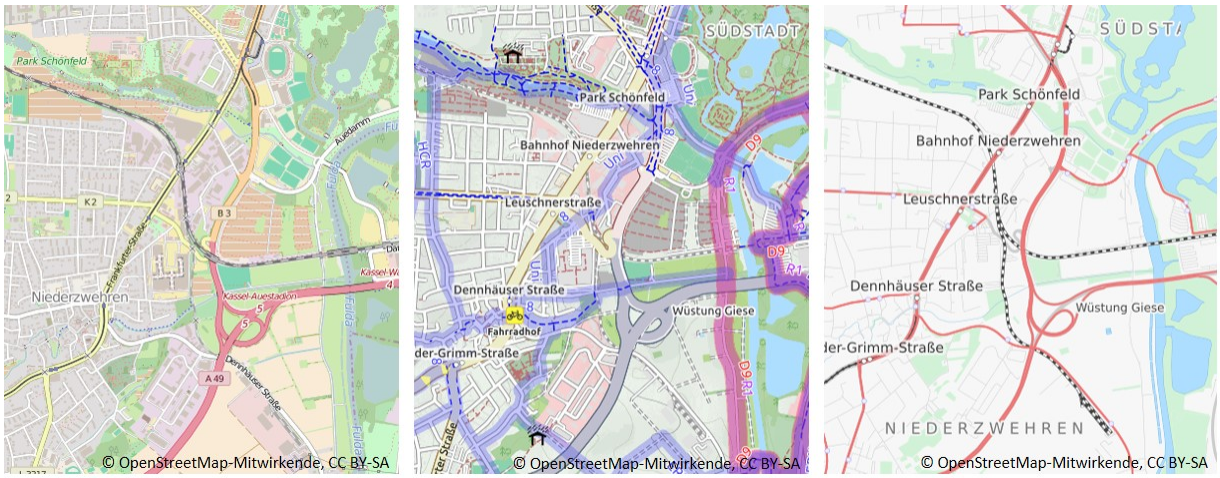

Abbildung 3.1 – OSM-Standardkarte (links), OSM-Radfahrerkarte (Mitte) und OSM-Verkehrskarte (rechts) [OpenStreetMap-Mitwirkende, 2016]

Die OSM-Karten basieren auf drei Grundelementen: dem Objekttyp *Node* (Punkt oder Knoten), dem Objekttyp *Way* (Linie) und dem Datentyp *Relation*. Allen drei Grundelementen werden zur genaueren Beschreibung sogenannte *Tags* zugeordnet; dabei handelt es sich um Attribute, bestehend aus *Key* (Schlüssel) und *Value* (Wert). Nodes werden entweder eigenständig verwendet, z.B. als "Point of Interest", indem sie den Standort eines Restaurants oder Kinos markieren, oder zu Ways verbunden. Ways dienen primär zur Darstellung linienförmiger Objekte, etwa Straßen, Bahnlinien oder Flüssen. Flächen wie Gebäude oder Seen werden in OSM als geschlossene Ways repräsentiert, d.h. der erste und der letzte Node sind identisch. Eine Relation ist eine sortierte Liste von Datenelementen (Nodes, Ways, Relationen) und Attributen, die zur Modellierung der Beziehung zwischen den Objekten verwendet wird. Oft werden Relationen zur Kennzeichnung von Verkehrswegen genutzt. Beispielsweise werden bei einer Buslinie die einzelnen Straßen (modelliert als Ways), die von dem Bus befahren werden, zu einer Relation zusammengefasst. Relationen können jedoch auch zur Definition von komplexen Flächen wie Wäldern oder landwirtschaftlichen Flächen angewandt werden.
Die auf einer Karte eingezeichneten Objekte werden *Map Features* genannt. Ein Map Feature bzw. ein Objekt (z.B. eine Straße, eine Ampel oder ein Briefkasten) wird entweder durch einen Node oder Way modelliert.
Wie bereits erwähnt, stehen die OSM-Daten jedermann zur Verfügung; so können auch die OSM-Rohdaten genutzt werden. Diese können bis zu einer bestimmten Größe direkt auf der OSM-Seite exportiert werden. Dabei handelt es sich um die OpenStreetMap-Daten im Extensible Markup Language (XML)-Format [Ramm u. Topf, 2010].
In Abbildung 3.2 ist ein beispielhafter Auszug aus den in dieser Arbeit verwendeten OSM-Rohdaten im XML-Format dargestellt.

```
<way id="37113421" visible="true" version="6" changeset="17208119"
   timestamp="2013-08-03T20:15:45Z" user="Rowil" uid="1672230">
 <nd ref="431672285"/>
 <nd ref="431672517"/>
 <tag k="highway" v="primary"/>
 <tag k="lanes" v="2"/>
 <tag k="maxspeed" v="50"/>
 <tag k="name" v="Frankfurter Straße"/>
 <tag k="oneway" v="yes"/>
 <tag k="ref" v="L 3219"/>
</way>
```

Abbildung 3.2 – Auszug aus den OSM-Rohdaten im XML-Format [OpenStreetMap-Mitwirkende, 2016]

Das Objekt in Abbildung 3.2 ist als Way modelliert, dessen Anfang und Ende jeweils durch einen Node (nd) gekennzeichnet sind. Das Objekt besitzt sechs Tags, die es näher beschreiben. Das erste Attribut - bestehend aus dem Key-Value Paar *highway = primary* - zeigt, dass es sich um eine Straße (*highway*) mit dem Wert *primary* handelt, was in Deutschland üblicherweise für Bundesstraßen oder Straßen mit übergeordneter Verkehrsbedeutung verwendet wird. Die weiteren Attribute besagen, dass die Straße zweispurig (*lanes = 2*) und nur in eine Richtung befahrbar (*oneway = yes*) ist sowie den Namen Frankfurter Straße (*name = Frankfurter Straße*) trägt. Die Maximalgeschwin-

digkeit beträgt 50 km/h (*maxspeed = 50*) und die Kurzbezeichnung der Straße lautet L3219 (*ref = L3219*).

3.2 Prozedurale Modellierung

Prozedurale Modellierung ist bereits seit über dreißig Jahren ein stetes Forschungsthema, dessen Anwendungsbereich breit gefächert ist und beispielsweise die Modellierung von Texturen, Pflanzen, Gebäuden oder Straßennetzwerken umfasst. Prozedurale Modellierung befasst sich mit der (semi-)automatischen Generierung von Modellen mittels eines Programms oder einer Prozedur. Vor allem die Datenkompression und die Möglichkeit, aus einer geringen Zahl an Eingangsdaten und mittels weniger Regeln eine große Variation von detaillierten Modellen zu erstellen, machen die prozedurale Modellierung für die Erzeugung von virtuellen Umgebungen attraktiv [Smelik u. a., 2014]. Die Schlüsseleigenschaft der prozeduralen Generierung ist die Beschreibung von Entitäten, ob Geometrie oder Textur, als Abfolge von Produktionsregeln und nicht als statische Objekte. Es existiert eine Vielzahl von prozeduralen Techniken; L-Systeme, Shape Grammatiken, Perlin Noise und Fraktale, um nur einige zu nennen [Kelly u. McCabe, 2006].
Im Folgenden werden die L-Systeme, die Split Grammatik und die darauf basierende CGA Shape Grammatik erläutert, da diese die Basis der Software CityEngine bilden.

3.2.1 L-Systeme

Der Biologe Aristid Lindenmayer schlug 1968 eine mathematische Theorie zur Beschreibung von Pflanzenwachstum vor, die heute unter dem Namen Lindenmayer-System, oder kurz L-System, bekannt ist [Lindenmayer, 1968]. Das zentrale Konzept der L-Systeme ist das der Ersetzungssysteme, auch *rewriting systems* genannt. Beim rewriting werden komplexe Objekte definiert, indem Teilstücke einfacher Ursprungsobjekte mithilfe von Produktionsregeln (rewriting rules) sukzessiv ersetzt werden. Das L-System zählt zudem zu den parallelen Ersetzungssystemen, was bedeutet, dass in jedem Ersetzungsschritt jeder Buchstabe umgeschrieben wird.
Es werden verschiedene Klassen von L-Systemen unterschieden. Das simpelste unter den L-Systemen ist das deterministische, kontextfreie L-System; dieses wird DOL-System genannt. Es besteht aus den folgenden drei Komponenten:

- den *Variablen* ***V***
- dem *Axiom* oder *Anfangswort* ω und
- den *Produktionsregeln* ***P***

Wobei die Variablen V das Alphabet des Systems darstellen. Die Abbildung 3.3 zeigt ein Beispiel für ein DOL-System mit n = 5 Iterationen.

		n=0	a
Variablen	$V = \{a, b\}$	n=1	b
Axiom	$\omega : a$	n=2	ba
Produktionsregeln	$p_1: a \rightarrow b$ $p_2: b \rightarrow ba$	n=3	bab
		n=4	babba
		n=5	babbabab

Abbildung 3.3 – Beispiel eines deterministischen und kontextfreien L-Systems

Beginnend beim Axiom *a* wird dieses durch Anwendung der Produktionsregel p_1 durch *b* ersetzt und *b* durch die Anwendung der Regel p_2 wiederum durch *ba*. Auf den Buchstaben *b* des Strings *ba* wird nun die Regel p_2 und auf den Buchstaben *a* die Regel p_1 angewandt und so weiter. Eine Produktionsregel $a \rightarrow b$ besteht aus dem *predecessor a* und dem *succesor b* [Prusinkiewicz u. Lindenmayer, 1990].
Die Anwendung von L-Systemen hat vor allem in der Modellierung von Pflanzen beachtliche Erfolge erzielt [Prusinkiewicz u. a., 1994, Měch u. Prusinkiewicz, 1996]. Die Abbildung 3.4 zeigt eine dreidimensionale strauchartige Struktur, generiert mithilfe eines L-Systems. Auch die Software CityEngine bedient sich der L-Systeme als Technik für die prozedurale Modellierung. Mithilfe von L-Systemen wird in der CityEngine ein System von Straßen und Geometrien für Gebäude erzeugt [Parish u. Müller, 2001].

Abbildung 3.4 – Dreidimensionale strauchartige Struktur, generiert mittels eines L-Systems [Prusinkiewicz u. Lindenmayer, 1990]

3.2.2 Split Grammatik

Die *Split Grammatik* wird von Wonka u. a. [2003] vorgestellt und ist vor allem für die automatische Modellierung von Gebäuden geeignet. Die Split Grammatik arbeitet auf der Grundlage von Shapes,

genannt *basic shapes*, denen Attribute, Parameter und Symbole zugeordnet sind. Basic Shapes sind einfache geometrische Objekte wie Quader, Zylinder oder Prismen. Die Neuheit der Split Grammatik liegt in der Begrenzung der erlaubten Regeln. Diese Beschränkung sorgt dafür, dass die Split Grammatik einerseits mächtig genug für die Modellierung von Gebäuden ist und andererseits einfach genug für einen kontrollierten und automatischen Ableitungsprozess. Es sind zwei Arten von Regeln erlaubt, die *Split Regel* und *Conversion Regel* genannt werden. Auf Gebäude angewandt generiert die Grammatik - ausgehend von einem Initialshape - die Fassade des Gebäudes, die wiederum in strukturelle Elemente zerlegt wird, bis hin zur Ebene individueller Bauelemente wie beispielsweise Fensterbänken. Die folgende Abbildung 3.5 zeigt eine einfache Split Grammatik für eine Gebäudefassade und deren Ableitungsprozess mit entsprechendem Ergebnis. Zu Beginn wird das Initialshape (START) in vier identische Fassadenelemente (F) unterteilt. Das Fassadenelement wird wiederum in ein Fensterelement (W), ein Schlusssteinelement (KS) (engl.: keystone) und mehrere Wandelemente gegliedert. Die Fensterelemente (W) erfahren noch eine weitere Unterteilung in Fenster (WIN) und Fensterrahmen [Wonka u. a., 2003].

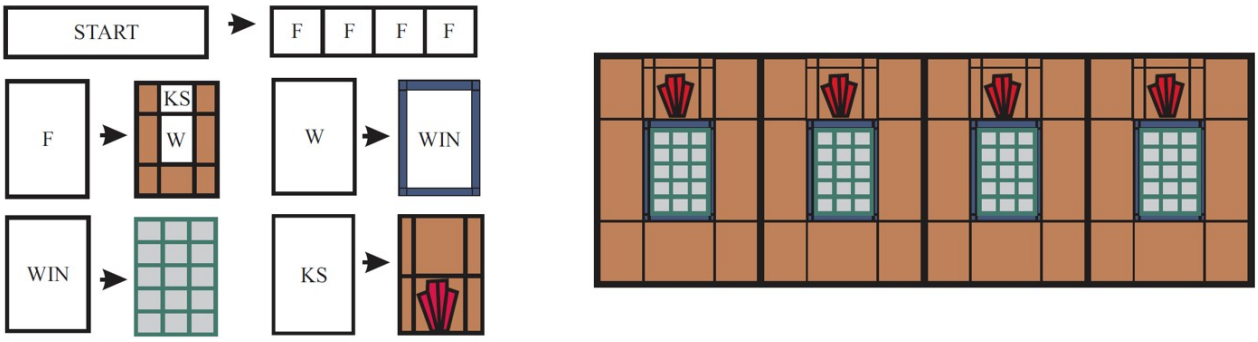

Abbildung 3.5 – Beispiel einer einfachen Split Grammatik (links) mit entsprechendem Ergebnis nach dem Ableitungsprozess (rechts) [Wonka u. a., 2003]

3.2.3 CGA Shape Grammatik

Neben den L-Systemen findet auch die CGA Shape Grammatik Anwendung in der CityEngine, wobei CGA für Computer Generated Architecture steht. Die Idee, Shape Grammatik für die Modellierung von urbanen Umgebungen zu nutzen, wird von Parish u. Müller [2001] zusammen mit Wonka u. a. [2003] vorgestellt. Auf der einen Seite zeigen Parish und Müller, wie große urbane Umgebungen generiert werden können, wobei jedes Gebäude aus einem simplen Grundmodell besteht und Schattierungen für die Darstellung von Fassadendetails genutzt werden. Auf der anderen Seite demonstriert Wonka u. a., wie es mithilfe von Split Grammatiken möglich ist, Fassaden einzelner Gebäude mit geometrischen Details zu erzeugen. Die CGA Shape Grammatik vereint diese beiden Ansätze und versucht so, große und detaillierte urbane Umgebungen zu generieren.
Die CGA Shape Grammatik arbeitet auf der Grundlage von Shapes, wobei ein Shape aus einem Symbol, einer Geometrie und numerischen Attributen besteht. Zu den wichtigsten geometrischen Attributen zählen die Position P, drei orthogonale Vektoren X, Y und Z, welche ein Koordinatensystem

bilden, und ein Vektor S, der die Größe beschreibt. Abbildung 3.6 zeigt diese Attribute, mittels denen eine im Raum orientierte Bounding Box definiert wird, die als *scope* bezeichnet wird.

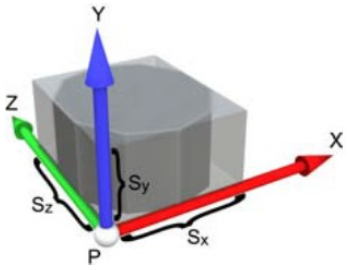

Abbildung 3.6 – Darstellung der Bounding Box mit ihren Attributen, genannt *scope* [Müller u. a., 2006]

Die CGA Shape Grammatik umfasst fünf verschiedene Regeln, die teilweise auf der Idee der Split Grammatik (vgl. Kapitel 3.2.2) beruhen. Mit den *Scope Regeln* können Shapes mittels Translation, Rotation und Skalierung verändert werden; die *Basic Split Regel* dient zur Aufteilung des Scopes in mehrere Bestandteile. Zwei weitere Regeln umfassen das *Skalieren* von Objekten und das *Wiederholen* von Objekten. Zuletzt ermöglicht der *Komponenten Split* das Zerlegen von Shapes in Komponenten niedrigerer Dimensionen.
In der CGA Shape Grammatik wird mithilfe von Produktionsregeln in einem iterativen Prozess eine immer höhere Detailtiefe gewonnen. Zuerst wird mittels der Produktionsregeln ein einfaches Grundmodell erzeugt, genannt *mass model*; anschließend wird die Fassade strukturiert, danach folgen die Details von Fenster, Türen und Ornamenten.
Die Notation der Grammatik sowie die generelle Verwendung von Regeln sind angelehnt an das Schema der L-Systeme (vgl. Kapitel 3.2.1). Die CGA Shape Grammatik zählt zu den sequenziellen Grammatiken, was eine sequenzielle Anwendung der Regeln und somit die Beschreibung einer Struktur ermöglicht [Müller u. a., 2006].
Die Abbildung 3.7 zeigt ein Beispiel für eine CGA-Regel, deren Hierarchie und das resultierende 3D-Modell. Die Startregel A wird auf das initiale Shape angewandt und unterteilt dieses in drei Bereiche, die in den untergeordneten Regeln B, C und D näher definiert werden. Wobei B 20 %, C 50 % und D 30 % des ursprünglichen Shapes einnehmen. In der Regel B, C und D werden die Shapes um die Höhen 3, 1 und 4 extrudiert; zudem werden diesen die Farben rot, grün und blau zugewiesen. Nach der erfolgreichen Anwendung der Regel auf das initiale Shape entsteht das rechts abgebildete 3D-Modell. Mittig ist die Hierarchie der Regel aufgeführt. Es ist zu erkennen, dass die Regel A über den gleichberechtigten Regeln B, C und D steht. In der Hierarchie sind ebenfalls die jeweiligen Shape Operationen (split, extrude, color) aufgeführt.

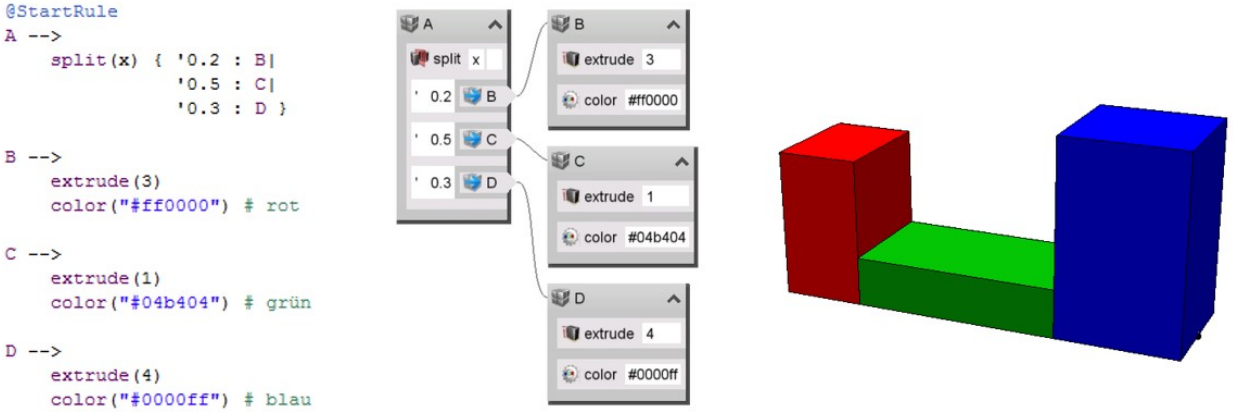

Abbildung 3.7 – Beispiel einer CGA-Regel (links), deren Hierarchie (mittig) und das resultierende 3D-Modell (rechts)

3.3 Verwendete Software

Dieses Kapitel gibt einen Überblick über die in dieser Arbeit verwendete Software. Mithilfe der Feature Manipulation Engine (FME) werden die OSM-Daten aufbereitet und die Schienentrasse in Bauelemente zerlegt. In der CityEngine werden die CGA-Regeln für die Bauelemente der Schienentrasse erstellt und angewandt, sodass ein 3D-Modell erzeugt wird.

3.3.1 Feature Manipulation Engine

Die Feature Manipulation Engine (FME) ist ein Softwareprodukt des kanadischen Unternehmens Safe Software Inc. und dient zur formatunabhängigen Verarbeitung von räumlichen und nicht-räumlichen Daten. FME zählt zu den Spatial ETL-Werkzeugen. Hierbei steht E für **E**xtrahieren (Extract), T für **T**ransformieren (Transform) und L für **L**aden (Load); Spatial repräsentiert die räumlichen Daten, die im Englischen Spatial Data genannt werden. Mithilfe von FME können Daten aus unterschiedlichen Datenquellen, mittels sogenannter *Reader*, extrahiert und in ein FME-internes Format übersetzt werden. Das Programm bietet eine Vielzahl an *Transformern*, durch deren Anwendung die Daten je nach Belieben des Nutzers bearbeitet, angereichert, ausgedünnt, in ihrer Struktur verändert oder transformiert werden können. Nach deren Bearbeitung lassen sich die Daten mithilfe sogenannter *Writer* in einem oder mehreren Zielformaten ausgeben oder in Datenbanken laden.
Das Softwareprodukt FME Desktop untergliedert sich in drei Komponenten: die FME Workbench, den FME Data Inspector und den FME Quick Translator. In dieser Arbeit wird die FME Workbench zur Aufbereitung der OpenStreetMap-Daten genutzt. In der FME Workbench werden Prozesse zur Datenkonvertierung und -transformation in einer grafischen Oberfläche modelliert und ausgeführt [con terra GmbH, 2015]. Die Abbildung 3.8 zeigt an einem einfachen Beispiel das Prinzip, auf dem die Arbeitsweise mit der FME Workbench basiert. Mittels Readern werden die Daten eingelesen, durch die Anwendung von Transformern manipuliert und schließlich mithilfe von Writern neue Daten erzeugt. In dem beispielhaften FME Workflow gibt es die zwei Reader *Tramlinie* und *Gebäude_allgemein*. Um

die Tramlinie wird mittels des Transformers *GeographicBufferer* ein Puffer von 100m erzeugt; anschließend werden die Gebäude und der eben erzeugte Puffer mittels des Transformers *SpatialFilter* miteinander verschnitten. Das Resultat, welches durch den Writer *Gebäude_allgemein_100m* in einer Datei abgespeichert wird, bilden alle Gebäude, die komplett oder teilweise innerhalb des Puffers liegen. FME wird in dieser Arbeit für die Datenaufbereitung genutzt.

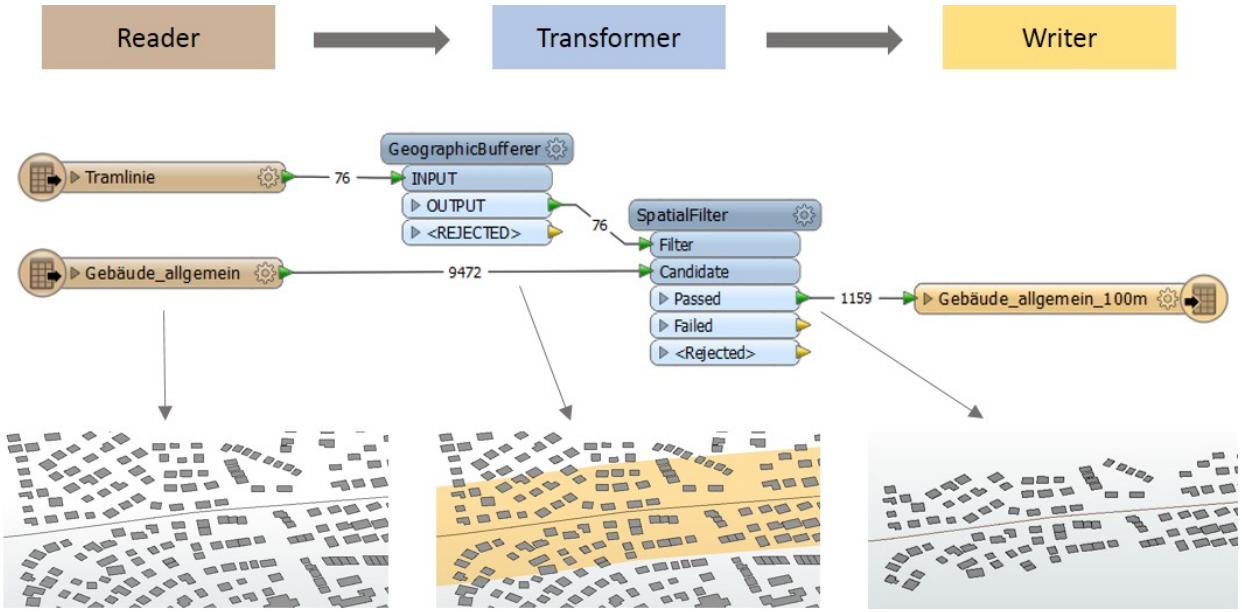

Abbildung 3.8 – Prinzip der FME Workbench anhand eines Beispiels

3.3.2 CityEngine

CityEngine ist eine Software der Firma ESRI Inc. für die 3D-Modellierung von urbanen Umgebungen. Mithilfe der CityEngine können mittels einer vergleichsweise kleinen Datenmenge Städte von Grund auf erzeugt werden, basierend auf einer hierarchischen Struktur verständlicher Regeln, die der Nutzer seinen Bedürfnissen entsprechend verändern kann [Parish u. Müller, 2001]. Das wesentliche Konzept der CityEngine ist dabei der prozedurale Lösungsansatz, um die Modellierung möglichst effizient zu gestalten. Anstatt mit dem Modell manuell zu interagieren und die 3D-Geometrie zu modellieren, wird die Aufgabe in abstrakter Form beschrieben. Dies geschieht durch eine Reihe von Modellierungsanweisungen in Form von Regeln. Bei der Erstellung der Regeln findet die CGA Shape Grammatik (vgl. Kapitel 3.2.3) Anwendung; typische Operationen sind z.B. *extrude* oder *split*. In der Abbildung 3.9 ist der typische CityEngine Modellierungsablauf dargestellt, der dazu gedacht ist große urbane Umgebungen möglichst effizient zu erstellen. Zuerst wird dabei ein Straßennetzwerk generiert; daraufhin werden die freien Flächen zwischen den Straßen in Parzellen unterteilt. Schließlich wird die Geometrie für die Gebäude erzeugt, wobei diese durch die Anwendung der CGA Regeln eine hohe Detailtiefe erreichen kann. Der Modellierungsablauf kann an beliebiger Stelle begonnen werden. In dieser Arbeit wird auf eine automatische Erzeugung des Straßennetzwerks ver-

zichtet, stattdessen wird eine bereits bestehende Straßentrasse eingeladen [Esri R&D Center Zurich, 2016].

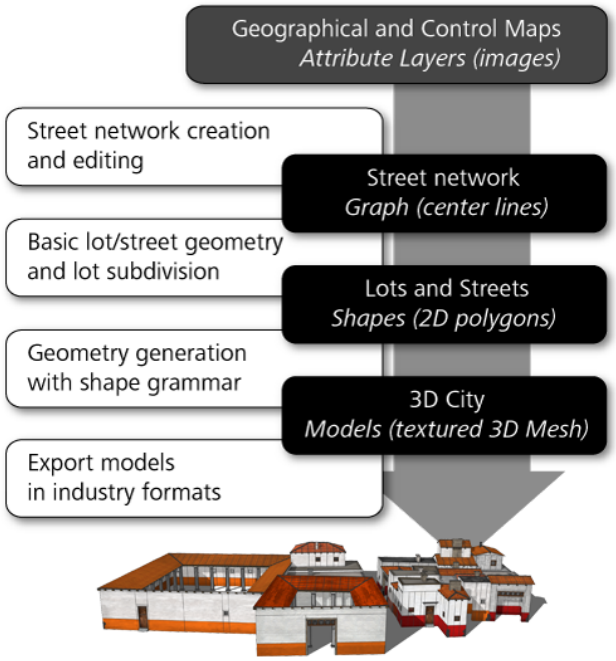

Abbildung 3.9 – CityEngine Modellierungsablauf [Esri R&D Center Zurich, 2016]

4 Testgebiet und Datengrundlage

Gegenstand dieser Arbeit ist, die vorwiegend manuelle Generierung des 3D-Modells für einen Straßenbahnsimulator mithilfe prozeduraler Modellierung soweit wie möglich zu automatisieren und zu untersuchen, inwieweit eine Automatisierung sinnvoll ist. Als Datengrundlage dienen OSM-Daten (siehe Kapitel 3.1). Gegenüber herkömmlichen Karten verfügt OSM über einige Vorteile, die für diese Arbeit von Nutzen sind:

- Bei den OSM-Daten handelt es sich um freie Daten, d.h. sie sind für jedermann kostenlos und frei verfügbar. Somit müssen keine Lizenzgebühren gezahlt werden müssen.
- OSM stellt nicht nur das Kartenmaterial, sondern auch die zugrundeliegenden Geodaten (Rohdaten) zur freien Verfügung.
- Bei OSM-Daten handelt es sich um aktuelle Daten. Im Gegensatz zu herkömmlichen Karten wird die OSM-Datenbank kontinuierlich erweitert und somit die OSM-Karte laufend aktualisiert.
- OSM-Daten sind weltweit verfügbar. Dies bedeutet, dass das Ergebnis dieser Arbeit theoretisch weltweit angewandt werden kann.
- OSM-Daten enthalten viele hilfreiche und wertvolle Informationen, die herkömmliche Karten nicht enthalten. Angefangen bei der Anzahl der Straßenspuren und der erlaubten Fahrtrichtungen über die Spurbreite der Straßenbahnschienen bis hin zu den Standorten von Ampeln.

Jedoch gilt es zu beachten, dass OSM auf der Arbeit von Freiwilligen basiert; die Korrektheit der Daten ist nicht zwingend gewährleistet. Weiterhin ist zu berücksichtigen, dass unterschiedliche Orte unterschiedliche Datenquantität und -qualität aufweisen; so sind Metropolen wie New York besser erfasst als andere Städte.
Die Datengrundlage in dieser Arbeit bilden die OSM-Rohdaten. Dabei handelt es sich um eine Datei mit der Endung .osm. Die OSM-Rohdaten können bis zu einer Größe von rund 1000 m^2, je nach Datendichte, direkt über die OSM-Seite heruntergeladen werden [OpenStreetMap, 2016]. Der Ausschnitt des Testgebiets in dieser Arbeit überschreitet diese Maximalgröße. Die Daten für den Testdatensatz werden daher als zwei .osm-Dateien exportiert und anschließend in der Datenaufbereitung mithilfe von FME wieder vereint.
Die Abbildung 4.1 zeigt das Testgebiet dieser Arbeit, das sich in der Stadt Kassel befindet. Die Teststrecke besitzt eine Länge von etwa 9 km und wird von der Tramlinie 5 befahren. Sie umfasst insgesamt 19 Haltestellen, wobei die beiden Endhaltestellen der Teststrecke „Großenritte“ und „Auestadion“ heißen. Die Haltestelle „Großenritte“ befindet sich im Südwesten der Karte in der Stadt Baunatal

und die Haltestelle „Auestadion“ liegt im Nordosten in der Stadt Kassel.
Da für einen Straßenbahnsimulator lediglich das Sichtfeld des Straßenbahnführers von Bedeutung ist, beschränkt sich das Testgebiet auf einen Teilbereich des abgebildeten Ausschnitts. Dieser Bereich wird von einem Puffer mit einer Pufferdistanz von 100 m erfasst. Somit entspricht das Testgebiet einem Bereich von 200 m Breite, 100 m auf der linken und 100 m auf der rechten Seite der Schienentrasse.
Das Testgebiet wurde ausgewählt, da für diese Strecke bereits ein 3D-Modell der Firma Müller Systemtechnik GmbH vorhanden ist, welches manuell erstellt wurde. Am Ende der Arbeit werden die in dieser Arbeit gewonnenen Ergebnisse zur automatisierten Generierung der manuellen Erstellung des 3D-Modells gegenübergestellt.

Abbildung 4.1 – Teststrecke und dazugehöriges Testgebiet

5 Praktische Umsetzung

Ziel dieser Arbeit ist eine möglichst automatisierte Generierung eines 3D-Modells für einen Straßenbahnsimulator. Die automatisierte Generierung wird exemplarisch für eine Teststrecke implementiert, wobei die Teststrecke in dieser Arbeit durch eine Teilstrecke der Tramlinie 5 in Kassel (Haltestellen „Großenritte“ bis „Auestadion“, vgl. Kapitel 4) repräsentiert wird.
Das Kapitel „Praktische Umsetzung“ zeigt die Vorgehensweise dieser Arbeit auf. Angefangen beim Export der OSM-Rohdaten über die Erstellung der Shapes für die CityEngine bis hin zur 3D-Modellierung. Die Abbildung 5.1 zeigt das zugrundeliegende Gesamtkonzept.
Der erste Schritt stellt das Exportieren der OSM-Rohdaten dar. Dies geschieht mittels zwei .osm-Dateien (vgl. Kapitel 4). Im darauffolgenden Schritt werden die OSM-Daten gesichtet, reduziert und umstrukturiert. Weiterhin wird die Schienentrasse in ihre Elemente zerlegt. Das Ergebnis der Datenaufbereitung sind die Shapes für die 3D-Modellierung in der CityEngine. Wobei zwischen der Datenaufbereitung und der Modellierung mit CityEngine die Option für die Durchführung manueller Korrekturen besteht. Anschließend werden in der CityEngine die CGA-Regeln für die 3D-Modellierung erstellt und auf die Shapes angewandt, sodass als Ergebnis ein 3D-Modell entsteht. Da es im Rahmen dieser Arbeit nicht möglich ist ein gesamtes 3D-Modell zu erstellen, liegt der Fokus dieser Arbeit auf der Erstellung der CGA-Regeln und Modellierung der Schienentrasse. Nach der 3D-Modellierung können wiederum manuelle Korrekturen vorgenommen werden.
Im Folgenden werden die einzelnen Schritte näher erläutert.

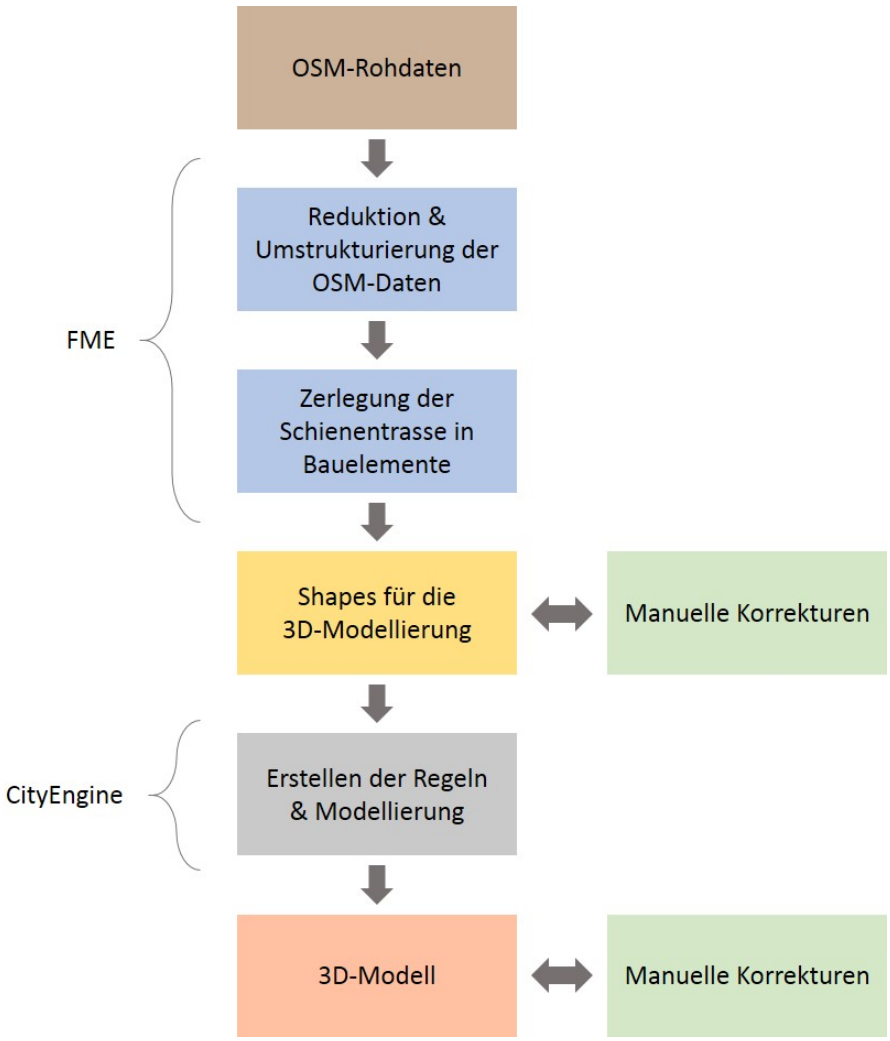

Abbildung 5.1 – Gesamtkonzept

5.1 Datenaufbereitung

Dieses Kapitel befasst sich mit der Aufbereitung der OSM-Rohdaten mithilfe von FME für die spätere 3D-Modellierung in der CityEngine. Das Ziel ist die Umwandlung der OSM-Rohdaten in Shape-Dateien (.shp-Format), auf deren Grundlage die Modellierung stattfindet. Die Datenaufbereitung ist in zwei Blöcke untergliedert, wobei jeder Block in einem FME-Workspace umgesetzt wird. Im ersten Block werden die OSM-Rohdaten sortiert, auf die relevanten Daten reduziert und in eine für diese Arbeit günstige Struktur gebracht. Dies geschieht für alle übernommenen OSM-Daten. Im zweiten Block erfolgt die Zerlegung der Tramlinie in Bauelemente, was nur für die OSM-Daten im Testgebiet erfolgt. Nachdem die Datenaufbereitung abgeschlossen ist, besteht die Möglichkeit manuelle Korrekturen an den in FME erzeugten Shapes vorzunehmen. Die Abbildung 5.2 veranschaulicht die Vorgehensweise der Datenaufbereitung. Nachfolgend werden die einzelnen Schritte näher erläutert.

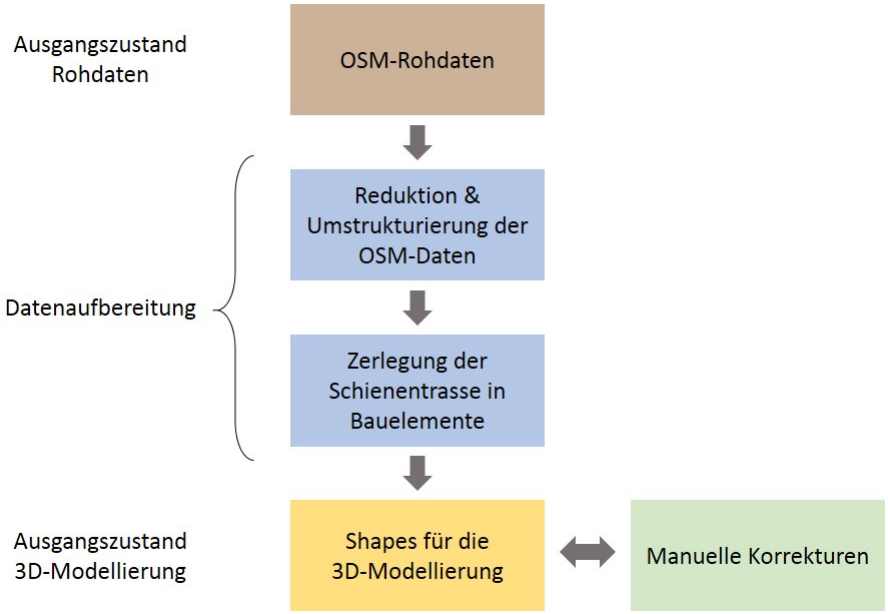

Abbildung 5.2 – Konzept für die Aufbereitung der OSM-Rohdaten

5.1.1 Reduktion und Umstrukturierung der OSM-Daten

Als Datengrundlage dienen in dieser Arbeit OSM-Rohdaten. Um die Daten für die 3D-Modellierung mit der CityEngine nutzen zu können, müssen sie zuvor aufbereitet werden. Die Abbildung 5.3 zeigt die Vorgehensweise im ersten Block der Aufbereitung der OSM-Daten. Wobei die wesentlichen Schritte nach dem Laden und Sichten der Daten deren Reduktion und Umstrukturierung ist. Die praktische Umsetzung der in der Abbildung aufgezeigten Schritte findet im FME Workspace *ReduktionUmstrukturierung.fmw* statt. Die „Reduktion und Umstrukturierung der OSM-Daten“ findet für den gesamten OSM-Rohdatensatz statt. Im Folgenden werden die einzelnen Schritte näher erklärt.
Wie bereits im vorherigen Kapitel erwähnt, werden die OSM-Rohdaten für das Testgebiet in zwei .osm-Dateien exportiert (vgl. Kapitel 4). Der erste Schritt stellt die Zusammenführung der beiden Dateien dar, die durch das gemeinsame Laden in FME erfolgt.

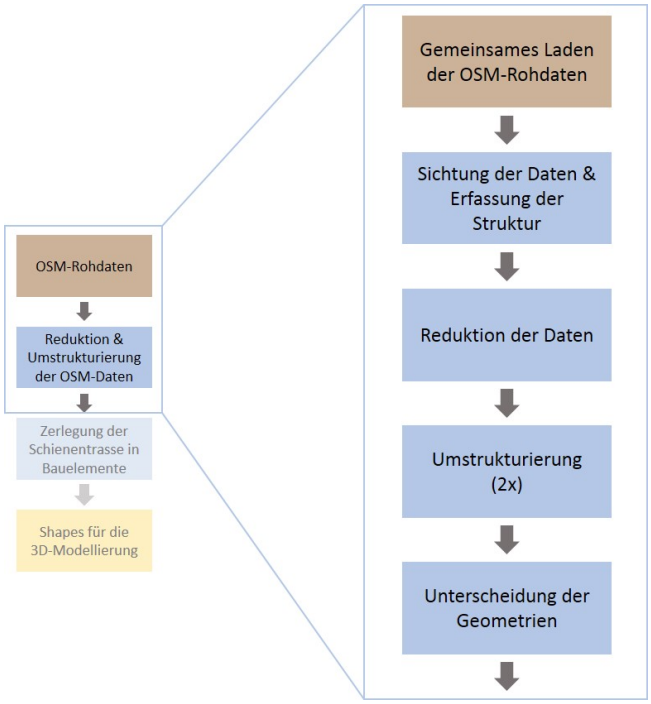

Abbildung 5.3 – Konzept für die Reduktion und die Umstrukturierung der OSM-Daten

Im nächsten Schritt werden die OSM-Daten gesichtet und deren Struktur wird erfasst. Bereits zu Beginn der Arbeit wurde auf die relevanten Komponenten eines 3D-Modells für einen Straßenbahnsimulator eingegangen, zudem wurden diese aufgeführt (vgl. Kapitel 1.2). Bei der Sichtung der Daten fällt auf, dass diese deutlich mehr Informationen enthalten als benötigt werden. Eine Reduktion auf die für diese Arbeit relevanten Daten ist daher sinnvoll. Beim Laden der OSM-Daten wird jeder, in den Daten enthaltene Schlüssel (Key) in FME als eigener Reader dargestellt. Die Abbildung 5.4 zeigt die Reduktion der in FME geladenen OSM-Daten von 39 auf 12 Reader.

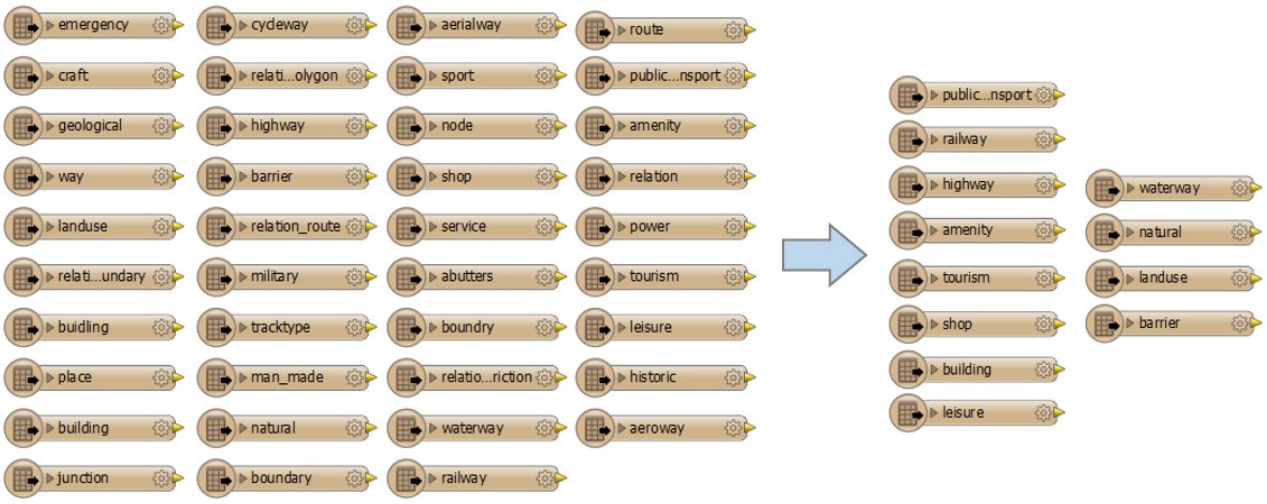

Abbildung 5.4 – Reduktion der OSM-Daten

Die nachfolgende Tabelle 5.1 führt die 12 verbliebenen Schlüssel auf, nennt deren Bedeutung und einige beispielhafte Werte (Value). Beim Betrachten der Tabelle fällt auf, dass die OSM-Daten eine andere Struktur als die Anforderungen (vgl. Kapitel 1.2) aufweisen. So sind die OSM-Daten in „Themen“ untergliedert. Beispielsweise werden unter dem Schlüssel *tourism* alle Objekte aufgeführt, die in die Kategorie Tourismus fallen. Dabei wird nicht differenziert, ob es sich etwa um ein Gebäude (z.B. ein Hotel oder ein Museum) oder um eine Sehenswürdigkeit handelt; unter dem Begriff Sehenswürdigkeit werden wiederum viele verschiedene Objekte (u.a. Gebäude, Bauwerke wie Tore und Türme) zusammengefasst. Auch die in OSM dargestellten Gebäude unterliegen einer ungünstigen Struktur. So wird nur ein Teil der Gebäude unter dem Schlüssel *building* aufgeführt, die restlichen Gebäude verteilen sich auf die Schlüssel *amenity*, *tourism* und *shop*.

Schlüssel (Key)	**Bedeutung**	**Beispielhafte Werte (Value)**
public_transport	Öffentlicher Nahverkehr	Bahnsteige, Haltepositionen (Zug)
railway	Eisenbahn (Schienenwege)	Eisenbahn, U-Bahn, Straßenbahn, Haltestellen, Bahnübergänge, Weichen, …
highway	Straßen, Wege	Autobahnen, Bundesstraßen, Landstraßen, Kreisstraßen, Fußwege, Wanderwege, Radwege, …
amenity	Nutzung/Einrichtung	Bars, Cafés, Hochschulen, Schulen, Tankstellen, Parkplätze, Geldautomaten, Krankenhäuser, …
tourism	Tourismus	Sehenswürdigkeiten, Hotels, Ferienwohnungen, Museen, Berghütten, Freizeitparks, Rastplätze, …
shop	Geschäfte	Getränkemärkte, Gemüsehändler, Einkaufszentren, Bekleidungsgeschäfte, Optiker, Floristen, Autohäuser, …
building	Gebäude	Häuser, Kirchen, Gewächshäuser, Palaste, …
leisure	Freizeit	Angelstellen, Gärten, Minigolf, Grünanlagen, Spielfelder (Fußball, Tennis, usw.), Spielplätze, Schwimmbecken, …
waterway	Wasserläufe	Flüsse, Bäche, Flussufer, Kanäle, Abwassergräben, Wasserfälle, Staudämme, …
natural	Natur	Wälder, Alleen, Bäume, Gletscher, Strände, Gebirge, …
landuse	Landnutzung	Kleingärten, Forste, Garagen, Rasenflächen, Erholungsgebiete, Wohngebiete, Obstplantagen, …
barrier	Barrieren	Zäune, Gräben, Hecken, Bordsteine, Mauern, Schranken, Mautstellen, Geländer, Leitplanken, …

Tabelle 5.1 – Schlüssel mit Bedeutung und beispielhaften Werten

Des Weiteren enthalten die OSM-Daten auch einige für das 3D-Modell überflüssige Informationen. So ist es beispielsweise für den Wiedererkennungswert eines 3D-Modells nicht von Bedeutung, an welchen Orten sich Geldautomaten (vgl. *amenity*) befinden. Folglich bestehen die nächsten Schritte darin, die relevanten Daten herauszufiltern und die gefilterten Daten in eine geeignetere Struktur zu bringen.
Die Tabelle 5.2 zeigt, wie die Daten strukturiert werden und welche Inhalte welchen Shapes zugeord-

net werden. Obwohl die OSM-Daten Informationen zu Weichen, Gleiskreuzungen, Bahnübergängen und Fußgängerübergängen enthalten, wird an dieser Stelle darauf verzichtet, aus den Informationen Shapes zu erzeugen. Die Erzeugung der in Klammern gesetzten Shapes (siehe rechte Tabellenspalte) wird durch die in Kapitel 5.1.2 durchgeführte Zerlegung der Schienentrasse in Bauelemente vorgenommen.

Schlüssel (Key)	**Shapes für CityEngine**
public_transport	Bahnsteige
railway	Schienen Haltestellen (Weichen) (Gleiskreuzung) (Bahnübergang) (Fußgängerübergang)
highway	Straßen Fußwege Ampeln
amenity	Parkplätze Gebäude
tourism	Gebäude
shop	Gebäude
building	Gebäude
leisure	Gebäude Vegetation
waterway	Wasserläufe
natural	Vegetation
landuse	Vegetation
barrier	Vegetation

Tabelle 5.2 – Zuordnung der Schlüssel zu den Shapes

Nach der Anpassung der Struktur der OSM-Daten an die der Anforderungen, folgt die zweite Umstrukturierung.

Die Abbildung 5.5 zeigt am Beispiel eines Straßenabschnitts (orange markiert), welche Informationen dem Nutzer beim Anklicken eines Map Features angezeigt werden. Die Tabelle in der Abbildung stellt die Auflistung aller zu dem Straßenabschnitt zugehörigen Attribute dar. Die Attribute besagen, dass es sich um eine Straße vom „Typ“ *secondary* (*highway = secondary*) handelt, die den Namen Frankfurter Straße (*name = Frankfurter Straße*) trägt. Zudem ist die Straße dreispurig (*lanes = 3*), wobei die Fahrtrichtungen der Fahrspuren der Richtung des OSM-Weges entsprechen (*lanes:forward = 3*) und diese nur in der gezeichneten Richtung befahren werden dürfen (*oneway = yes*). Das Attribut *ref = L3219*, verweist auf den üblichen Namen des Elements, welcher in diesem Fall Landesstraße 3219 lautet. Zuletzt werden die Abbiegemöglichkeiten aufgezeigt, da der Straßenabschnitt an einer

Kreuzung endet. Auf der linken Spur muss links abgebogen werden, auf der mittleren Spur muss geradeaus gefahren werden und die rechte Spur erlaubt geradeaus fahren oder rechts abbiegen.

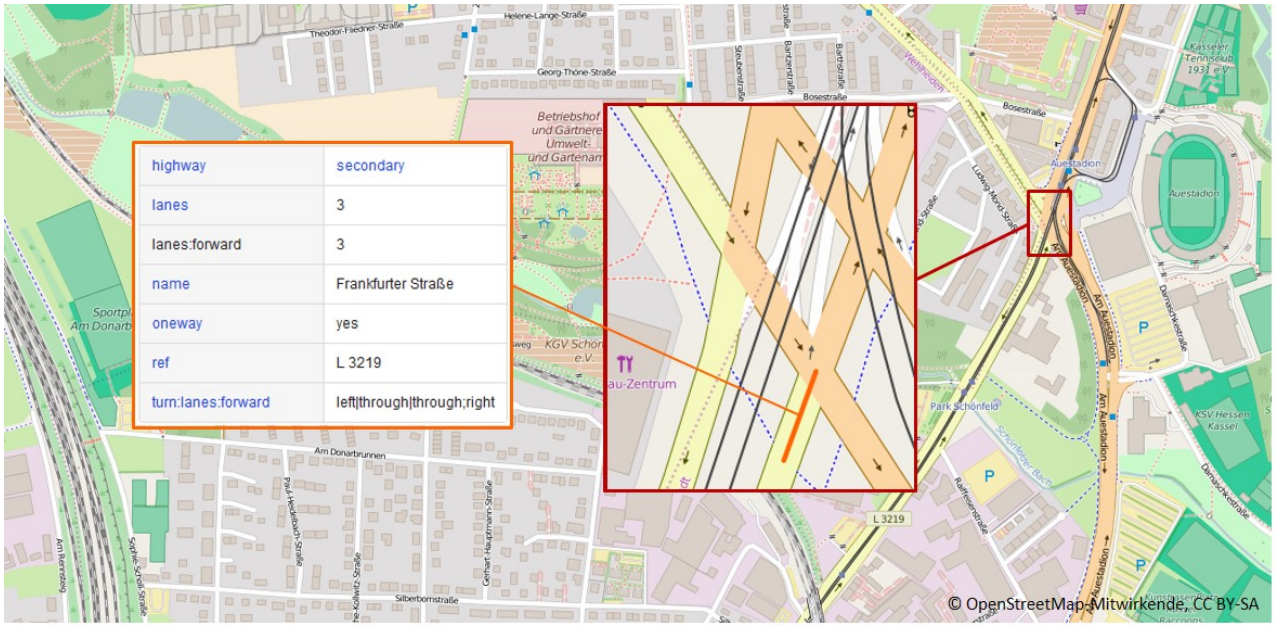

Abbildung 5.5 – Attribute eines Straßenabschnitts [OpenStreetMap-Mitwirkende, 2016]

Beim Laden der OSM-Daten in FME geht die in der Abbildung 5.5 gezeigte Struktur der Attribute verloren. Die Abbildung 5.6 beinhaltet eine vereinfachte, exemplarische Darstellung der Attribute nach dem Laden in FME. Sie zeigt eine Attributtabelle (links), in der mehrere Objekte, die alle den Schlüssel *highway* besitzen, untereinander aufgeführt werden. Weiterhin wird dargestellt, dass nur noch das jeweils erste Attribut in der Attributtabelle aufgelistet wird, was in diesem Fall dem Straßentyp entspricht, also *highway = secondary*, *highway = path* usw. Die weiteren Attribute, wie z.B. die Namen der Straßen oder die Anzahl der Spuren, werden nicht in der Attributtabelle angezeigt.
Die Attribute, die beim Laden der OSM-Daten in FME nicht in der Attributtabelle angezeigt werden, werden in Listen gespeichert. Die Abbildung 5.6 stellt zwei exemplarische Listen (mittig und rechts) dar: die grün umrandete Liste enthält die restlichen Attribute aus der ersten Zeile und die blau umrandete Liste die aus der vorletzten Zeile. Mittels der Angaben *tag(0).k* und *tag(0).v* wird die Zusammengehörigkeit der Schlüssel-Wert (**k**ey-**v**alue)-Paare dargestellt. Folglich enthält die mittige Liste vier Schlüssel-Wert-Paare (*lanes = 1, maxspeed = 50, name = Altenbaunaer Straße, oneway = yes*) und die rechte Liste zwei Schlüssel-Wert-Paare (*name = Steubenstraße, oneway = yes*).

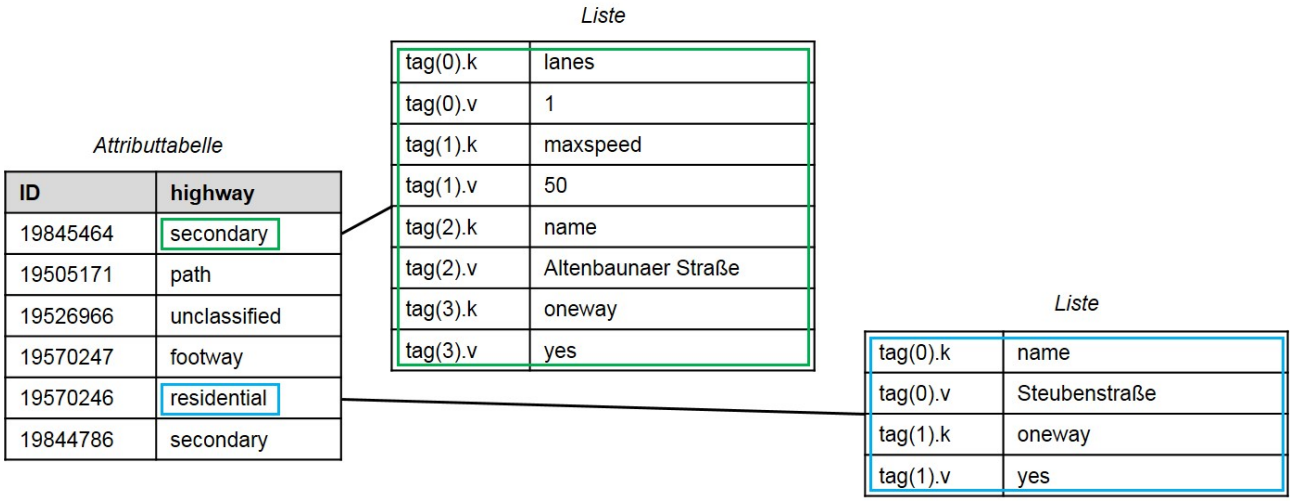

Abbildung 5.6 – Attributtabelle mit zugehörigen Listen

Mittels der zweiten Umstrukturierung wird eine vollständige Attributtabelle erzielt. Dies ist vor allem wichtig, da bei einer Umwandlung der OSM-Daten in Shape-Dateien nur die in der Attributtabelle vorhandenen Attribute übernommen werden. Somit gehen die Attribute verloren, die sich in den Listen befinden.
Abbildung 5.7 zeigt die Attributtabelle nach der zweiten Umstrukturierung. Im Gegensatz zur vorherigen Abbildung befinden sich die Attribute aus den Listen ebenfalls in der Attributtabelle. Bei fehlenden Attributen bleiben die Tabellenspalten leer; ein Beispiel dafür ist in der vierten Zeile dargestellt. Neben dem Attribut *ID = 19570247* existiert dort nur noch ein weiteres Attribut *highway = footway*, die restlichen Spalten bleiben leer.
Durch die Erzeugung einer vollständigen Attributtabelle ist der Erhalt aller Attribute bei einer Umwandlung der OSM-Daten in Shape-Dateien gegeben.

ID	highway	name	oneway	maxspeed	lanes
19845464	secondary	Altenbaunaer Straße	yes	50	1
19505171	path	Vereinsweg			
19526966	unclassified	Damaschkestraße	yes		
19570247	footway				
19570246	residential	Steubenstraße	yes		
19844786	secondary	Frankfurter Straße	yes	50	2

Abbildung 5.7 – Attributtabelle nach Umstrukturierung

Der letzte Schritt vor der Zerlegung der Schienentrasse in ihre Bauelemente, ist die Unterscheidung der Geometrien.
OSM differenziert nicht zwischen den Geometrien (Punkte, Linien, Polygonen), so wird beispielsweise unter dem Schlüssel *natural* die gesamte Vegetation zusammengefasst. Dabei ist es unwichtig, ob es sich um einen Baum handelt, der als Punkt dargestellt wird, oder um eine Wiese oder einen Wald, die als Polygone dargestellt werden.

Auch für die Umwandlung der Daten in Shape-Dateien ist die Unterscheidung der Geometrien wichtig, da ein Shape nur eine Geometrie enthalten kann. Mithilfe von FME kann zwischen den Geometrien differenziert werden.
Die Abbildung 5.8 zeigt dies beispielhaft für die Schlüssel *shop* (links) und *natural* (rechts). Der Schlüssel *shop* wird zum Markieren von Geschäften genutzt, d.h. es handelt sich in erster Linie um Gebäude oder um Räume in Gebäuden. Das große Gebäude rechts unten (im linken Bild) stellt ein Einkaufszentrum dar. Es wurde sowohl das gesamte Einkaufszentrum als Polygon, wie auch die einzelnen Geschäfte in dem Einkaufszentrum als Punkte modelliert. Für die im Schlüssel *shop* hinterlegten Geometrien bedeutet das einerseits die Trennung zwischen den Punkten und Polygonen und andererseits, dass nur die Polygone berücksichtigt werden. Da die Darstellung eines Gebäudes als Punkt wenig sinnvoll ist und in ein Gebäude integrierte Geschäftsräume für ein 3D-Modell nicht von Nutzen sind, werden nur die Polygone als Shape abgespeichert.
Beim Schlüssel *natural* wird lediglich zwischen den Geometrien Polygon und Punkt differenziert. Folglich werden alle Punktobjekte in einem Shape abgespeichert und alle Polygonobjekte in einem anderen.

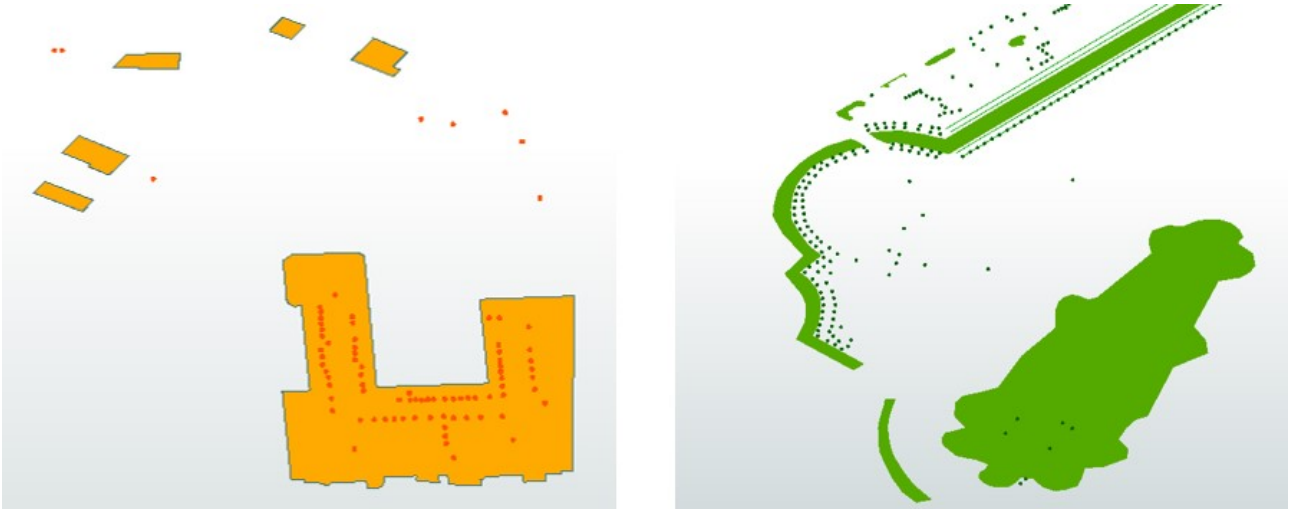

Abbildung 5.8 – Unterschiedliche Geometrien in *shop* (links) und *natural* (rechts)

Da es sich bei OSM um ein Projekt handelt, das auf der Arbeit von Freiwilligen basiert, ist die Übereinstimmung der Daten mit der Realität nicht zwingend gewährleistet. Es gibt keine konkreten Vorschriften, an die sich die Mapper zu halten haben. Es existiert jedoch eine Auflistung der Map Features, in der erklärt wird, welches Map Feature üblicherweise für welches Objekt verwendet wird, und die somit als Richtlinie dient. Letztendlich entscheidet jedoch jeder Mapper selbst, wie das Objekt dargestellt und mit welchen Attributen es benannt wird. Im Folgenden werden zwei Beispiele gezeigt, bei denen es Korrekturen bedarf.
Die Abbildung 5.9 stellt einen Ausschnitt der Teststrecke und die zum orange markierten Schienensegment zugehörige Attributtabelle dar. In der Attributtabelle wird das Map Feature *highway = cycleway* vor dem Map Feature *railway = tram* genannt. Dies hat zur Folge, dass das Schienensegment in FME fälschlicherweise als Straße dargestellt wird.

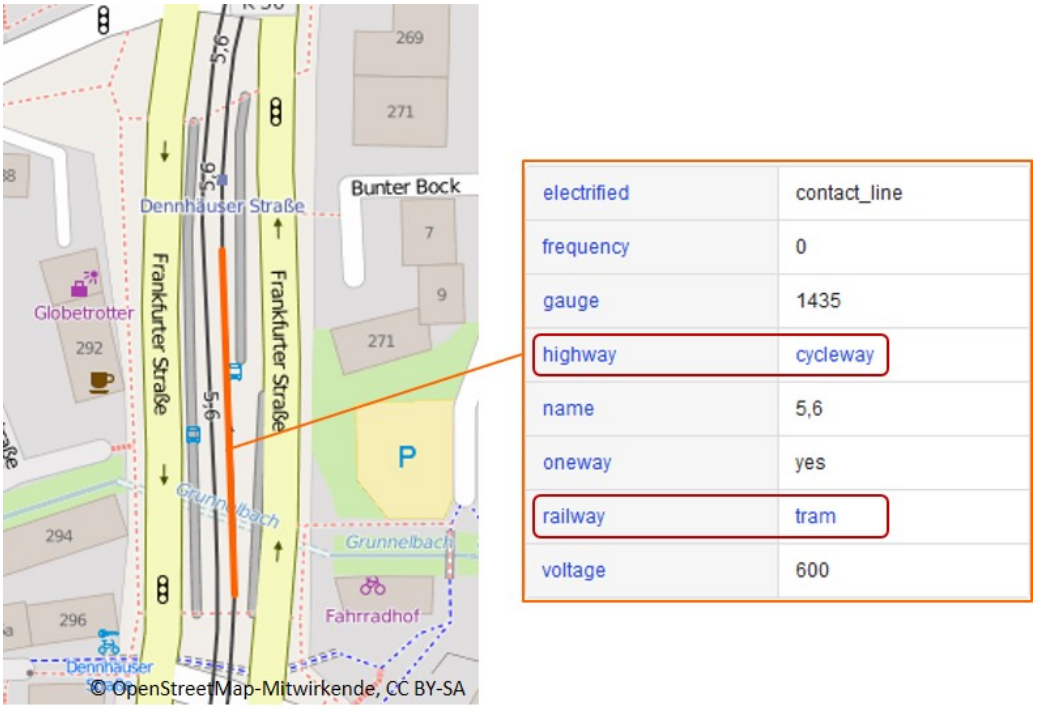

Abbildung 5.9 – Schienensegment mit zugehöriger Attributtabelle [OpenStreetMap-Mitwirkende, 2016]

Auf der linken Seite zeigt die Abbildung 5.10, dass das Schienensegment aus der vorherigen Abbildung und das darüber liegende Schienensegment fälschlicherweise als Straße (grau) anstatt als Schiene (rot) dargestellt werden. Die roten Linien neben der Schienentrasse stellen die Bahnsteige dar, die ursprünglich zum Schlüssel *railway* gehören. Zur Verdeutlichung ist auf der rechten Seite der Abbildung nur die Schienentrasse dargestellt, die Lücke der fehlenden Schienensegmente ist deutlich zu erkennen.

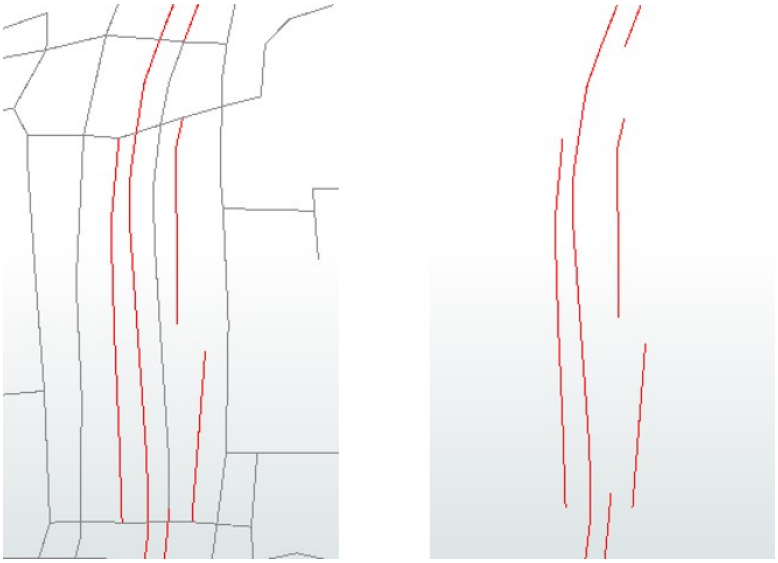

Abbildung 5.10 – Darstellung des Schienensegments als Straße (links) und fehlendes Schienensegment (rechts)

Ein weiteres Beispiel der selben Problematik ist in Abbildung 5.11 zu sehen. Es handelt sich um einen Ausschnitt der Teststrecke, der ein orange markiertes Parkhaus mit entsprechender Attributtabelle zeigt. Das Map Feature *amenity = parking* steht hierbei vor dem Map Feature *building = yes*.

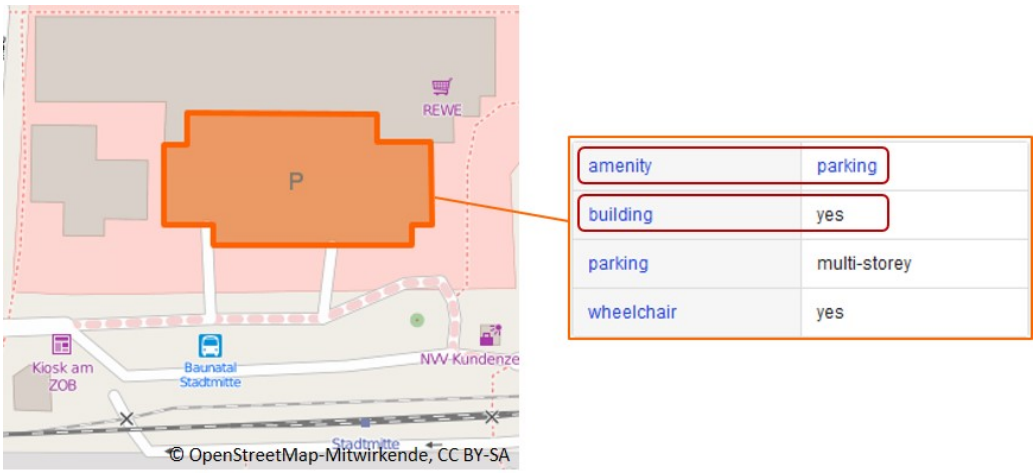

Abbildung 5.11 – Parkhaus mit zugehöriger Attributtabelle [OpenStreetMap-Mitwirkende, 2016]

Die Konsequenz ist, dass das Parkhaus bei der Darstellung der Gebäude in FME fehlt. Die Abbildung 5.12 stellt die Gebäude und die Straßenbahntrasse dar. Das rote Oval markiert die Stelle, an der das Parkhaus angezeigt werden müsste. Die in der Attributtabelle vorliegende Reihenfolge bestimmt folglich, wie die Objekte in FME dargestellt werden. Die Darstellung der zwei aufgeführten Beispiele wird in der Aufbereitung der OSM-Daten korrigiert, sodass die zwei Schienensegmente als Schienen und das Parkhaus als Gebäude angezeigt werden.

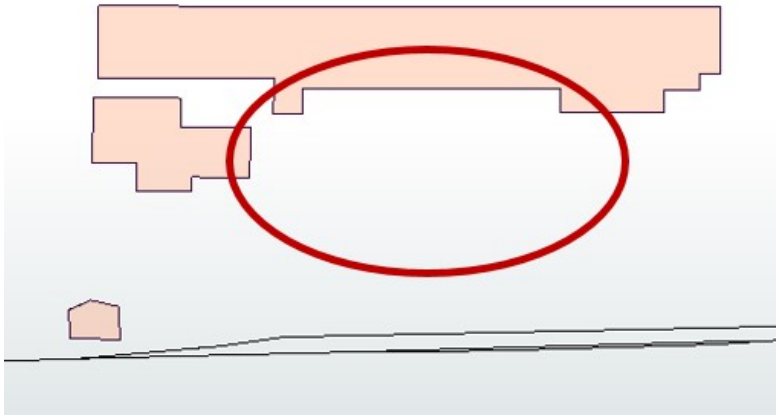

Abbildung 5.12 – Parkhaus fehlt bei der Darstellung der Gebäude [OpenStreetMap-Mitwirkende, 2016]

Alle Fehler dieser Art in einem automatisierten Prozess zu korrigieren, stellt eine Herausforderung dar, da unklar ist, ob und wie viele Fehler vorhanden sind. Die bei der Sichtung gefundenen Fehler werden in dieser Arbeit manuell korrigiert, d.h. sie werden im FME Workflow berücksichtigt. Für den Fall, das diese Fehler genau so in einem anderen Gebiet auftreten, müssen sie folglich nicht manuell korrigiert werden. Jedoch verfügt diese Arbeit nicht über einen automatischen Prozess, der Fehler in den OSM-Daten ausfindig macht und bei Bedarf korrigiert. Für den Fall, dass der in dieser Arbeit erstellte Workflow auf ein anderes Gebiet angewendet wird, muss dieses zunächst auf mögliche Korrekturen überprüft werden.

Abschließend zeigt die Tabelle 5.3 inwieweit die Anforderungen an ein 3D-Modell für einen Straßenbahnsimulator mithilfe von OSM-Daten erfüllt werden. Die rechte Spalte der Tabelle führt die Anforderungen auf, die mittlere Spalte zeigt unter welchem Schlüssel die jeweiligen Informationen in den OSM-Daten hinterlegt sind und die rechte Spalte gibt an, ob die benötigten Informationen in den OSM-Daten gegeben, teilweise gegeben oder nicht gegeben sind. Die Inhalte der Tabelle beziehen sich auf das Testgebiet dieser Arbeit. Im Folgenden werden nur ausgewählte Beispiele näher erläutert, die restlichen Punkte sind der Tabelle zu entnehmen.
Für die Darstellung der Straßenbahntrasse stellen die OSM-Daten eine geeignete Quelle dar. So sind in den OSM-Daten ausreichend Informationen enthalten, um den Verlauf der Straßenbahntrasse sowie die Spuranzahl der Schienen korrekt darzustellen. Weiterhin enthalten die OSM-Daten Informationen zu den Bahnsteigen, wobei diese nicht vollständig gegeben sind. Zur Beschaffenheit der Straßenbahntrasse liefert OSM keine Informationen.
Die Straßen sind in den OSM-Daten flächendeckend vorhanden, wobei die Anzahl der Spuren nur teilweise gegeben sind und zur Spurbreite keine Informationen enthalten sind. Die erlaubte Richtung der Befahrung ist meist gegeben. Zudem enthält OSM keine flächendeckenden Informationen zu Fußgänger- und Radwegen, diese sind teilweise in den Attributen der Straßen gegeben.
Weiterhin zeigt sich OSM als gute Datenquelle für Gebäudegrundrisse. Weitere Details wie Gebäudehöhen sind jedoch nicht vorhanden. Lediglich sind zu einigen Gebäuden die Gebäudetypen angegeben, die Aufschluss darüber geben, ob es sich bei dem Gebäude um ein Einfamilienhaus, eine

Kirche oder ein Geschäft handelt.
Die in der Tabelle aufgeführten Informationen wurden im Rahmen dieser Arbeit nur teilweise für das gesamte Testgebiet untersucht. Die Vollständigkeit der Daten ist somit nicht gewährleistet.

Anforderungen	**OSM**	**Gegeben?**
Straßenbahntrasse mit • Skelettlinie(n) • korrekter Spurzahl • korrektem Spurverlauf • Beschaffenheit	railway, public_transport	Straßenbahntrasse mit • Skelettlinien • Verlauf und Spuranzahl durch Skelettlinien gegeben • Beschaffenheit nicht gegeben • Bahnsteige teilweise gegeben • Haltestellen gegeben
Umgebende Straßen mit • korrekter Zahl der Spuren • Spurbreite • erlaubte Richtung der Befahrung (Einbahnstraße, Abbiegespuren, Parkspuren)	highway	Straßentrasse mit • Spuranzahl teilweise gegeben • Spurbreite nicht gegeben • erlaubte Richtung der Befahrung teilweise gegeben
Fußgänger- und Radwege	highway	teilweise gegeben
Kreuzungen zwischen allen Verkehrswegen und der Straßenbahntrasse	railway, highway	gegeben
Bahnübergänge	railway, highway	gegeben
Unter-/ und Überführungen	highway	gegeben
Straßenbahnsignale	X	nicht gegeben
Verkehrsampeln	highway	teilweise gegeben
Gebäude, wobei die markanten Gebäude der Realität entsprechen	building, amenity, tourism, shop, leisure	• Gebäudegrundrisse gegeben • Gebäudetypen teilweise gegeben • keine Höheninformationen, teilweise Anzahl der Stockwerke gegeben
Vegetation	natural, leisure, landuse, barrier	• Bäume • Grünflächen • Hecken
Strommasten für die Oberleitungen	X	nicht gegeben

Tabelle 5.3 – Gegenüberstellung der Anforderungen und der in den OSM-Daten enthaltenen Informationen

5.1.2 Zerlegung der Schienentrasse in Bauelemente

Nach der Reduktion und Umstrukturierung der OSM-Rohdaten wird die Schienentrasse in ihre Bauelemente zerlegt. Die Idee dahinter besteht darin, in CityEngine nicht eine einzige Regel für die gesamte Schienentrasse zu erstellen, sondern für jedes Bauelement eine eigene Regel zu schreiben. Die praktische Umsetzung der Zerlegung erfolgt in FME mit dem Workspace *Bauelemente.fmw*. Für die „Zerlegung der Schienentrasse in Bauelemente“ wird der OSM-Rohdatensatz auf das Testgebiet verkleinert (siehe Kapitel 4). Hierfür wird zuerst die in Kapitel 4 abgebildete Teststrecke erstellt. Dies geschieht in manueller Arbeit mittels der Software ArcMap der Firma ESRI. Um die neu erstellte Schienentrasse wird anschließend ein Puffer gelegt, um das in Kapitel 4 dargestellte Testgebiet zu erzeugen. Die Reduktion der OSM-Daten auf die Größe des Testgebiets erfolgt ebenfalls in ArcMap. Lediglich die Reduktion der Gebäude und Parkplätze erfolgt in FME mit dem Workspace *Zuschnitt.fmw*.
Die Zerlegung der Schienentrasse in ihre Bauelemente beschreibt die Aufgliederung des Schienenwegs in mehrere Komponenten. Für diese Arbeit wird definiert, dass sich die Schienentrasse aus den folgenden fünf Bauelementen zusammensetzt:

- „normale“ Schienen
- Weichen
- Gleiskreuzungen
- Bahnübergänge (Straße und Fußweg)
- Bahnsteige

Die Abbildung 5.13 verdeutlicht, welche Bereiche der Schienentrasse den jeweiligen Elementen zugeordnet werden. Die violett umrandeten Bereiche zeigen zwei Beispiele für das Bauelement Weiche. Das Bauelement Gleiskreuzung ist in der Abbildung rot umrandet. Das Bauelement Bahnübergang wird in zwei Kategorien unterschieden: einerseits in die Bahnübergänge, bei denen sich Straße und Schiene kreuzen (gelb umrandet), und andererseits in die Bahnübergänge, bei denen sich Fußweg und Schiene kreuzen (blau umrandet). Ein weiteres Bauelement stellen die Bahnsteige dar, in der Abbildung grün umrandet. Das Element Bahnsteig setzt sich aus der Schiene und dem parallel verlaufendem Bahnsteig zusammen. Zu den „normalen“ Schienen zählen alle Schienen, die nicht in eine der vier anderen Kategorien fallen.

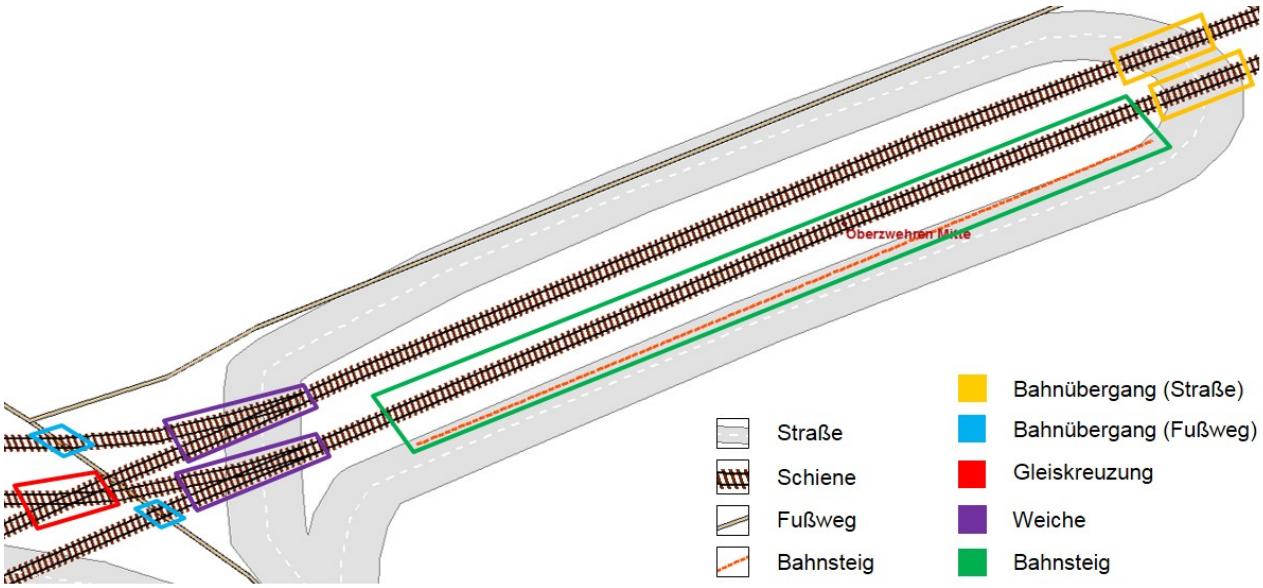

Abbildung 5.13 – Beispiel der Bauelemente

Damit in der CityEngine jedem Element automatisch die jeweilige Regel zugeordnet wird, erhält jedes Bauelement noch ein weiteres Attribut *Bauelement*. Der jeweilige Name des Elements entspricht gleichzeitig dem Wert des Attributs, für das Element Weiche also *Weiche*. Im Folgenden werden die verschiedenen Bauelemente genauer erläutert und deren Erstellung näher beschrieben.

Weichen und Gleiskreuzungen

Die Weichen und Gleiskreuzungen stellen zwei der fünf Bauelemente dar. Diese beiden Elemente werden gemeinsam behandelt, da deren Erstellung nahezu identisch ist. Das der Erzeugung der Bauelemente Weiche und Gleiskreuzung zugrunde liegende Konzept wird in der Abbildung 5.14 dargestellt.

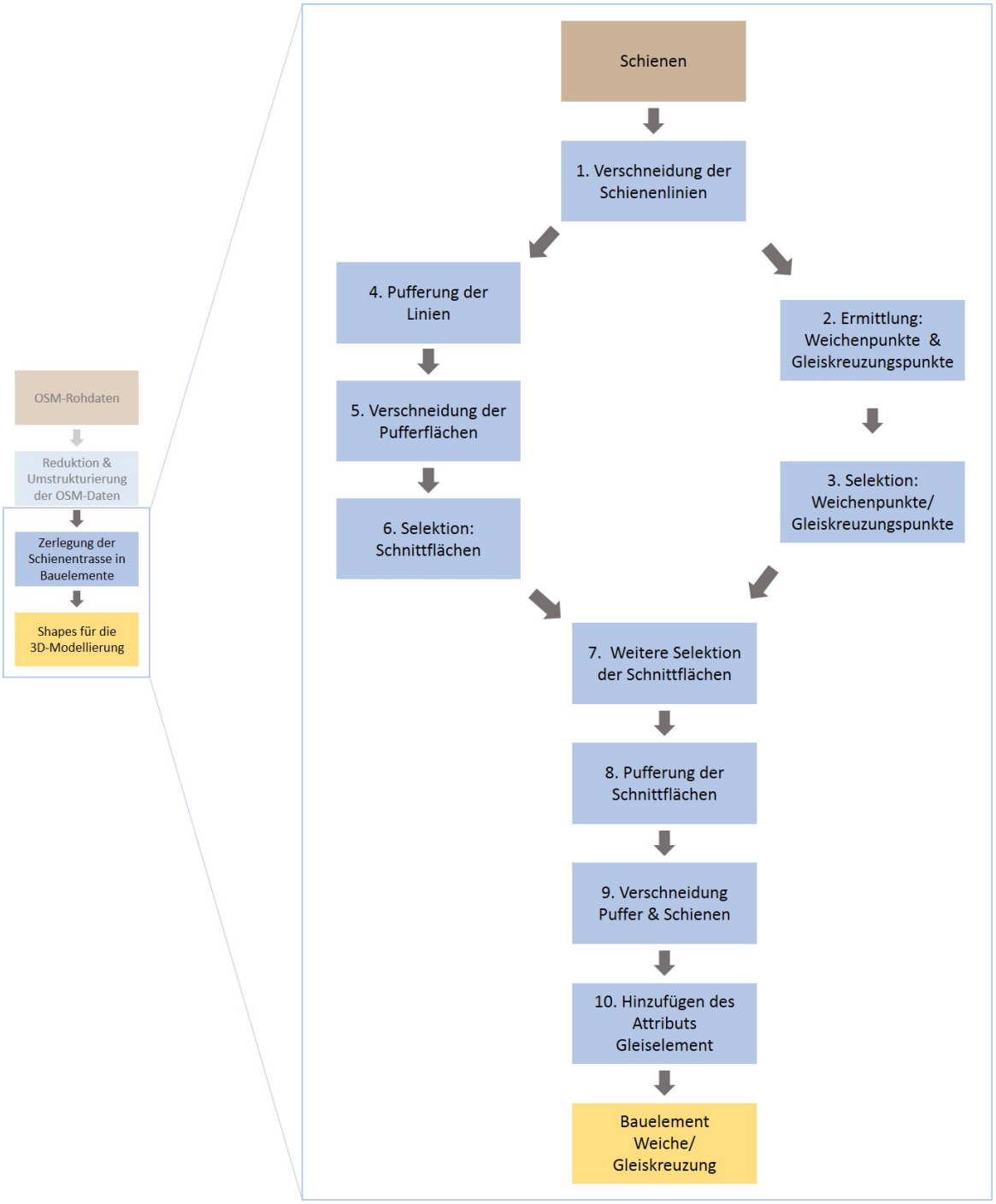

Abbildung 5.14 – Konzept zur Erstellung der Bauelemente Weiche und Gleiskreuzung

Nachfolgend wird die Erzeugung der Bauelemente Weiche und Gleiskreuzung erläutert. Um das Element Weiche zu erstellen, wird vorab definiert, welcher Bereich der Schienentrasse die Weiche darstellt. Die Abbildung 5.15 zeigt die Skelettlinie einer Weiche, die von einem Puffer umgeben ist, der die Spurbreite der Schienen kennzeichnet. Die Spurbreite beträgt für die Teststrecke 1,435 m und wird für die Aufbereitung in FME auf 1,4 m gerundet. Die Überlagerung der beiden abzweigenden Pufferflächen ist durch ein blaues Oval markiert; dieser Abschnitt stellt die Mindestgröße der Weiche

dar. Da eine Weiche jedoch nicht erst bei der Abzweigung des Gleises beginnt, wird ein etwas größerer Abschnitt für die Weiche veranschlagt; in der Abbildung durch ein rotes Oval gekennzeichnet. Zwischen der definierten Weiche (rot) und der Mindestgröße der Weiche (blau) liegen 1 m, dieser Wert unterliegt der Definition des Autors.

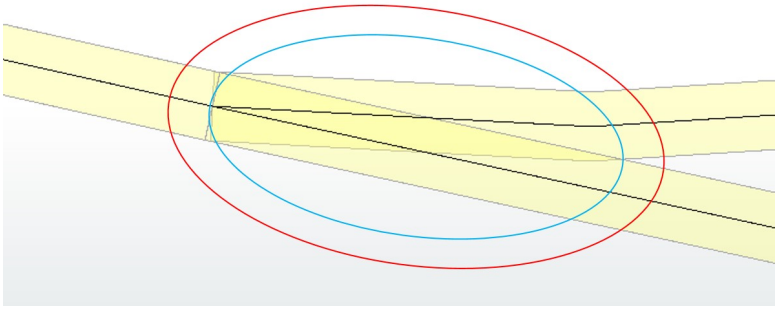

Abbildung 5.15 – Definition des Bauelements Weiche

Für die Erstellung des Bauelements Gleiskreuzung wird ebenfalls definiert, welcher Schienenabschnitt die Gleiskreuzung darstellt. Dies erfolgt analog zur Definition der Weiche und ist in Abbildung 5.16 dargestellt.

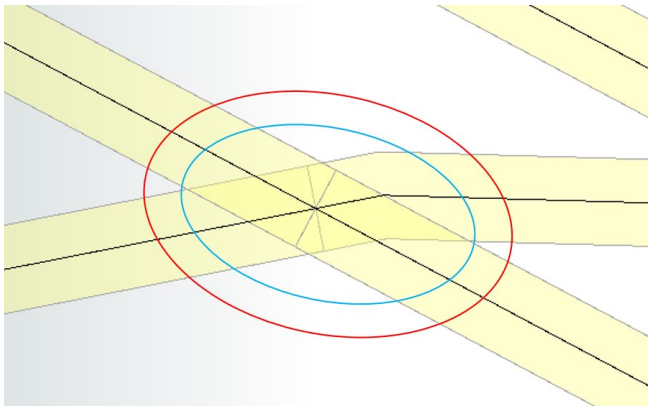

Abbildung 5.16 – Definition des Bauelements Gleiskreuzung

Im Folgenden wird das zugrundeliegende Konzept zur Erzeugung der Weiche bzw. der Gleiskreuzung näher erklärt. Die Datengrundlage der folgenden Schritte bilden die Schienen, die als Linien vorliegen.

1. Im ersten Schritt erfolgt die Verschneidung der Linien gegeneinander, d.h. die Tramlinie wird in mehrere Liniensegmente unterteilt. Einerseits an den Stellen, an denen sich Knoten auf der Linie befinden. Andererseits entstehen neue Liniensegmente an den Stellen, an denen sich eine weitere Linie abzweigt (Weiche), oder an denen sich zwei Linien kreuzen (Gleiskreuzung). An die Enden der Linien werden Punkte gesetzt. Die Abbildung 5.17 stellt dies dar. Das linke Bild zeigt die ursprüngliche Schienentrasse; es ist zu erkennen, dass die blau markierte Linie nicht durch das kreuzende Gleis beendet wird. Das rechte Bild zeigt die Schienentrasse nach der Verschneidung. Die ehemals durchgängige blaue Linie aus dem linken Bild wird im rechten

Bild in Liniensegmente aufgeteilt und endet nun an der kreuzenden Schiene. Zudem sind die Punkte an den Enden der Linien hinzugekommen.

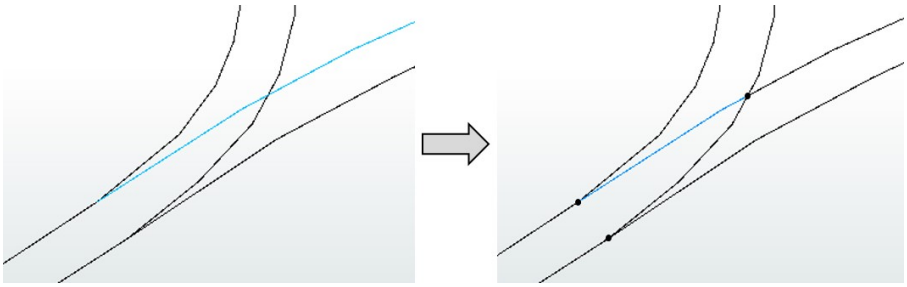

Abbildung 5.17 – Linien vor der Verschneidung (links) und nach der Verschneidung (rechts)

2. Nun wird ermittelt, wie viele Linien jeweils die eben gesetzten Punkte berühren. Dies ist für die Differenzierung zwischen den Weichen und Gleiskreuzungen notwendig. Ein Punkt auf einer Weiche wird immer von drei Linien berührt, ein Punkt auf einer Gleiskreuzung von vier Linien und ein Punkt auf einer normalen Schiene von zwei Linien, der Linie davor und der Linie danach.

3. Im dritten Schritt wird zwischen den Punkten, die drei Linien bzw. vier Linien berühren, differenziert. Für die Erstellung der Weichen sind nur noch die Punkte, die drei Linien berühren von Interesse. Diese werden für den weiteren Verlauf „Weichenpunkte“ genannt. Für die Gleiskreuzungen sind nur noch die Punkte, die vier Linien berühren, interessant. Diese werden folgend „Gleiskreuzungspunkte“ genannt.

   Nach dem ersten Schritt finden parallel zum zweiten und dritten Schritt der vierte, fünfte und sechste Schritte statt.

4. Im vierten Schritt werden die nach dem ersten Schritt entstandenen Liniensegmente gepuffert (siehe Abbildung 5.18); die Pufferdistanz beträgt dabei die halbe Spurbreite von 0,7 m. Die gesamte Pufferbreite entspricht somit der Spurbreite von 1,4 m.

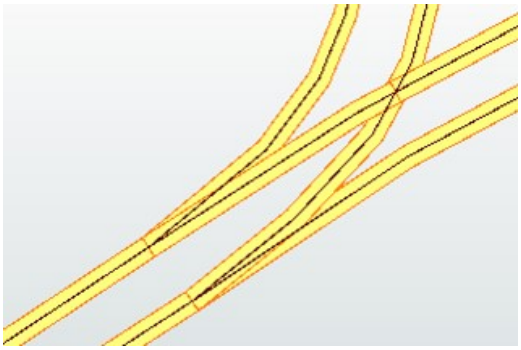

Abbildung 5.18 – Skelettlinie mit Puffer

5. Anschließend werden die Pufferflächen miteinander verschnitten. Neben den entstehenden Schnittflächen, wird zusätzlich festgehalten, wie viele Puffer sich auf einer Schnittfläche überlagern. An den Weichen und Gleiskreuzungen überlagern sich beispielsweise zwei Schnittflächen. Die Verschneidung der Puffer wird vorgenommen, da die zwei sich überlagernden

Pufferflächen an den Weichen und Gleiskreuzungen die Basis für die Erstellung der beiden Bauelemente bilden.

6. Nun werden alle Schnittflächen, auf denen sich zwei Puffer überlagern, ausgewählt. Das Ergebnis ist in Abbildung 5.19 dargestellt und zeigt die Schnittflächen exemplarisch für zwei Weichen und eine Gleiskreuzung.

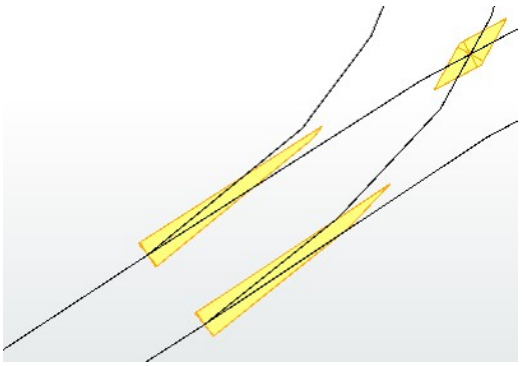

Abbildung 5.19 – Schnittflächen an zwei Weichen und einer Gleiskreuzung

Im siebten Schritt werden die Ergebnisse aus dem dritten und dem sechsten Schritt wieder zusammengeführt.

7. Ab diesem Schritt werden die Weichen und Gleiskreuzungen getrennt. Für die Weichen werden alle Schnittflächen ausgewählt, die einen Weichenpunkt berühren. Dies erfolgt ebenfalls für die Gleiskreuzungen, d.h. es werden alle Schnittflächen ausgewählt, die einen Gleiskreuzungspunkt berühren. Das Resultat bilden einerseits alle Schnittflächen, die an einer Weiche liegen (vgl. Abbildung 5.20 links), und andererseits alle Schnittflächen, die an einer Gleiskreuzung liegen (vgl. Abbildung 5.20 rechts).

Abbildung 5.20 – Auswahl der an den Weichen (links) bzw. Gleiskreuzungen (rechts) liegenden Schnittflächen

8. Um die Weichen (siehe Abbildung 5.15) und Gleiskreuzungen (siehe Abbildung 5.16) realitätsnäher abzubilden, werden die Schnittflächen um je einen 1 m gepuffert (siehe Abbildung 5.21).

Abbildung 5.21 – gepufferte Schnittflächen: Weichen (links) und Gleiskreuzungen (rechts)

9. Nun erfolgt die Verschneidung der im vorherigen Schritt erstellten Pufferflächen mit der Schienentrasse. Das Resultat ist anhand von Beispielen in der Abbildung 5.22 dargestellt und zeigt die Weichenelemente in blau und die Gleiskreuzungselemente in rot.

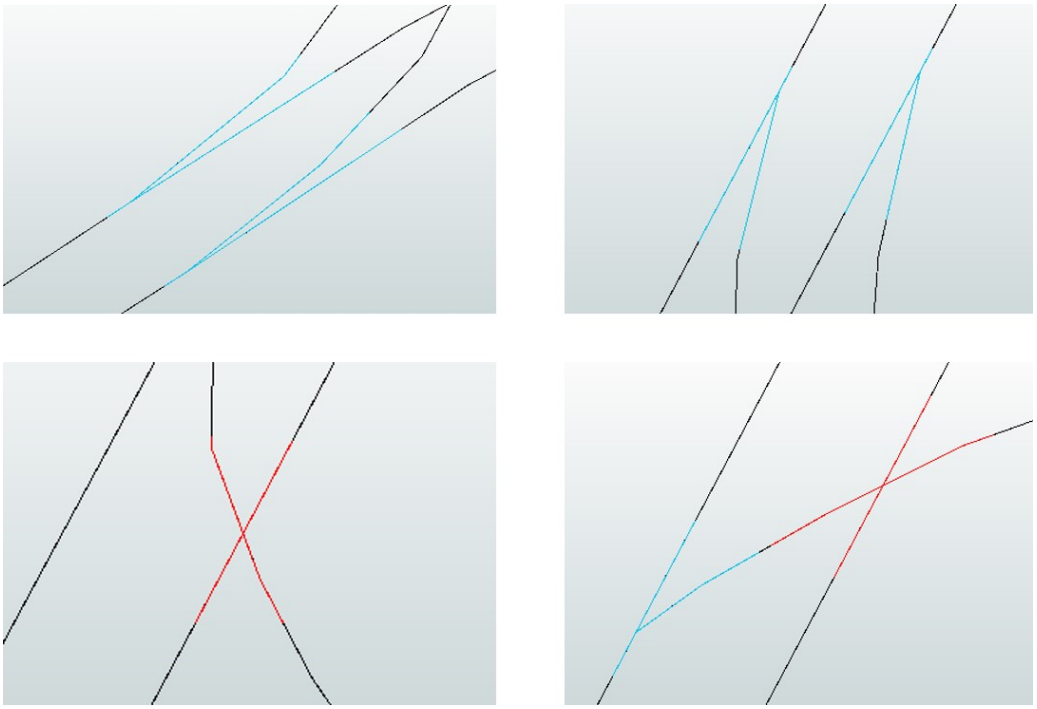

Abbildung 5.22 – Bauelemente Weiche (blau) und Gleiskreuzung (rot) mit Schienentrasse

10. Abschließend wird die Attributtabelle um das Attribut *Bauelement* mit dem Wert *Weiche* für die Weichen und mit dem Wert *Gleiskreuzung* für die Gleiskreuzungen erweitert.

Bahnübergänge

Im Folgenden wird die Erstellung des Bauelements Bahnübergang behandelt, wobei es zwei Typen von Bahnübergängen zu unterscheiden gilt. Einerseits die Bahnübergänge, die durch die Kreuzung von Schiene und Straße entstehen, und andererseits die Bahnübergänge, die durch die Kreuzung

von Schiene und Fußweg entstehen. Die Abbildung 5.23 zeigt das Konzept, das der Erstellung des Bauelements Bahnübergang zugrunde liegt.

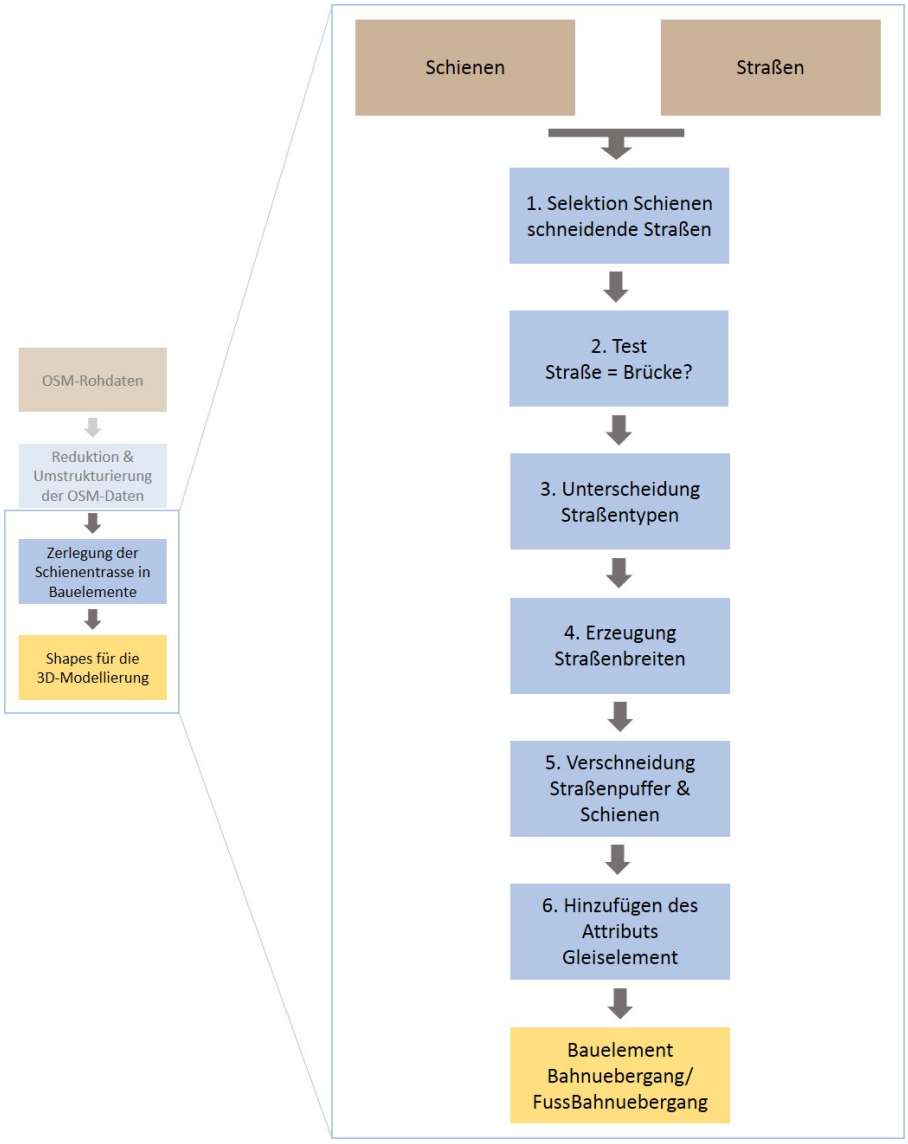

Abbildung 5.23 – Konzept zur Erstellung des Bauelements Bahnuebergang und FussBahnuebergang

Die Erstellung der Bahnübergänge ist für beide Typen identisch, somit basieren beide auf dem dargestellten Konzept. Je nachdem, welcher Typ generiert wird, dienen entweder die Schienen und Straßen als Datengrundlage oder die Schienen und Fußwege. Die Erzeugung der beiden Elemente unterscheidet sich lediglich durch deren Breite, d.h. die im Folgenden erzeugten Puffer weisen unterschiedliche Pufferdistanzen auf. Nachfolgend wird das Konzept am Beispiel der Bahnübergänge Schiene-Straße näher erläutert. Als Datengrundlage liegen die Schienen und Straßen vor.

1. Im ersten Schritt erfolgt eine Selektion der Straßen. Es werden alle Straßen bzw. Straßensegmente ausgewählt, die die Schienentrasse schneiden. Dies ist sinnvoll, da die folgenden Schritte nur noch mit dem selektierten und nicht mehr mit dem vollständigen Straßensatz ausgeführt werden müssen.

2. Im zweiten Schritt wird getestet, ob es sich bei der die Schienen schneidenden Straße um eine Brücke handelt. Falls es sich bei der Straße um eine Brücke handelt, wird auch diese für den weiteren Verlauf außer Acht gelassen.

3. Nun erfolgt die Unterscheidung der verschiedenen Straßentypen. Die Straßen werden in vier Straßentypen unterschieden: in Autobahnen, Bundesstraßen, Landesstraßen und Nebenstraßen. Diese Unterteilung folgt der Wahl des Autors. Die nachfolgende Tabelle 5.4 zeigt, welche Map Features den jeweiligen Straßentypen zugeordnet werden.

Straßentyp	**Map Features** (highway = …)
Autobahn	motorway, motorway_link
Bundesstraße	primary, primary_link
Landesstraße	secondary, secondary_link
Nebenstraße	tertiary, unclassified, residential, livingstreet, service

Tabelle 5.4 – Straßentypen mit den zugehörigen Map Features

4. Im vierten Schritt erfolgt die Erzeugung der Straßenbreiten. Hierfür wird jedem Straßentyp eine Spurbreite zugeordnet. Die Spurbreiten sind repräsentativ für den jeweiligen Straßentyp und unterliegen in dieser Arbeit der Wahl des Autors. Um die Straßenbreite zu erhalten, wird die Spurbreite mit der Anzahl der Spuren multipliziert. Da jedoch nicht zu jedem Straßensegment eine Spuranzahl vorhanden ist, wird zusätzlich für jeden Straßentyp ein Standardwert definiert. Der Standardwert folgt der Wahl des Autors und orientiert sich dabei an den in dieser Arbeit zugrundeliegenden Daten des Testgebiets. Die Tabelle 5.5 zeigt die Spurbreiten und die Standardwerte für die Anzahl der Spuren der verschiedenen Straßentypen.

Straßentyp	**Spurbreite [m]**	**Standardwert für Anzahl der Spuren**
Autobahn	3,75	3
Bundesstraße	3,50	2
Landesstraße	3,00	1
Nebenstraße	2,75	1

Tabelle 5.5 – Straßentypen mit den zugehörigen Spurbreiten und Standardwerten für die Anzahl der Spuren

Die Abbildung 5.24 zeigt eine exemplarische Darstellung dreier Landesstraßen ohne gepufferte Straßenbreiten (links) und mit gepufferten Straßenbreiten (rechts).

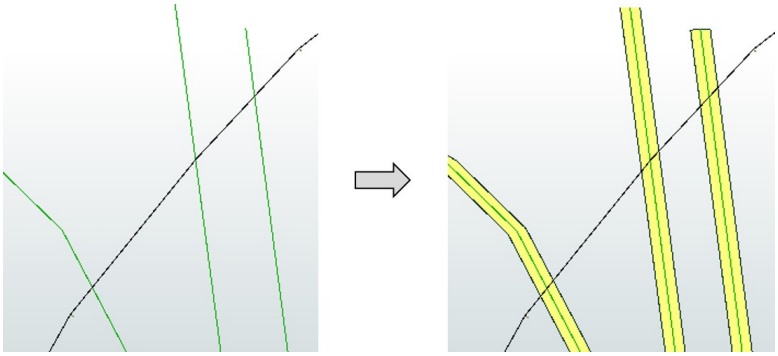

Abbildung 5.24 – Straßen ohne gepufferte Straßenbreite (links) und mit Straßenpuffer (rechts)

5. Anschließend werden die erzeugten Straßenpuffer bzw. -breiten mit der Schienentrasse verschnitten, um somit das Bauelement Bahnübergang zu erhalten. Die Abbildung 5.25 stellt das Bauelement Bahnübergang dar; links ist zudem die Straßenbreite eingeblendet.

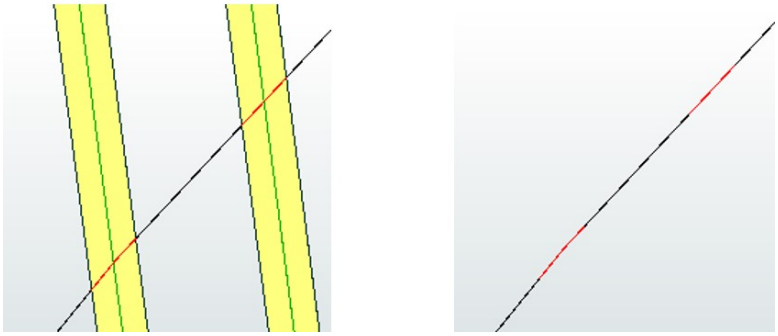

Abbildung 5.25 – Bauelement Bahnübergang mit hinterlegter Straßenbreite (links) und ohne Straßenbreite (rechts)

6. Im letzten Schritt erhalten die Bahnübergänge das zusätzliche Attribut *Bauelement* mit dem Wert *Bahnuebergang*.

Die Erstellung der Bahnübergänge, an denen sich Schienen und Fußwege kreuzen, erfolgt analog. Im dritten Schritt werden jedoch nicht die Straßentypen Autobahn, Bundesstraße, Landesstraße und Nebenstraße unterschieden, sondern die zwei Typen Fußweg und Fußgängerzone. Die Tabelle 5.6 zeigt die Zuordnung der Map Features zu den jeweiligen Typen und die zugehörigen Breiten. Die Breiten werden als Standardwerte definiert, da die Daten keine Informationen enthalten. Auch hier unterliegt die Definition des Standardwerts der Wahl der Autors. Wie die Bahnübergänge, an denen sich Schiene und Straßen kreuzen, erhalten auch die Bahnübergänge, an denen sich Schiene und Fußweg kreuzen, das Attribut Bauelement; der Attributwert ist hier *FussBahnuebergang*.

Fußwegtyp	Map Features	Breite [m]
Fußweg	bicycle = designated, foot = designated, highway = footway	2,50
Fußgängerzone	highway = pedestrian	6,00

Tabelle 5.6 – Fußwegtypen mit den zugehörigen Map Features und Breiten

Bahnsteige

Nachfolgend wird die Erstellung des Bauelements Bahnsteig erläutert. In den OSM-Daten liegen zwei verschiedene Arten von Bahnsteigen vor: einerseits die Bahnsteige, die durch eine „einfache“ Linie repräsentiert werden (vgl. Abbildung 5.26 unten); hier ist abgesehen von der Lage und der Länge der Bahnsteige nichts weiter bekannt. Andererseits gibt es Bahnsteige, die mittels einer geschlossenen Linie (vgl. Abbildung 5.26 oben), die den Umriss des Bahnsteigs kennzeichnet, etwas detailgenauer dargestellt werden.

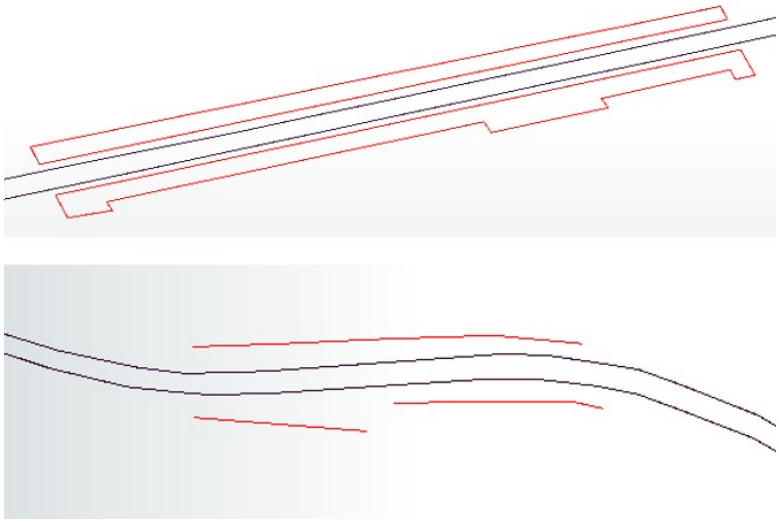

Abbildung 5.26 – Darstellung eines Bahnsteig als geschlossenen Linie (oben) und als einfache Linie (unten)

Zwischen diesen beiden Darstellungsarten wird im Folgenden unterschieden. Die geschlossenen Linien werden in Polygone umgewandelt und in CityEngine als eigene Objekte modelliert. Die einfachen Linien werden zusammen mit der benachbarten Schienentrasse modelliert. Die gemeinsame Modellierung hat zudem den Vorteil, dass so auch in den Daten fehlende Bahnsteigobjekte ersetzt werden könnten. Die Abbildung 5.27 zeigt das Konzept zur Erstellung des Bauelements, welches im Anschluss genauer beschrieben wird. Als Datengrundlage dienen die Schienen und die Bahnsteige.

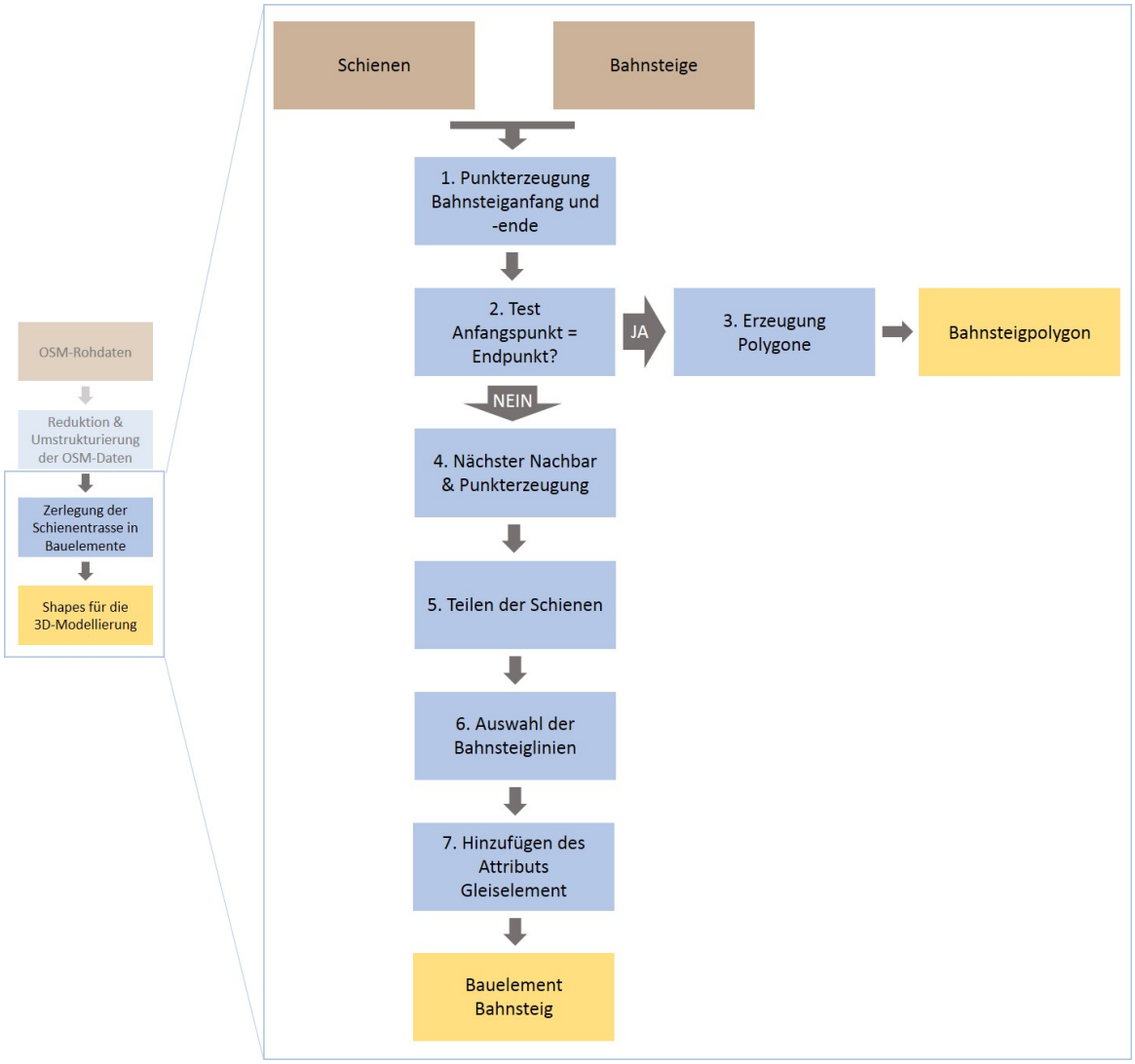

Abbildung 5.27 – Konzept zur Erstellung des Bauelements Bahnsteig

1. Im ersten Schritt wird für jeden Bahnsteig, jeweils am Bahnsteiganfang und am Bahnsteigende, ein Punkt erzeugt. Die Anfangs- und Endpunkte erhalten dabei die ID des zugehörigen Bahnsteigs.

2. Um zu unterscheiden, ob es sich bei den Bahnsteigen um eine einfache Linie oder eine geschlossene Linie handelt, wird im zweiten Schritt getestet, ob der Anfangspunkt die gleichen Koordinaten wie der Endpunkt aufweist. Bei den einfachen Linien ist dies nicht der Fall, wohingegen die Anfangs- und Endpunkte der geschlossenen Linien identisch sind. Nach dieser Differenzierung werden die Linien in den folgenden Schritten separat behandelt.

3. Fällt der Test aus Schritt zwei positiv aus (d.h. Anfangspunkt gleich Endpunkt), werden im dritten Schritt alle geschlossenen Linien in Polygone transformiert. Diese werden in CityEngine als eigenständige Objekte behandelt und nicht gemeinsam mit der Schienentrasse modelliert. Die Abbildung 5.28 zeigt den Ausgangszustand von zwei geschlossenen Linien (links) und das Ergebnis nach der Transformation zu Polygonen (rechts).

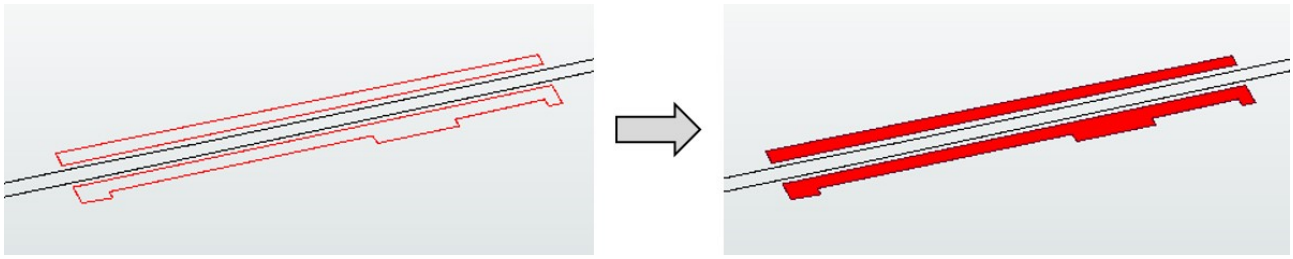

Abbildung 5.28 – Von der geschlossenen Bahnsteiglinie (links) zum Bahnsteigpolygon (rechts)

4. Wird die Frage in Schritt zwei mit „NEIN“ beantwortet, wird mit Schritt vier fortgefahren, indem die „nächsten Nachbarn“ der Punkte auf den Schienen gesucht werden. Unter dem „nächsten Nachbarn“ ist der Punkt zu verstehen, der den geringsten Abstand zum Ausgangspunkt aufweist. Nach der Identifizierung der nächsten Nachbarn werden an diesen Stellen wiederum Punkte erzeugt. Die Abbildung 5.29 zeigt zwei Bahnsteiglinien mit ihren Anfangs- und Endpunkten sowie die Punkte an den Stellen der nächsten Nachbarn.

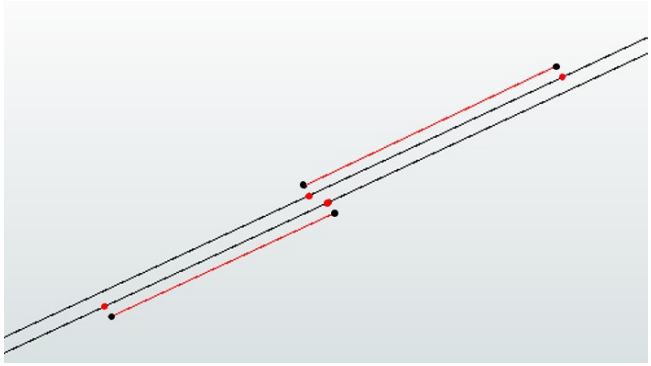

Abbildung 5.29 – Anfangs- und Endpunkte der Bahnsteiglinien und „nächste Nachbarn“- Punkte auf den Schienen

5. Im fünften Schritt werden die Schienen an den eben erstellten Punkten geteilt.

6. Anschließend erfolgt die Auswahl der Linien, die das Bauelement Bahnsteig darstellen. Hierfür müssen zwei Kriterien erfüllt werden. Das erste Kriterium besagt, dass am Anfang und am Ende der Linie ein „nächster Nachbar“-Punkt liegen muss. Das zweite Kriterium ist, dass die beiden Punkte die selbe ID aufweisen müssen (vgl. Schritt 1). Die Abbildung 5.30 zeigt anhand von zwei Beispielen das Resultat nach Schritt sechs und somit die finalen Bauelemente Bahnsteig. Die ursprünglichen Bahnsteiglinien sind dabei in rot dargestellt. Die blauen Linien auf der Schienentrasse stellen das Bauelement Bahnsteig dar.

7. Im letzten Schritt erfolgt die Erweiterung der Attributtabelle um das Attribut *Bauelement* mit dem Wert *Bahnsteig*.

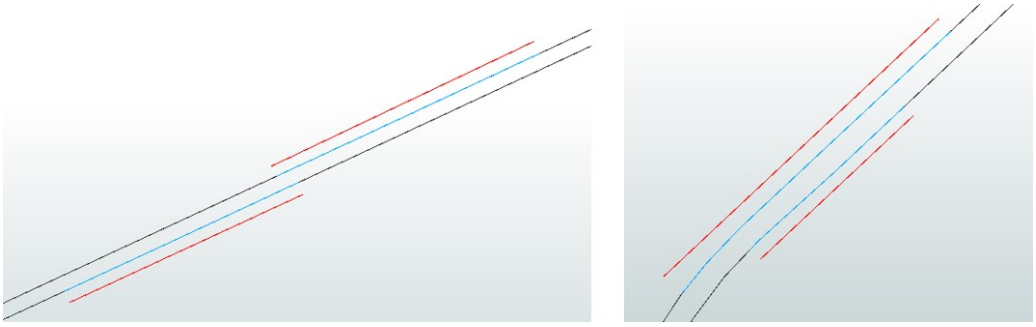

Abbildung 5.30 – Bauelement Bahnsteig anhand von zwei Beispielen (blau)

Die im Rahmen dieser Arbeit erstellte Methode zur Erzeugung der Bauelemente Bahnsteig ist für folgende Gegebenheiten erfolgreich:

- Wenn es sich um eine zweigleisige Schienentrasse handelt.
- Wenn es sich um eine eingleisige Schienentrasse handelt, die Bahnsteige jedoch nicht direkt gegenüber, sondern versetzt zueinander liegen. Die Versetzung muss dabei so groß sein, dass das Ende des ersten Bahnsteigs vor dem Beginn des zweiten Bahnsteigs liegt.

Für den Fall, dass es sich um eine eingleisige Strecke handelt und sich die Bahnsteige gegenüber liegen sowie deren Versetzung nicht ausreichend groß ist, kann das im sechsten Schritt genannte zweite Kriterium nicht erfüllt werden, da die Anfangs- und Endpunkte nicht die selbe ID aufweisen. Für diesen Fall erbringt die Methode zur Erzeugung des Bauelements Bahnsteig kein Ergebnis.

„Normale“ Schienen

Unter den normalen Schienen sind diejenigen Schienen zu verstehen, die nicht in eine der vier Kategorien Weiche, Gleiskreuzung, Bahnübergang oder Bahnsteig fallen. Um die normalen Schienen zu identifizieren, wird die Schienentrasse mit den eben genannten Bauelementen verschnitten. Bei der Verschneidung fallen die Schienenabschnitte weg, die von anderen Bauelementen überlagert werden. Somit bleiben die Schienenabschnitte übrig, die zum Bauelement normale Schienen gehören. Die Attributtabelle wird um das Attribut *Bauelement* mit dem Wert *Schiene* erweitert. Die Abbildung 5.31 zeigt einen exemplarischen Ausschnitt der Bauelemente Schiene.

Abbildung 5.31 – Bauelement Schiene

Nach der Erstellung der einzelnen Bauelemente, gilt es ein weiteres Problem zu beheben. Die verschiedenen Bauelemente reihen sich nicht wie gewünscht aneinander, sondern überlappen sich teilweise. In dem Testgebiet stellen die Weichen und Gleiskreuzungen die kritischen Bauelemente dar. Diese überschneiden sich mit den Bahnübergängen und Fußbahnübergängen. Eine Überlappung zweier Bauelemente ist ungünstig, da nach der Modellierung zwei Modelle übereinander liegen, was unerwünscht ist. Die Abbildung 5.32 zeigt auf der linken Seite die Schienentrasse mit zwei Weichen (orange) und einer Gleiskreuzung (rot) und auf der rechten Seite die zwei Weichen und die Gleiskreuzung mit eingeblendeten Bahnübergängen (grün) und Fußbahnübergängen (blau). Eine starke Überlappung ist deutlich zu erkennen, wobei der rechte Bahnübergang gänzlich im Bereich der Weiche liegt.

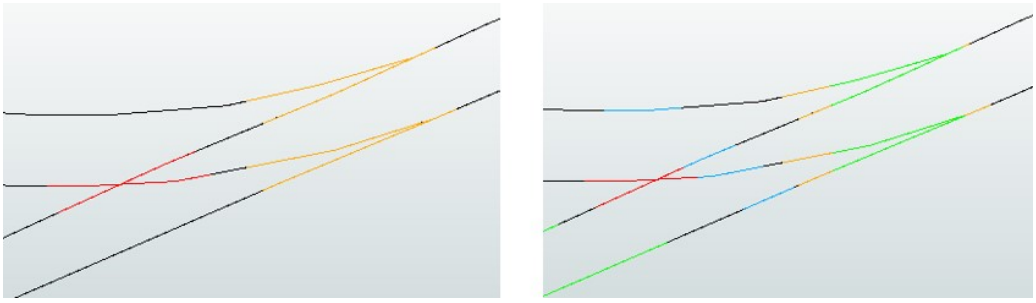

Abbildung 5.32 – Überlappung der Weichen (orange) bzw. Gleiskreuzungen (rot) mit den Bahnübergängen (grün) bzw. Fussbahnübergängen (blau)

Die Abbildung 5.33 zeigt das Konzept zur Lösung der überlappenden Bauelemente auf. Die praktische Umsetzung erfolgt im FME Workspace *Bauelemente.fmw*. Als Ausgangsbasis dienen alle erstellten Bahnübergänge und Fußbahnübergänge sowie alle Weichen und Gleiskreuzungen.

1. Im ersten Schritt werden die Elemente herausgefiltert, die von der Problematik der Überlappung betroffen sind. Die betroffenen Bahnübergänge und Fußbahnübergänge folgen dem linken „JA“-Pfeil und werden in Schritt zwei und drei weiterverarbeitet. Die betroffenen Weichen und Gleiskreuzungen folgen dem rechten „JA“-Pfeil und werden in den Schritten vier und fünf weiterverarbeitet. Die Elemente, die nicht betroffen sind, werden direkt in die Zielelemente weitergeleitet und folgen in der Abbildung dem „NEIN“-Pfeil. Die Schritte zwei und drei sowie die Schritte vier und fünf laufen in FME parallel ab.
2. Die Bahnübergänge und Fußbahnübergänge werden nun um die Länge der Überlappung gekürzt. Somit schließen die Bahnübergänge bzw. Fußbahnübergänge direkt an die Weichen bzw. Gleiskreuzungen an.
3. Im dritten und letzten Schritt der Bahnübergänge und Fußbahnübergänge werden diese differenziert, sodass die Elemente wieder in ihrer Ausgangsform vorliegen.
4. Die Abschnitte der Weichen und Gleiskreuzungen, die sich mit den Bahnübergängen und Fußbahnübergängen überlappen, erhalten neue Attributwerte *Weiche_Asphalt* bzw. *Gleiskreuzung_Asphalt*. Diese werden im Attribut *Bauelement* abgespeichert und ersetzen die bisherigen Attributwerte *Weiche* bzw. *Gleiskreuzung*.

5. Im fünften und letzten Schritt der Weichen und Gleiskreuzungen werden auch diese wieder differenziert.

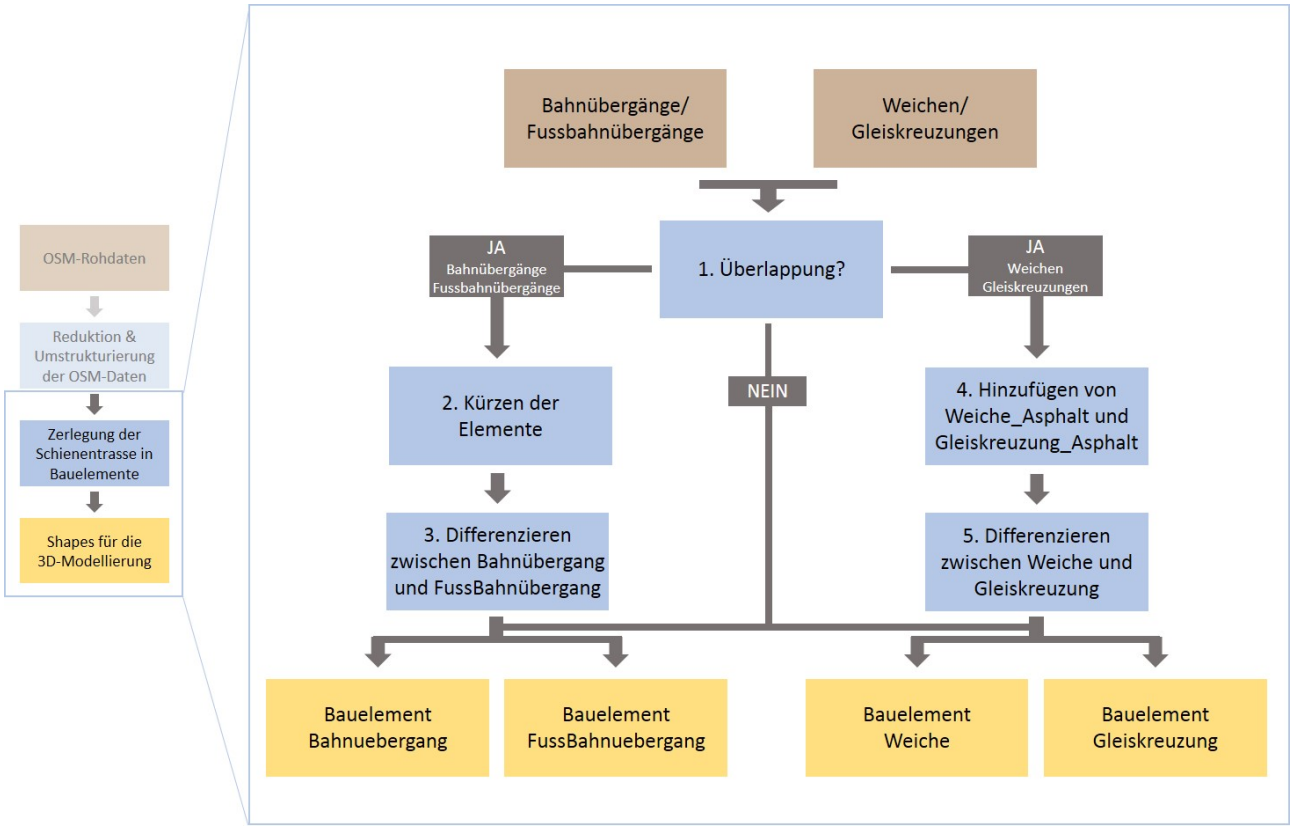

Abbildung 5.33 – Konzept zur Lösung der überlappenden Bauelemente

Das Ergebnis ist in Abbildung 5.34 dargestellt. Es ist zu erkennen, dass die Bahnübergänge, die in Abbildung 5.32 über die Weichen laufen, entfernt wurden. Die grünen Abschnitte der Weiche haben den Attributwert *Weiche_Asphalt* erhalten. Dem gelben Weichenabschnitt ist der Attributwert *Weiche* geblieben. Auch einige Abschnitte der Gleiskreuzung haben ihre alten Attribuwerte *Gleiskreuzung* behalten (rot) und einige haben den neuen Attributwert *Gleiskreuzung_Asphalt* (dunkelblau) erhalten. Einige Abschnitte des Fußbahnübergangs (blau) wurden deutlich gekürzt.

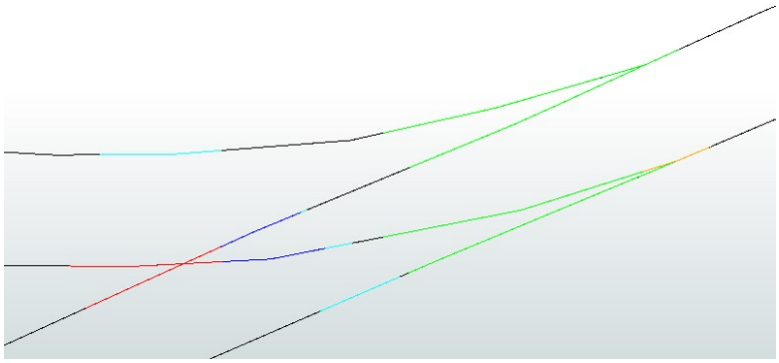

Abbildung 5.34 – Weichen und Gleiskreuzungen ohne überlappende Bahnübergänge bzw. Fussbahnübergänge

5.1.3 Manuelle Korrektur der Bahnsteige

Nach der Datenaufbereitung besteht die Möglichkeit, manuelle Korrekturen an den in der Datenaufbereitung erzeugten Shapes vorzunehmen. Da der Fokus in der Modellierung mit CityEngine auf der Schienentrasse liegt, werden in diesem Kapitel nur Elemente berücksichtigt, die für die Darstellung der Schienentrasse von Bedeutung sind.
In dieser Arbeit werden an zwei Bahnsteigpolygonen Korrekturen vorgenommen. In den Abbildungen 5.35 und 5.36 sind auf der linken Seite der Abbildung zwei Bahnsteige zu erkennen, die sich fälschlicherweise mit dem Verlauf der Schienen überlappen. Diese Fehler in den OSM-Daten werden mithilfe der Software ArcMap manuell korrigiert. Das Ergebnis ist in den nachfolgenden Abbildungen auf der linken Seite dargestellt.

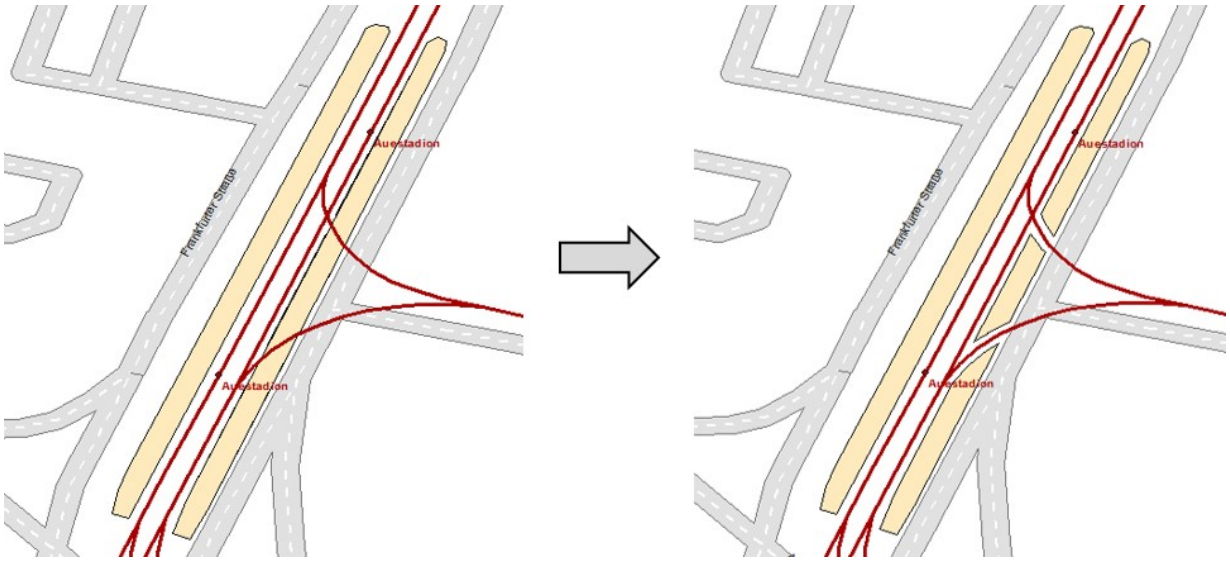

Abbildung 5.35 – Durchgängiger Bahnsteig (links) und korrigierter Bahnsteig (rechts) an der Haltestelle Auestadion

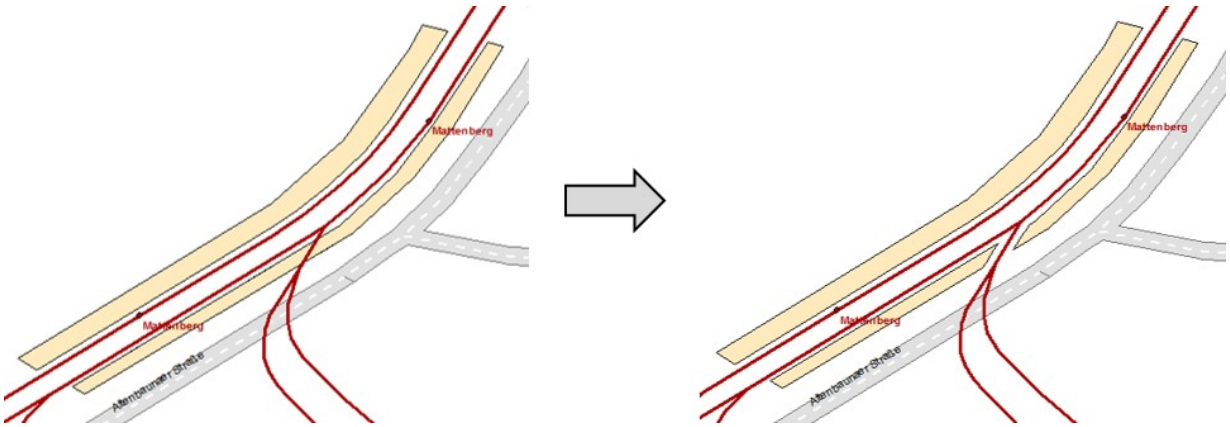

Abbildung 5.36 – Durchgängiger Bahnsteig (links) und korrigierter Bahnsteig (rechts) an der Haltestelle Mattenberg

5.2 Modellierung in CityEngine

Nach der Datenaufbereitung mittels FME folgt die Erstellung der CGA-Regeln sowie die Modellierung mithilfe der Software CityEngine. Da es im Rahmen dieser Arbeit nicht möglich ist, ein gesamtes 3D-Modell für einen Straßenbahnsimulator zu erzeugen, liegt das Hauptaugenmerk auf der automatisierten Generierung der Schienentrasse.
Die Abbildung 5.37 zeigt das Konzept für die Modellierung in CityEngine. Als Eingangsinformationen dienen die in der Datenaufbereitung erstellten Shapes.
Wie bereits erwähnt, liegt der Fokus auf der Erzeugung der Schienentrasse. Somit werden in der CityEngine für die folgenden Elemente CGA-Regeln erstellt:

- normale Schienen
- Bahnübergänge und Fußbahnübergänge
- Weichen und Gleiskreuzungen
- Bahnsteige und Bahnsteigpolygone

Für die Modellierung der normalen Schienen wird die CGA-Regel *Schiene.cga* erstellt. Die Bahnübergänge und Fußbahnübergänge werden mittels der gemeinsamen CGA-Regel *Bahnuebergang.cga* modelliert. Weiterhin werden auch die Elemente Weiche und Gleiskreuzung in einer gemeinsamen CGA-Regel modelliert, deren Name *Weiche.cga* lautet. Die Modellierung der Bahnsteige erfolgt mithilfe der CGA-Regel *Bahnsteig.cga*, die Bahnsteigpolygone werden mit der CGA-Regel *Bahnsteigpolygone.cga* modelliert. Durch die Anwendung der CGA-Regeln auf die entsprechenden Shapes wird in der CityEngine das 3D-Modell erzeugt.
Anschließend ist ein Export des Modells möglich, wobei CityEngine für den Export zehn verschiedene Formate anbietet, unter anderem COLLADA (.dae) oder ESRI FileGDB (.gdb). Nach dem Export besteht die Option, manuelle Korrekturen vorzunehmen. Ein Export findet im Rahmen dieser Arbeit nicht statt.
In den nachfolgenden Kapiteln wird jeweils der Aufbau der CGA-Regel erläutert und die resultierenden Modelle dargestellt.

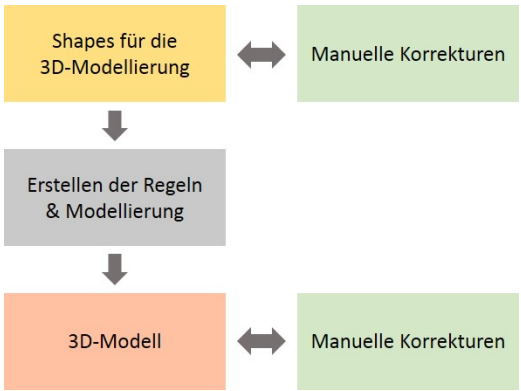

Abbildung 5.37 – Konzept für die Modellierung in CityEngine

5.2.1 „Normale“ Schiene

Die normalen Schienen stellen den Großteil der Schienentrasse dar. Die Ausgangsbasis bilden die Skelettlinien der Schienen. Diese können anhand ihres Attributs *Bauelement = Schiene* identifiziert werden. Beim Laden der Schienen in CityEngine ist es wichtig, die Option „Run Graph Cleanup Tool after Import“ zu deaktivieren, damit der Verlauf der Schienen korrekt dargestellt wird. Die Schienentrasse wird durch Kanten und Knoten dargestellt. Beim Laden von Daten in die CityEngine ist es von Vorteil, bereits zu Beginn die korrekten Einstellungen zu treffen. Wenn Linien in die CityEngine eingeladen werden, geht CityEngine davon aus, dass es sich um Straßen handelt. Die vordefinierten Parameter für die Erzeugung der initialen Shapes, die durch die Pufferung der Skelettlinien entstehen, gilt es an die Gegebenheiten einer Schienentrasse anzupassen. So wird die Erzeugung von Gehwegen deaktiviert, indem die Parameter für die Gehwege auf null gesetzt werden. Für die Straßen- bzw. Schienenbreite wird vom Autor der Wert von 2,40 m festgelegt, der einen plausiblen Wert für die Breite einer Bahnschwelle darstellt. Weiterhin ist es wichtig, den korrekten Kreuzungstyp für die Knotenpunkte festzulegen. Die „Kreuzungspunkte“ treten immer an den Enden der Schienentrasse auf, d.h. an den Stellen, an denen auf einen Knoten keine weiterführende Kante folgt. Dies ist im vorliegenden Datensatz sehr oft der Fall da die Schienentrasse aufgrund der Zerlegung in Bauelemente, aus vielen einzelnen Elementen besteht. Für diese Arbeit wird der Kreuzungstyp *Crossing* gewählt, der die korrekte Darstellung der Schienentrasse gewährleistet. In der Abbildung 5.38 sind auf der linken Seite die Skelettlinien mit Knoten und Kanten dargestellt, auf der rechten Seite sind zusätzlich die initialen Shapes eingeblendet. Auf die initialen Shapes werden die CGA-Regeln angewandt, die Shapes bilden somit die Grundlage der Modellierung. In der Abbildung ist weiterhin zu erkennen, dass die Schienentrasse nicht durchgängig verläuft, was daran liegt, dass nur die Bauelemente normale Schienen eingeblendet sind.

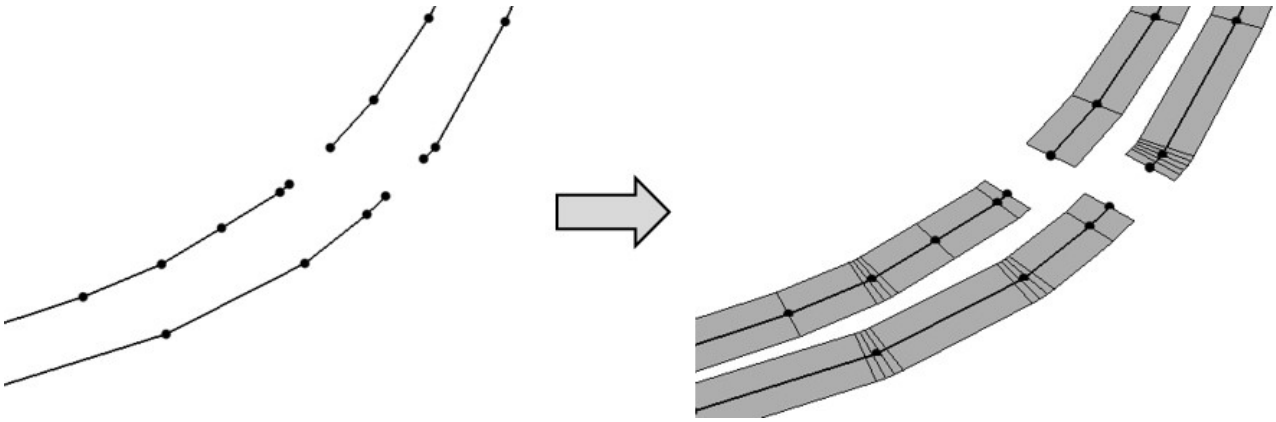

Abbildung 5.38 – Skelettlinien der normalen Schienen mit Kanten und Knoten (links) und mit Knoten, Kanten und initialen Shapes (rechts)

Zudem zeigt die Abbildung den Verlauf einer Kurve in der Schienentrasse. CityEngine ist jedoch nicht in der Lage Bögen darzustellen, daher werden diese durch kleine Geraden angenähert. Die in der Abbildung dargestellten Kurven weisen unterschiedlich starke Richtungsänderungen auf. Bei leichten Richtungsänderungen werden die Shapes aneinander angepasst. Bei starken Richtungsänderungen

überbrückt CityEngine die Kurve mithilfe kleiner aneinander gereihter Polygone; dies ist an vier Stellen in der Abbildung der Fall.
Wie bereits erwähnt bilden die initialen Shapes die Grundlage für die 3D-Modellierung in CityEngine. Nachfolgend wird die CGA-Regel *Schiene.cga* erläutert, die das Modellierungskonzept für die normalen Schienen darstellt. Die Abbildung 5.39 zeigt den ersten Abschnitt der Regel *Schiene.cga*. Zu Beginn jeder Regel werden die in der Regel benötigten Attribute definiert. Die Attribute dienen als Eingabeparameter für die geometrischen Operationen, die auf das initiale Shape angewendet werden. Die Werte der Attribute können entweder in der Regel übergeben werden oder Werte aus den Attributtabellen der Shapes übernehmen. In der vorliegenden Regel werden die sechs Attribute *Bauelement*, *Schienenabstand*, *Schienenbreite*, *Schienenhoehe*, *Schwellenbreite* und *Schwellenhoehe* definiert. Die Werte der Attribute *Schienenbreite*, *Schienenhoehe*, *Schwellenbreite* und *Schwellenhoehe* unterliegen der Wahl des Autors und orientieren sich an realistischen Schienen. Die Attribute *Bauelement* und *Schienenabstand* liegen in der Attributtabelle der normalen Schienen vor. Das Attribut Schienenabstand beschreibt den Abstand zwischen beiden Schienen, der standardmäßig 1,435 m beträgt. Die numerischen Attributwerte werden in der Einheit Meter angegeben. Im ersten Absatz der Regel wird die sogenannte Startregel festgelegt, die durch *@StartRule* gekennzeichnet wird. Die Startregel lautet in diesem Fall *Gleistrasse* und besagt: Für den Fall, dass das Bauelemente Schiene lautet, wird die Regel *Gleisbett* angewandt, für alle anderen Fälle erfolgt die Anweisung nichts zu tun. Mithilfe dieser Bedingung wird sichergestellt, dass die Regel Schiene automatisch dem Bauelement Schiene zugeordnet wird, und zwar nur diesem.

```
attr Bauelement = "Schiene"

attr Schienenabstand = 1.435
attr Schienenbreite = 0.07
attr Schienenhoehe = 0.15

attr Schwellenbreite = 0.26
attr Schwellenhoehe = 0.07



@StartRule
Gleistrasse-->
    case Bauelement == "Schiene": Gleisbett
    else                        : NIL
```

Abbildung 5.39 – CGA-Regel *Schiene.cga* (1 von 2)

Die Regel *Gleisbett* definiert den Aufbau des Gleisbetts, sie ist gemeinsam mit den weiteren Regeln der *Schiene.cga* in Abbildung 5.40 dargestellt. In *Gleisbett* wird einerseits festgelegt, dass im Gleisbett ein Gleis verläuft, andererseits wird bestimmt, dass alle 2 m eine Schwelle folgt. In der Regel *Gleisbett* werden zwei neue Regeln *Gleis* und *Schwelle* erzeugt. Diese werden im weiteren Verlauf der Regel definiert.
Die CGA-Regeln folgen einem hierarchischen Prinzip, d.h. zuerst werden die übergeordneten Elemente in den Regeln definiert und anschließend die untergeordneten Elemente.
In der Regel *Schwelle* werden Bahnschwellen definiert. Die Regel besagt, dass die Schwelle um

die *Schwellenhoehe* extrudiert wird, und teilt ihr gleichzeitig eine Farbe zu, in dem vorliegenden Fall die Farbe grau. In der Regel *Gleis* wird das Shape der Breite nach unterteilt und eine neue Regel Schienenkorpus erzeugt. Zudem werden der Ort und die Maße des Schienenkorpus definiert. Die Regel *Schienenkorpus* weist dem Schienenkorpus die Farbe braun zu und unterteilt diesen in seine Komponenten. Der Vorder- und Rückseite des Schienenkorpus wird die Regel *Schienenseite* zugewiesen. Die Regel *Schienenseite* ist die letzte Regel in der Hierarchie der Regel *Schiene.cga*; mit deren Hilfe gewinnt der Schienenkorpus an Detailtiefe.

```
Gleisbett --> NIL Gleis NIL Gleis NIL
    split (u, unitSpace, 0) {~1 : NIL | ~Schwellenbreite : Schwelle | ~1 : NIL}*

Schwelle -->
    extrude(Schwellenhoehe)
    color("#bfbfbf") # graue Farbe der Schwellen

Gleis -->
    split(v, unitSpace, 0){~0.5: NIL | Schienenbreite: t(0,Schienenbreite,0)
    extrude(Schienenhoehe) Schienenkorpus | Schienenabstand: NIL |
    Schienenbreite: t(0,Schienenbreite,0) extrude(Schienenhoehe) Schienenkorpus | ~0.5: NIL}

Schienenkorpus -->
    color("#8b4513") # braune Farbe der Schienen
    comp(f){horizontal: Schiene. | front: Schienenseite | back: Schienenseite | all: NIL}


Schienenseite -->
    split(y){'0.2 : Schiene. | ~1: t(0,0,-0.02) Schiene. extrude(0.02) comp(f){front: Schiene. |
     back: Schiene.} | '0.2 : Schiene.}
```

Abbildung 5.40 – CGA-Regel *Schiene.cga* (2 von 2)

Nach der erfolgreichen Anwendung der Regel *Schiene.cga* auf die initialen Shapes entstehen die 3D-Modelle der normalen Schienen. Die Abbildung 5.41 zeigt ein Shape und das darauf entstandene 3D-Modell der Schiene. Die Schwellen in grau sind deutlich zu erkennen, ebenso wie die einzelnen Schienen in braun. Die Details der Einrückungen an den Schienenseiten entstehen durch die Regel *Schienenseite*.

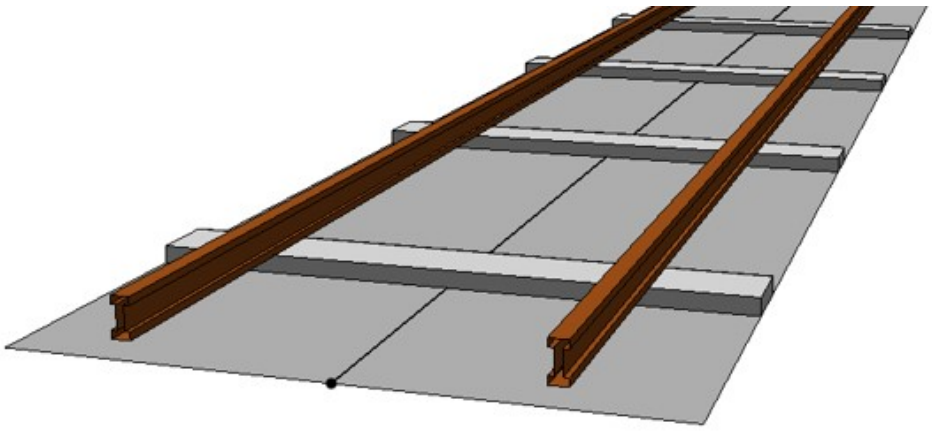

Abbildung 5.41 – 3D-Modell der normalen Schiene

In Abbildung 5.42 ist das Modell der normalen Schienen in einer Kurve der Schienentrasse dargestellt. Die Annäherung der Kurve durch die einzelnen kleinen Shapes ist deutlich zu erkennen.

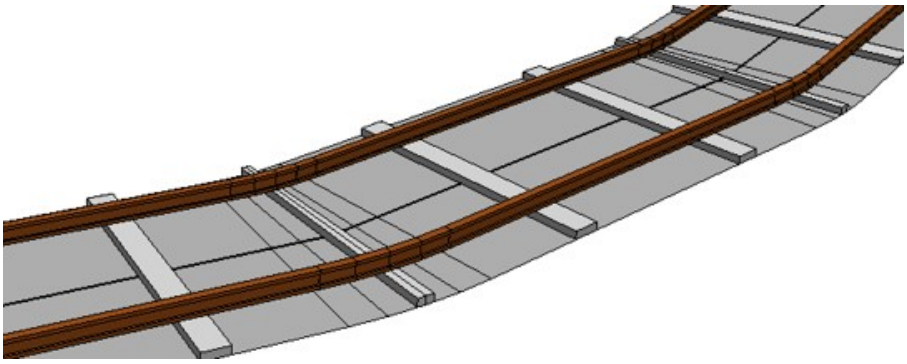

Abbildung 5.42 – 3D-Modell der normalen Schienen in einer Kurve

Zum besseren Verständnis zeigt die Abbildung 5.43 die bereits beschriebene Hierarchie des Aufbaus der CGA-Regeln am Beispiel der Regel *Schiene.cga* und die dazugehörigen Modelle.

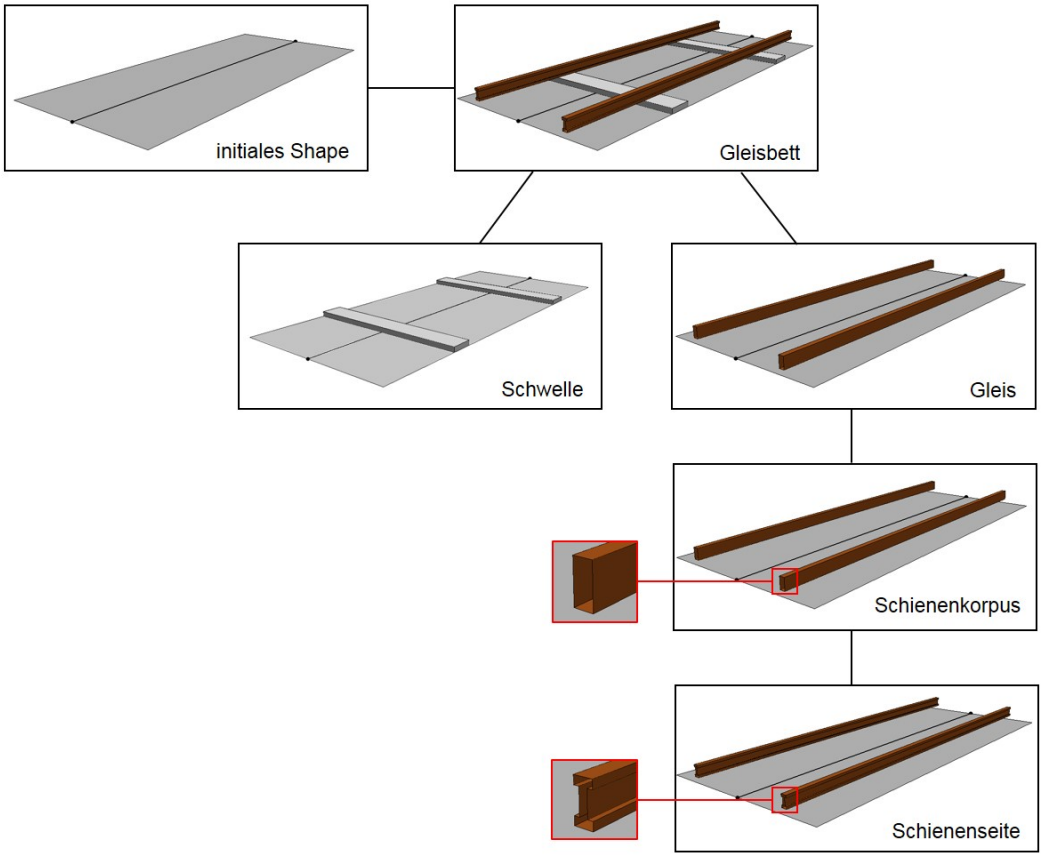

Abbildung 5.43 – Hierarchie der Regel *Schiene.cga*

5.2.2 Bahnübergang

Ein weiteres Element, das mithilfe von CityEngine modelliert wird, stellen die Bahnübergänge dar. Ebenso wie bei den normalen Schienen sind beim Laden der Bahnübergänge die richtigen Einstellungen festzulegen. Diese entsprechen bei den Bahnübergängen denen der normalen Schienen. Folglich werden die Gehwege deaktiviert und die Breite für die Shapes auf 2,40 m gesetzt sowie der Typ *Crossing* für die Kreuzungspunkte definiert. Die Bahnübergänge setzen sich aus zwei verschiedenen Typen zusammen. Einerseits aus den Bahnübergängen, an denen sich Schiene und Straße kreuzen, andererseits aus den Bahnübergängen, an denen sich Schiene und Fußweg kreuzen, die in dieser Arbeit Fußbahnübergänge genannt werden. Für die Modellierung in CityEngine im Rahmen dieser Arbeit werden diese beiden Typen zusammengefasst und erhalten eine gemeinsame CGA-Regel *Bahnuebergang.cga*. Generell ist eine Trennung der beiden Typen durchaus sinnvoll, falls eine getrennte Modellierung erwünscht ist, kann so jederzeit - ohne großen Aufwand - eine weitere Regel erstellt werden.
In der Abbildung 5.44 wird der erste Abschnitt der Regel *Bahnuebergang.cga* dargestellt. Zu den aus der vorherigen Regel *Schiene.cga* bekannten Attributen *Schienenbreite*, *Schienenhoehe* und *Bauelement* sind die beiden Attribute *Spalt* und *SchienenabstandAsphalt* hinzugekommen, wobei das Attribut *SchienenabstandAsphalt* den gleichen Zweck wie das vorherige Attribut *Schienenabstand* erfüllt. Der Wert des *SchienenabstandAsphalts* ist jedoch etwas geringer, damit Platz für die Räder der Straßenbahn geschaffen wird. Um die Schienen für die überquerenden Fahrzeuge befahrbar zu machen, werden die Lücken zwischen den einzelnen Schienen mit Asphalt aufgefüllt. Damit die Schienen weiterhin von der Straßenbahn befahren werden können, muss jedoch ein Spalt für die Straßenbahnräder bleiben, dessen Breite im Attribut *Spalt* definiert wird. Die Werte der Attribute *Spalt* und *SchienenabstandAsphalt* unterliegen der Wahl des Autors. Dem bereits bekannten Attribut *Bauelement* werden in der vorliegenden Regel die Werte *Bahnuebergang* und *FussBahnuebergang* aus der Attributtabelle der Shapes zugewiesen; der in der Regel abgebildete Wert *Bahnuebergang* dient somit als „Platzhalter“.
Die Startregel *BahnuebergangStart* besagt, dass für das Attribut *Bauelement* mit dem Wert *Bahnuebergang* oder *FussBahnuebergang* die Regel *Bahnuebergang* ausgeführt werden soll.

```
attr Bauelement = "Bahnuebergang"
attr Spalt = 0.05 # Spalt für Straßenbahnräder

attr SchienenabstandAsphalt = 1.335
attr Schienenbreite = 0.07
attr Schienenhoehe = 0.15


@StartRule
BahnuerbgangStart -->
    case Bauelement == "Bahnuebergang" || Bauelement == "FussBahnuebergang" : Bahnuebergang
    else                                                                    : NIL
```

Abbildung 5.44 – CGA-Regel *Bahnuebergang.cga* (1 von 2)

Die Abbildung 5.45 zeigt den zweiten Abschnitt der Regel *Bahnuebergang.cga*. Die Regel *Bahnuerbgang* gleicht im Wesentlichen dem Aufbau der Regel *Gleis*. Neben der Regel *Schienenkorpus* wird zudem die Regel *Asphalt* erstellt, die Regel *Schwelle* entfällt. In der Regel *Asphalt* wird dem Asphalt eine dunkelgraue Farbe zugeordnet, der Asphalt wird um 21,5 cm extrudiert. Somit liegt der Asphalt 5 mm unter der Höhe der Schienen, was für die Erstellung der Regel *Weiche.cga* (vgl. Kapitel 5.2.3) von Bedeutung ist. Die Regeln Schienenkorpus und Schienenseite sind analog zu denen der Regel *Schiene.cga*.

```
Bahnuebergang -->
    split(v, unitSpace, 0){~0.5: Asphalt | 0.01: NIL |
    Schienenbreite: t(0,Schienenbreite,0) extrude(Schienenhoehe) Schienenkorpus |
    Spalt: NIL | SchienenabstandAsphalt: Asphalt | Spalt: NIL |
    Schienenbreite: t(0,Schienenbreite,0) extrude(Schienenhoehe) Schienenkorpus |
    0.01 : NIL | ~0.5: Asphalt}

Asphalt -->
    color("#696969")# dunkelgrau
    extrude(0.215)

Schienenkorpus -->
    color("#8b4513") # braune Farbe der Schienen
    comp(f){horizontal: Schiene. | front: Schienenseite | back: Schienenseite | all: NIL}

Schienenseite -->
    split(y){'0.2 : Schiene. |
     ~1: t(0,0,-0.02) Schiene. extrude(0.02) comp(f){front: Schiene. |
     back: Schiene.} | '0.2 : Schiene.}
```

Abbildung 5.45 – CGA-Regel *Bahnuebergang.cga* (2 von 2)

Die Abbildung 5.46 zeigt das 3D-Modell für das Bauelement *Bahnuebergang* nach der erfolgreichen Anwendung der Regel *Bahnuebergang*.cga.

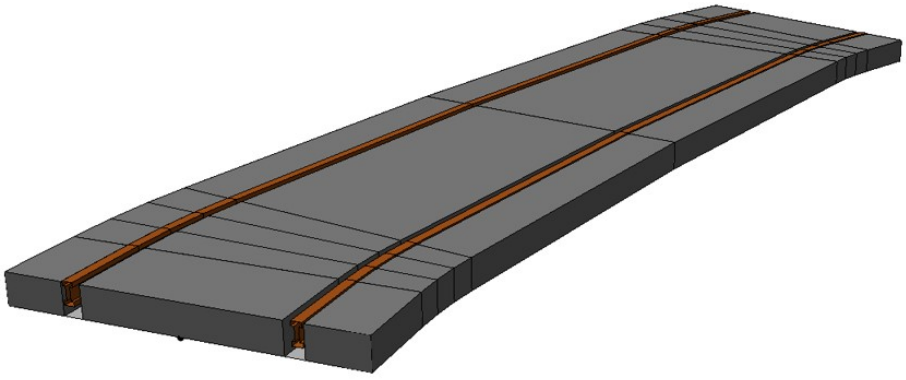

Abbildung 5.46 – 3D-Modell eines Bahnübergangs

In der Abbildung 5.47 sind die 3D-Modelle zweier Fußbahnübergänge dargestellt, die Elemente zweier parallel verlaufender Gleise sind. Im Vergleich zur Abbildung 5.46 sind die Längen der Elemente deutlich kürzer, was die deutlich geringere Breite der Fußwege im Gegensatz zu den Straßen zeigt.

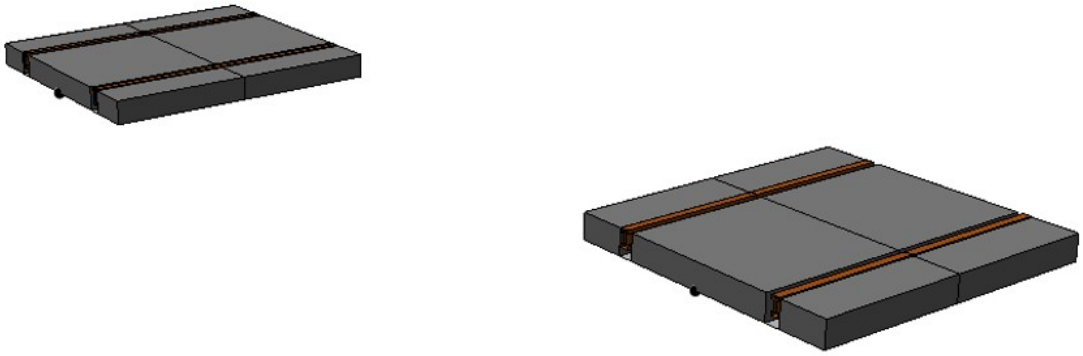

Abbildung 5.47 – 3D-Modell von zwei Fußbahnübergängen

5.2.3 Weiche und Gleiskreuzung

Die Weichen und Gleiskreuzungen stellen zwei weitere Elemente dar, für die in der CityEngine CGA-Regeln erstellt werden sollen, um 3D-Modelle zu erzeugen.
Die Idee hinter der Zerlegung der Schienentrasse in Elemente ist, für jedes Element eine CGA-Regel zu erstellen. Bei den Weichen und Gleiskreuzungen stellt die Modellierung der sich kreuzenden Gleise eine Besonderheit dar. So sind die Schienen an den Kreuzungen der Gleise nicht durchgängig, damit die Straßenbahn entweder die Möglichkeit hat das kreuzende Gleis zu überqueren (Gleiskreuzung) oder das Gleis zu wechseln (Weiche). Die Abbildung 5.48 zeigt dies am Beispiel einer Weiche, der rote Kreis markiert die Kreuzung der Gleise. Die Gleiskreuzungen weisen die selbe Problematik auf.

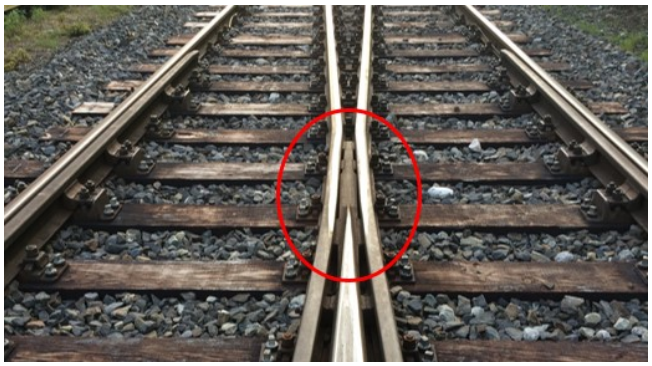

Abbildung 5.48 – Weiche [eigene Aufnahme]

Die Abbildung 5.49 zeigt exemplarisch für eine Weiche (rechts) und eine Gleiskreuzung (links), dass die in CityEngine geladenen Weichen und Gleiskreuzungen nicht aus einem, sondern aus mehreren Shapes bestehen. Bedingt durch die Richtungsänderungen und die kreuzenden bzw. abzweigenden

Linien werden die Kanten durch mehrere Knoten unterbrochen. Somit bestehen die Elemente nicht aus einem Shape, wie es die Modellierung für die Weichen und Gleiskreuzungen benötigen würde, sondern aus vielen. Es ist somit nicht möglich eine CGA-Regel für die Weichen und Gleiskreuzungen als „Ganzes“ zu erstellen. Daher werden diese im Rahmen dieser Arbeit ähnlich zu den normalen Schienen modelliert.

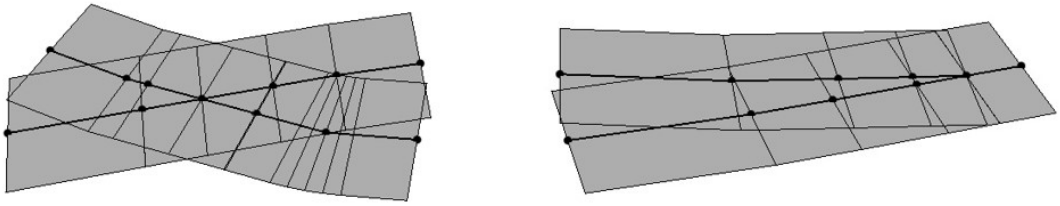

Abbildung 5.49 – Initialen Shapes einer Gleiskreuzung (links) und eine Weiche (rechts)

Eine weitere Besonderheit der Weichen und Gleiskreuzungen ist, dass sich einige Weichen und Gleiskreuzungen mit den Bahnübergängen (wird in diesem Kapitel stellvertretend für die Bahn- und Fußbahnübergänge verwendet) überlagern (vgl. Kapitel 5.1.2). Für diesen Fall besitzen die Weichen und Gleiskreuzungen das Attribut *Bauelement = Weiche_Asphalt* bzw. *Gleiskreuzung_Asphalt*. Für die Modellierung im Rahmen dieser Arbeit wird für die Weichen und Gleiskreuzungen die gemeinsame Regel *Weiche.cga* erstellt, die eine Kombination der Regeln *Schiene.cga* und *Bahnuebergang.cga* darstellt. Die Abbildung 5.50 beschreibt den ersten Abschnitt der Regel *Weiche.cga*. Die in der Regel *Weiche.cga* definierten Attribute entsprechen den Attributen der Regeln *Schiene.cga* und *Bahnuebergang.cga*, wobei der Attributwert *Weiche* wieder als Platzhalter für die Attributwerte in den Tabellen der Weichen und Gleiskreuzungen dient. In der Startregel *WeicheGleiskreuzung* wird definiert, dass für die Attribute *Bauelement = Weiche* und *Bauelement = Gleiskreuzung* die Regel *Weiche* angewandt wird und für die Attribute *Bauelement = Weiche_Asphalt* bzw. *Gleiskreuzung_Asphalt* die Regel *WeicheAsphalt* angewandt wird.

```
attr Bauelement = "Weiche"
attr Spalt = 0.05 # Spalt für Straßenbahnräder

attr Schienenabstand = 1.435
attr SchienenabstandAsphalt = 1.335
attr Schienenbreite = 0.07
attr Schienenhoehe = 0.15

attr Schwellenbreite = 0.26
attr Schwellenhoehe = 0.07

@StartRule
WeicheGleiskreuzung-->
    case Bauelement == "Weiche" || Bauelement == "Gleiskreuzung"                : Weiche
    case Bauelement == "Weiche_Asphalt" || Bauelement == "Gleiskreuzung_Asphalt" : WeicheAsphalt
    else                                                                        : NIL
```

Abbildung 5.50 – CGA-Regel *Weiche.cga* (1 von 3)

In der Abbildung 5.51 ist der zweite Abschnitt der Regel *Weiche.cga* dargestellt. Dieser Abschnitt stellt die Regel Weiche und deren untergeordnete Regeln dar, die ausgeführt werden, wenn das Attribut *Bauelemente = Weiche* bzw. *Gleiskreuzung* lautet. Die in diesem Abschnitt aufgezeigten Regeln entsprechen denen der Regel *Schiene.cga*, lediglich der Regelname *Gleisbett* wird durch den Regelnamen *Weiche* ersetzt.

```
Weiche --> Gleis
    split (u, unitSpace, 0) {~1 : NIL | ~Schwellenbreite : Schwelle | ~1 : NIL}*

Schwelle -->
    extrude(Schwellenhoehe)
    color("#bfbfbf") # graue Farbe der Schwellen

Gleis -->
    split(v, unitSpace, 0){~0.5: NIL | Schienenbreite: t(0,Schienenbreite,0)
    extrude(Schienenhoehe) Schienenkorpus | Schienenabstand: NIL |
    Schienenbreite: t(0,Schienenbreite,0) extrude(Schienenhoehe) Schienenkorpus | ~0.5: NIL}

Schienenkorpus -->
    color("#8b4513" # braune Farbe der Schienen
    comp(f){horizontal: Schiene. | front: Schienenseite | back: Schienenseite | all: NIL}


Schienenseite -->
    split(y){'0.2 : Schiene. | ~1: t(0,0,-0.02) Schiene. extrude(0.02) comp(f){front: Schiene. |
    back: Schiene.} | '0.2 : Schiene.}
```

Abbildung 5.51 – CGA-Regel *Weiche.cga* (2 von 3)

Die Abbildung 5.52 zeigt den dritten und letzten Abschnitt der Regel *Weiche.cga*. Für den Fall, dass dem Attribut *Bauelement* der Wert *Weiche_Asphalt* oder *Gleiskreuzung_Asphalt* zugewiesen wird, werden die in diesem Abschnitt enthaltenen Regeln ausgeführt. Die abgebildeten Regeln entsprechen denen der Regel *Bahnuebergang.cga*. Wie bereits im zweiten Abschnitt der Regel *Weiche.cga*, wird auch in diesem Abschnitt der ursprüngliche Regelname *Bahnuebergang* aus der Regel *Bahnuebergang.cga* durch den Namen *WeicheAsphalt* ersetzt. Die in der Regel *WeicheAsphalt* erstellte Regel *Schienenkorpus* erhält den Zusatz *Asphalt*, sodass deren Name *SchienenkorpusAsphalt* lautet. Da der Regelname Schienenkorpus bereits vergeben ist, wird somit eine eindeutige Zuordnung innerhalb der Regel möglich. Selbes gilt für die Regel *Schienenseite*; aus dem Namen *Schienenseite* wird *SchienenseiteAsphalt*. Wie bereits erwähnt, liegt die Höhe des extrudierten Asphalts 5 mm unter der Höhe der Schienen. Dies ist wichtig, da die Schienen im Bereich der sich überlappenden Shapes - bei gleicher Höhe von Schiene und Asphalt - nicht mehr sichtbar wären.

```
WeicheAsphalt -->
    split(v, unitSpace, 0){~0.5: Asphalt | 0.01: NIL |
    Schienenbreite: t(0,Schienenbreite,0) extrude(Schienenhoehe) SchienenkorpusAsphalt |
    Spalt: NIL | SchienenabstandAsphalt: Asphalt | Spalt: NIL |
    Schienenbreite: t(0,Schienenbreite,0) extrude(Schienenhoehe) SchienenkorpusAsphalt |
    0.01 : NIL | ~0.5: Asphalt}

Asphalt -->
    color("#696969")# dunkelgrau
    extrude(0.215)

SchienenkorpusAsphalt -->
    color("#8b4513") # braune Farbe der Schienen
    comp(f){horizontal: Schiene. | front: SchienenseiteAsphalt | back: SchienenseiteAsphalt | all: NIL}


SchienenseiteAsphalt -->
    split(y){'0.2 : Schiene. | ~1: t(0,0,-0.02) Schiene. extrude(0.02) comp(f){front: Schiene. |
    back: Schiene.} | '0.2 : Schiene.}
```

Abbildung 5.52 – CGA-Regel *Weiche.cga* (3 von 3)

Die Abbildung 5.53 zeigt das 3D-Modell einer Weiche nach der Anwendung der Regel *Weiche.cga* auf die initialen Shapes. Die Struktur einer Weiche ist in dem Modell klar zu erkennen. Da die Regel auf jedes einzelne Shape angewandt wird, wird die Breite der Schwellen der Größe der Shapes angepasst. Es ist gut zu erkennen, dass die Schwellen vor der Abzweigung deutlich schmaler sind als danach. In der Realität werden die Gleise für die Länge der Abzweigung auf langen, gemeinsamen Schwellen befestigt. Dieser Tatsache werden die mittels der CGA-Regeln erstellten Modelle nicht gerecht. Durch die durchgängigen Schienen fehlt dem Modell ein weiteres Stück Realitätsnähe, da dieses aufgrund des fehlenden Spalts für die Straßenbahnräder streng genommen nicht befahrbar ist. Diese Problematik wurde bereits zu Beginn dieses Kapitels angesprochen.

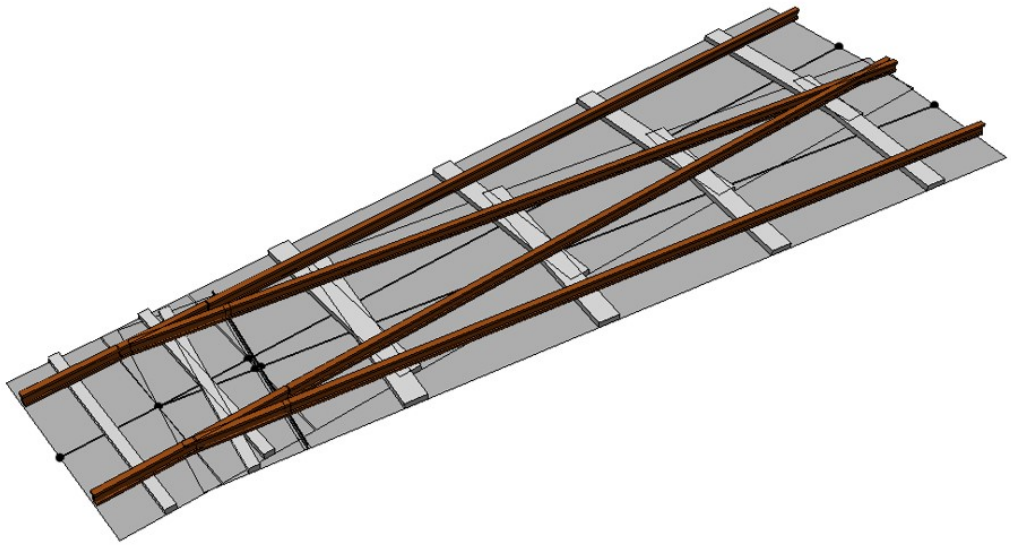

Abbildung 5.53 – 3D-Modell einer Weiche

Die Abbildung 5.54 stellt die 3D-Modelle zweier Weichen dar. Der Großteil wird jedoch nicht durch normale Schienen modelliert, sondern durch asphaltierte Abschnitte. Diese Abschnitte entsprechen dem Attribut *Bauelement = Weiche_Asphalt*.

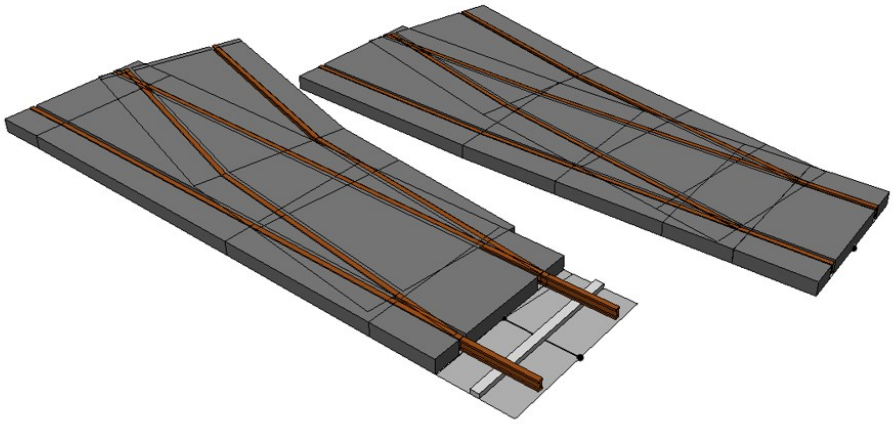

Abbildung 5.54 – 3D-Modell zweier Weichen mit großenteils asphaltierten Abschnitten

In der Abbildung 5.55 ist das 3D-Modell nach der erfolgreichen Anwendung der Regel *Weiche.cga* auf eine Gleiskreuzung zu sehen. Wie bei den Weichen ist auch hier die Form einer Gleiskreuzung klar zu erkennen. Die in der Realität durchgängigen Schwellen werden auch in diesem Modell nicht realisiert. Die Schwellen folgen dem Verlauf der Shapes und überkreuzen sich somit stark, zudem passt sich die Breite der Schwellen der Größe der Shapes an. Auch hier ist streng genommen keine Befahrung möglich, da zwischen den kreuzenden Schienen kein Spalt für die Räder der Straßenbahn existiert.

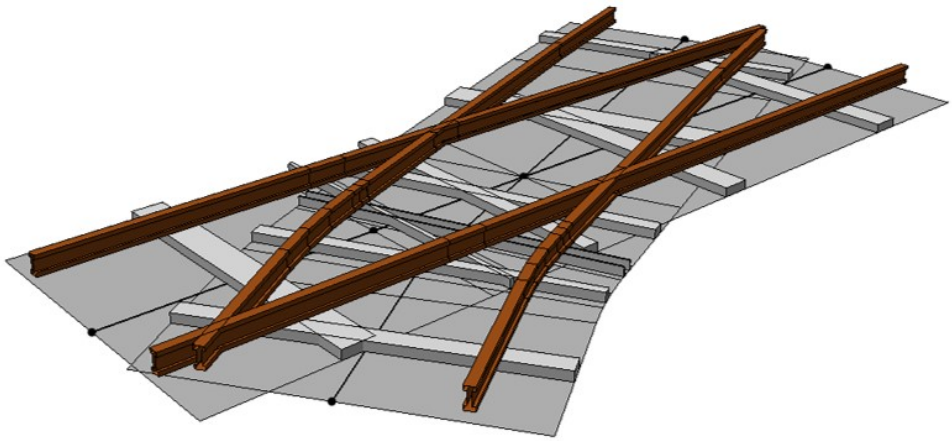

Abbildung 5.55 – 3D-Modell einer Gleiskreuzung

Die Abbildung 5.56 zeigt das 3D-Modell einer Gleiskreuzung, welches zwei unterschiedlich modellierte Abschnitte besitzt. Ein Abschnitt ist mithilfe von Schienen modelliert, der andere Abschnitt ist asphaltiert. Wie bei der vorherigen Abbildung fallen die unterschiedlichen Schwellenbreiten und die sich

kreuzenden Schwellen ins Auge. Hinzukommt der unrealistisch „zackige“ Übergang zwischen Asphalt und Schiene. Dieser kommt durch die Lage der Shapes zustande.

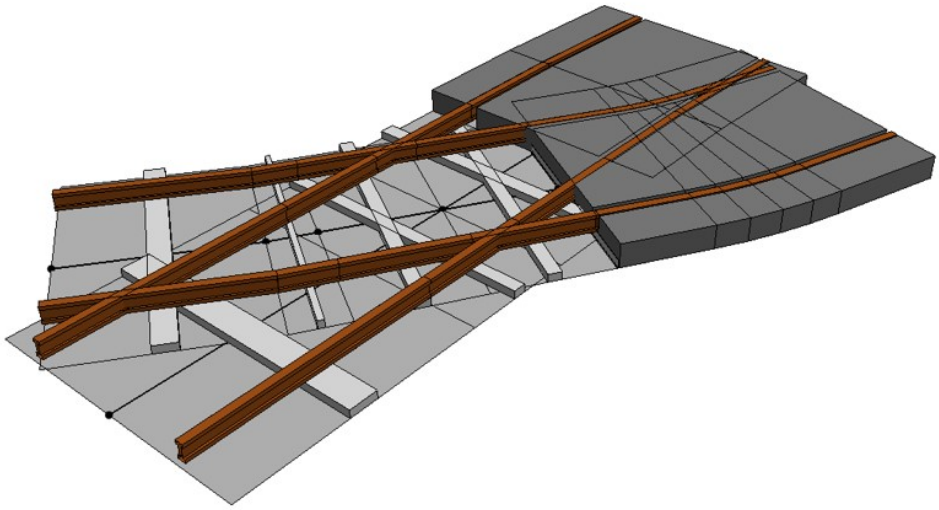

Abbildung 5.56 – Modell einer Gleiskreuzung mit asphaltiertem Abschnitt

5.2.4 Bahnsteig und Bahnsteigpolygon

Die Bahnsteige und Bahnsteigpolygone stellen die letzten zwei Elemente dar, für die im Rahmen dieser Arbeit Modellierungskonzepte in Form von CGA-Regeln erstellt werden. Bei dem Bauelement Bahnsteig handelt es sich ursprünglich um Linien aus den OSM-Daten, die die Lage und Länge der Bahnsteige kennzeichnen und parallel zur Schienentrasse verlaufen (vgl. Kapitel 5.1.2). Die Bahnsteigpolygone hingegen zeigen die komplette Form des Bahnsteigs auf (vgl. Kapitel 5.1.2). Im Folgenden wird die Erstellung der beiden Regeln *Bahnsteig.cga* und *Bahnsteigpolygon.cga* näher erläutert. Die Idee hinter dem Bauelement Bahnsteig ist die automatische Erzeugung eines parallel zu den Gleisen verlaufenden Bahnsteigpolygons. In der Regel *Bahnsteig.cga* werden somit einerseits die Schienen und andererseits die Bahnsteigpolygone erzeugt. Die Abbildung 5.57 zeigt den ersten Abschnitt der Regel *Bahnsteig.cga*. Neben den Attributen *Bauelement*, *Schienenabstand*, *Schienenbreite*, *Schienenhoehe*, *Schwellenbreite* und *Schwellenhoehe*, die aus den bisher erläuterten Regeln bekannt sind, wird das Attribut *Bahnsteighoehe* definiert. Der Wert des Attributs *Bahnsteighoehe* unterliegt hier der Wahl des Autors und wird auf 0,20 m festgesetzt. In der Startregel *Bahnsteig-Start* wird bestimmt, dass für das Attribut *Bauelement = Bahnsteig* die Regel *Bahnsteig* angewandt wird.

```
attr Bauelement = "Bahnsteig"

attr Schienenabstand = 1.435
attr Schienenbreite = 0.07
attr Schienenhoehe = 0.15

attr Schwellenbreite = 0.26
attr Schwellenhoehe = 0.07

attr Bahnsteighoehe = 0.2


@StartRule
BahnsteigStart -->
    case Bauelement == "Bahnsteig" : Bahnsteig
    else                           : NIL
```

Abbildung 5.57 – CGA-Regel *Bahnsteig.cga* (1 von 3)

In der Abbildung 5.58 wird der zweite Abschnitt der Regel *Bahnsteig.cga* beschrieben. Die Regel *Bahnsteig* unterteilt die initialen Shapes in die Regeln *Gleisbett* und *Plattform*. Die Regel *Gleisbett* sowie deren untergeordnete Regeln *Schwelle*, *Gleis*, *Schienenkorpus* und *Schienenseite* entsprechen dabei den Regeln aus *Schiene.cga* und werden an dieser Stelle nicht näher erläutert.

```
Bahnsteig -->
    alignScopeToGeometry(zUp,0) Gleisbett Plattform


Gleisbett --> Gleis
    split (u, unitSpace, 0) {~1 : NIL | ~Schwellenbreite : Schwelle | ~1 : NIL}*

Schwelle -->
    extrude(Schwellenhoehe)
    color("#bfbfbf") # graue Farbe der Schwellen

Gleis -->
    split(v, unitSpace, 0){~0.5: NIL |
     Schienenbreite: t(0,0,Schienenbreite) extrude(Schienenhoehe) Schienenkorpus |
     Schienenabstand: NIL |
      Schienenbreite: t(0,0,Schienenbreite) extrude(Schienenhoehe) Schienenkorpus | ~0.5: NIL}

Schienenkorpus -->
    color("#8b4513") # braune Farbe der Schienen
    comp(f){horizontal: Schiene. | front: Schienenseite | back: Schienenseite | all: NIL}


Schienenseite -->
    split(y){'0.2 : Schiene. | ~1: t(0,0,-0.02) Schiene. extrude(0.02) comp(f){front: Schiene. |
     back: Schiene.} | '0.2 : Schiene.}
```

Abbildung 5.58 – CGA-Regel *Bahnsteig.cga* (2 von 3)

In der Abbildung 5.59 ist der dritte Abschnitt der Regel *Bahnsteig.cga* dargestellt. Dieser Abschnitt bildet die Regel *Plattform* ab, die die Erzeugung des parallel zu den Schienen verlaufenden Bahnsteigs definiert. Zu Beginn wird die Größe des Bahnsteigs festgelegt. So wird diesem eine Breite von 3 m zugeteilt, die Länge entspricht der Länge des initialen Shapes. Weiterhin wird dessen Lage

und Farbe festgelegt. Zuletzt wird das in der Regel *Plattform* definierte Bahnsteigpolygon um die *Bahnsteighoehe* extrudiert.

```
Plattform-->
    s(scope.sx,3,0)
    t(0,3.2,0) # Vorzeichen entsprechend der Bahnsteigseite ändern
    color("#c3c3c3") #grau
    extrude(world.y, Bahnsteighoehe)
```

Abbildung 5.59 – CGA-Regel *Bahnsteig.cga* (3 von 3)

Die Abbildung 5.60 beinhaltet das 3D-Modell des Elements Bahnsteig, welches nach der erfolgreichen Anwendung der CGA-Regel Bahnsteig.cga entsteht. Das Modell enthält neben den Schienen, den in der Regel definierten parallel zu den Schienen verlaufenden Bahnsteig.

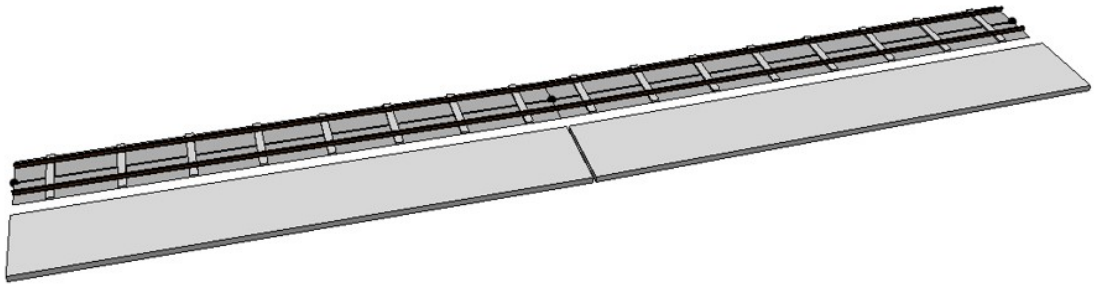

Abbildung 5.60 – 3D-Modell des Elements Bahnsteig

Neben den Bahnsteigen, die automatisch in der Regel *Bahnsteig.cga* an die Schienen modelliert werden, gibt es noch die „eigenständigen“ Bahnsteigpolygone (vgl. Kapitel 5.1.2). Die Abbildung 5.61 beschreibt die Regel *Bahnsteigpolygon.cga* zur Modellierung der Bahnsteigpolygone in CityEngine. Neben dem Attribut *Bauelement* wird das Attribut *Bahnsteighoehe* definiert. Der Wert des Attributs *Bahnsteighoehe* wird vom Autor auf 0,20 m bestimmt. In der Startregel *BahnsteigpolygonStart* wird festgelegt, dass für das Attribut *Bauelement = Bahnsteigpolygon* die Regel *Bahnsteigpolygon* ausgeführt wird. Diese besagt, dass die Polygone um die *Bahnsteighoehe* extrudiert werden und die Farbe grau erhalten.

```
attr Bauelement = "Bahnsteigpolygon"
attr Bahnsteighoehe = 0.2

@StartRule
BahnsteigpolygonStart -->
    case Bauelement == "Bahnsteigpolygon" : Bahnsteigpolygon
    else                                  : NIL

Bahnsteigpolygon -->
    extrude(Bahnsteighoehe)
    color("#c3c3c3") # grau
```

Abbildung 5.61 – CGA-Regel *Bahnsteigpolygon.cga*

Die Abbildung 5.62 zeigt zwei parallel zueinander liegende 3D-Modelle von Bahnsteigpolygonen nach der erfolgreichen Anwendung der Regel *Bahnsteigpolygon.cga*. Im Gegensatz zu den in der Regel *Bahnsteig.cga* erzeugten Bahnsteigen bestehen die Bahnsteigpolygone nur aus einem Shape.

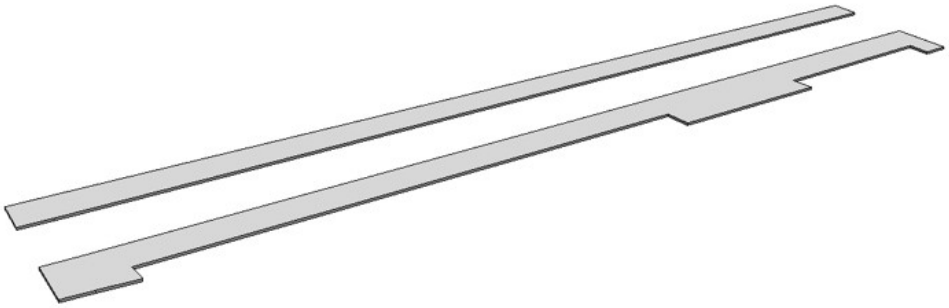

Abbildung 5.62 – 3D-Modell von zwei Bahnsteigpolygonen

6 Visualisierung und Bewertung der Ergebnisse

Das Ziel dieser Arbeit ist es, die vorwiegend manuelle Generierung des 3D-Modells für einen Straßenbahnsimulator mittels prozeduraler Modellierung soweit wie möglich zu automatisieren und zu untersuchen, inwieweit eine Automatisierung sinnvoll ist. Da es im Rahmen dieser Arbeit nicht möglich ist, ein gesamtes 3D-Modell zu erstellen, liegt der Fokus auf der Modellierung der Schienentrasse. In diesem Kapitel werden die gewonnenen Ergebnisse visualisiert und evaluiert.
Das vorliegende Kapitel wird hierzu in drei Abschnitte unterteilt. Zuerst werden Beispiele des mittels prozeduraler Modellierung erzeugten 3D-Modells der Straßenbahntrasse gezeigt und auf die positiven sowie negativen Aspekte eingegangen. Anschließend erfolgt ein Vergleich zwischen der manuellen Erstellung des 3D-Modells und der in dieser Arbeit entwickelten Methodik. Abschließend werden Verbesserungsmöglichkeiten aufgezeigt.

6.1 Realitätsnähe der generierten 3D-Modelle

In diesem Kapitel werden die auf Basis der Bauelemente erzeugten 3D-Modelle zu einer Schienentrasse zusammengeführt. Im Rahmen dieser Arbeit werden 3D-Modelle für die Elemente normale Schiene, Bahnübergang und Fußbahnübergang, Weiche und Gleiskreuzung sowie Bahnsteig und Bahnsteigpolygon erzeugt. Für die Visualisierung der Ergebnisse wird das Modell der Schienentrasse um generische Baummodelle erweitert, die an den in den OSM-Daten enthaltenen Stellen erzeugt werden. Die CGA-Regel für die Erzeugung der Baummodelle stammt aus der in CityEngine integrierten ESRI Bibliothek. Für diese Arbeit wird das Baummodell *Weißeiche* (engl.: white oak) gewählt.
Die Abbildung 6.1 zeigt eine zweispurige Schienentrasse. Das Modell lässt klar erkennen, dass es sich um Schienen handelt. Die Schwellen sind in grauer Farbe gehalten und stellen somit Betonschwellen dar. In der Abbildung 6.2 ist eine „Nahaufnahme“ der Schiene dargestellt, die die Details an den Schienen selbst gut erkennen lässt. Auch die Darstellung, dass die Schienen auf den Schwellen aufsetzen entspricht der Realität. Die Abbildung 6.3 zeigt den Schienenverlauf in einer Kurve, wobei die Gleise einen „welligen“ Eindruck hinterlassen. Dieser entsteht, da die Darstellung von Kurven in der CityEngine nicht möglich ist und diese durch Geraden angenähert werden. Besonders bei stärkeren Richtungsänderungen fällt die Annäherung negativ ins Auge (rot markiert). Die engen Kurven werden mittels Polygonen approximiert, was die Entstehung von Schwellen mit verhältnismäßig geringen Breiten zur Folge hat. Das Gesamtbild der Schienentrasse in Kurven wirkt dadurch „unrund“.

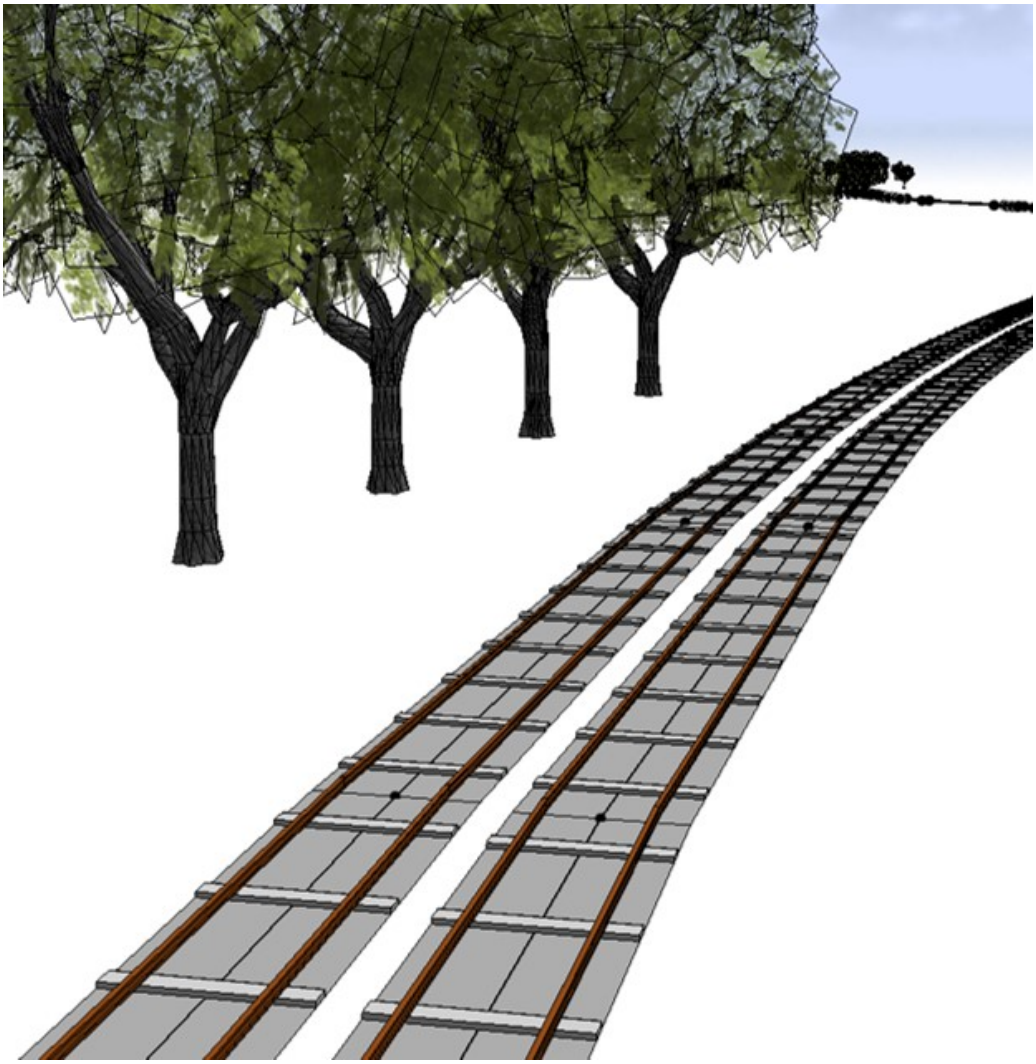

Abbildung 6.1 – Zweispurige Schienentrasse

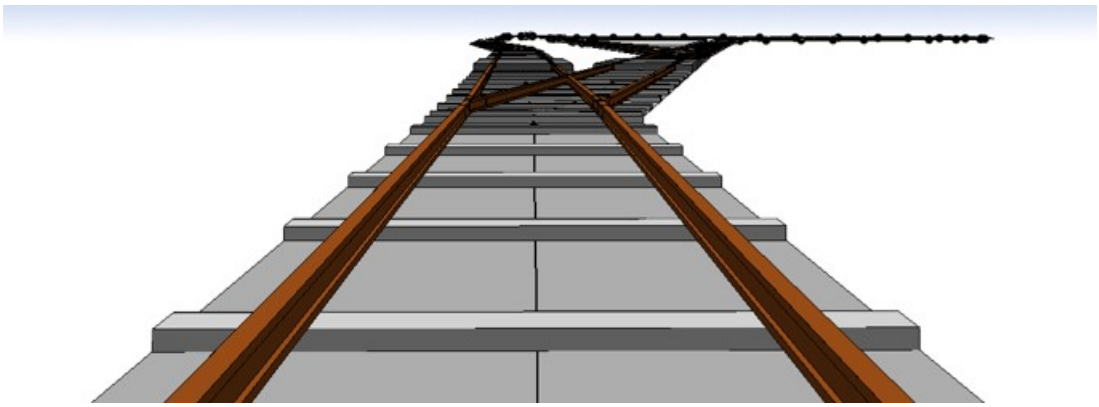

Abbildung 6.2 – „Nahaufnahme“ der Schiene

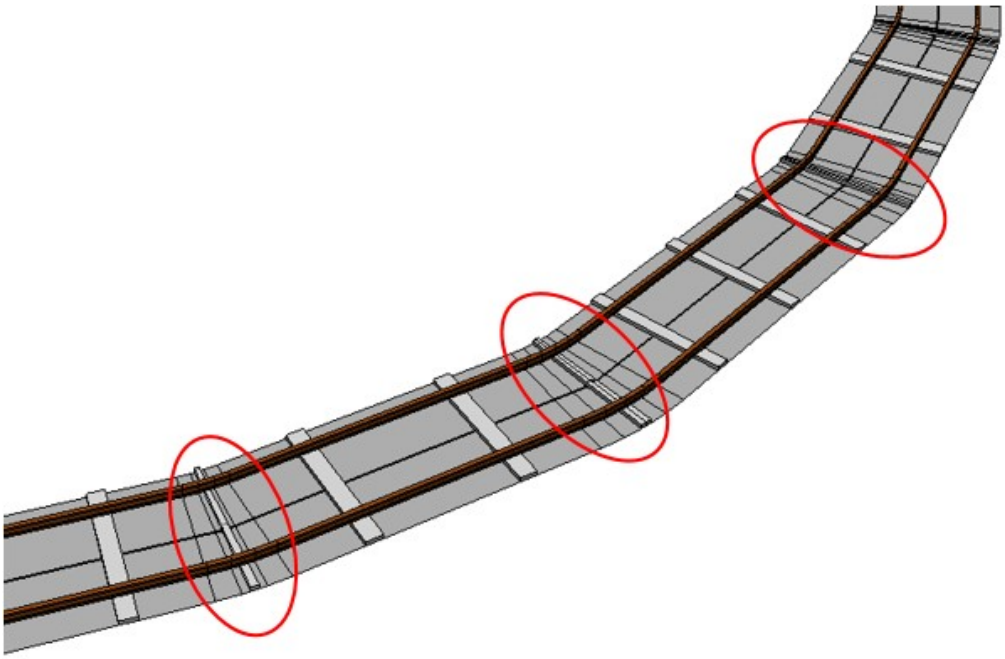

Abbildung 6.3 – Kurvenverlauf der Schienentrasse

Die Abbildung 6.4 zeigt das 3D-Modell einer Weiche. Wie bereits erwähnt, werden die Schienen in der Realität für die Länge der Abzweigung auf langen gemeinsamen Schwellen befestigt (vgl. Kapitel 5.2.3). Im Fall der vorliegenden Abbildung weist das 3D-Modell große Ähnlichkeit zur Realität auf, da die Schwellen den Großteil der Abzweigung nebeneinander liegen und somit den Eindruck einer langen gemeinsamen Schwelle erzielen. Etwa im letzten Drittel der Weiche vergrößert sich der Versatz zwischen den Schwellen. Die Realitätsnähe wird durch die langgezogene Form der Weiche erzielt. Die Abbildung 6.5 hingegen enthält zwei kurze Weichen mit einer darauffolgenden Gleiskreuzung. Im Vergleich zur vorherigen Abbildung ist die vorliegende Anordnung der Schwellen deutlich realitätsferner. Aufgrund der kleinen Polygone, die durch den gekrümmten Verlauf der Schienentrasse bedingt sind, entstehen viele schmale Schwellen, die sich überkreuzen. In der Abbildung 6.6 sind wieder zwei Weichen und eine Gleiskreuzung visualisiert. Im Gegensatz zur vorherigen Abbildung besitzen die vorliegenden Weichen sowie die Gleiskreuzung asphaltierte Abschnitte, die die Bahnübergänge innerhalb der Weiche oder der Gleiskreuzung kennzeichnen. Dies hat den Vorteil, dass das Modell aufgrund der fehlenden schmalen Schwellen realistischer wirkt. Einen negativen Aspekt stellen jedoch die Übergänge zwischen den normalen Schienen und den asphaltierten Abschnitten, vor allem im Bereich der Gleiskreuzung, dar. Der Übergang schließt nicht gerade ab, sondern in einer Art Zacke (rot markiert). Den Grund dafür stellen die teilweise überlappenden und schräg zueinander liegenden Shapes dar. Befindet sich der Übergang Schiene Asphalt im normalen Schienenverlauf oder am Anfang eines Elements, wie in der Abbildung vor der Weiche, ist der Übergang gerade und somit auch realer (grün markiert). Auch die abschnittweise Modellierung der Bahnübergänge, besonders wenn diese in die Weichen oder Gleiskreuzungen integriert sind, ist nicht realistisch.

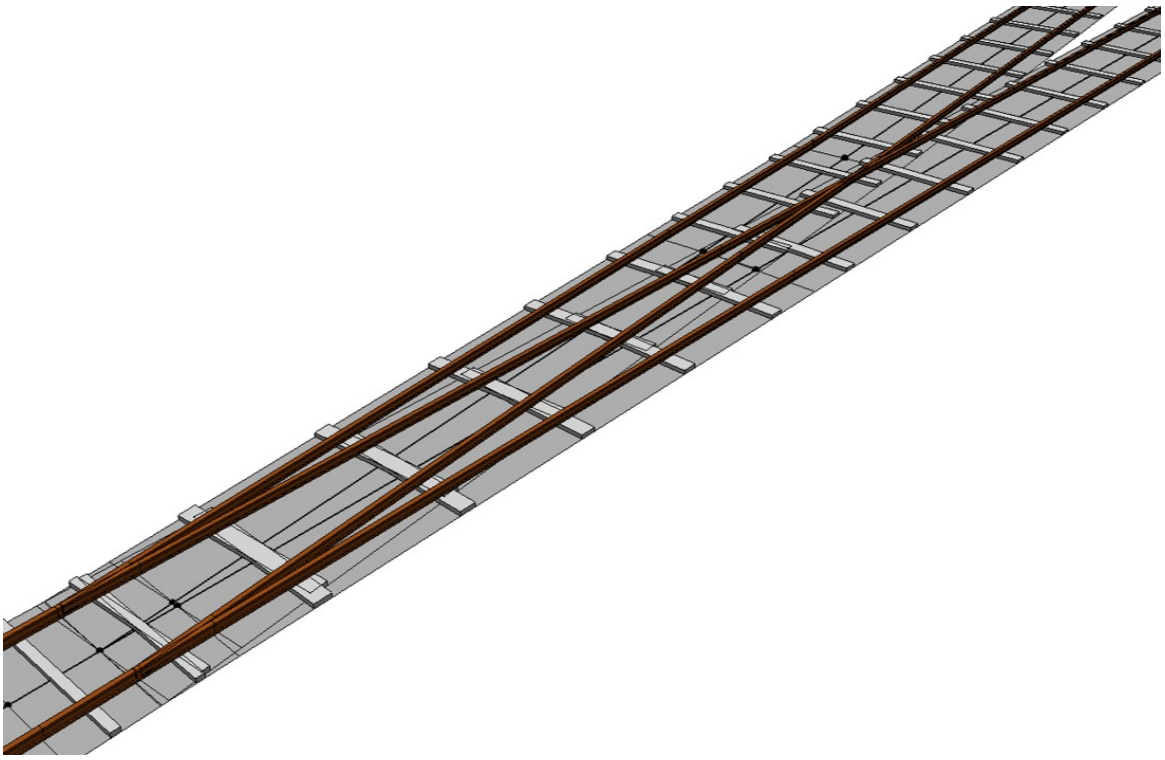

Abbildung 6.4 – 3D-Modell einer Weiche

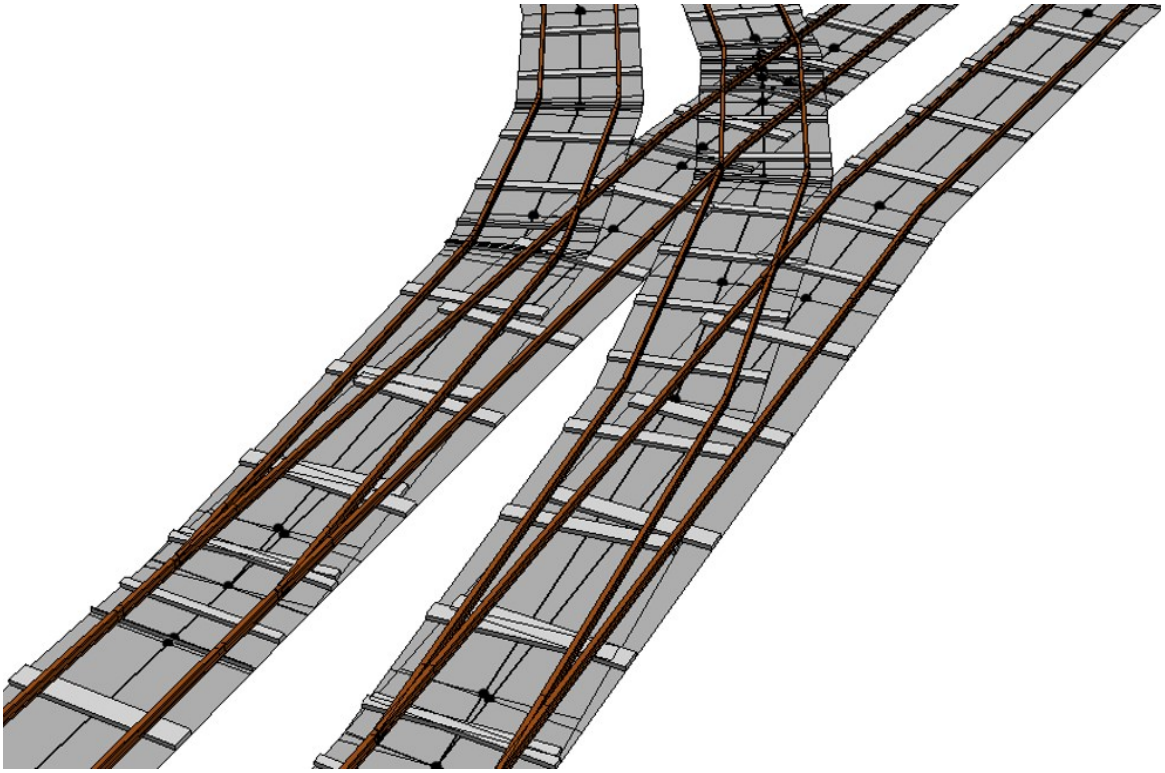

Abbildung 6.5 – Zwei Weichen mit anschließender Gleiskreuzung

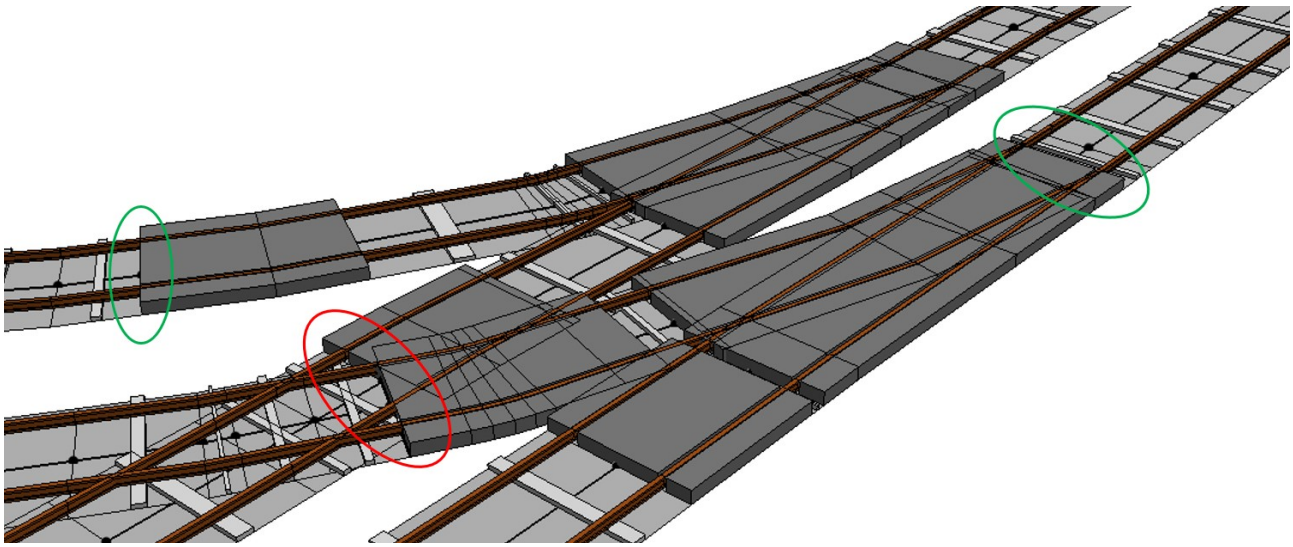

Abbildung 6.6 – Zwei Weichen mit anschließender Gleiskreuzung und asphaltierten Abschnitten

In der Abbildung 6.7 ist eine zweispurige Schienentrasse mit Bahnsteigen zu erkennen. Die Bahnsteige werden mittels der Regel *Bahnsteig.cga* parallel zu den Schienen erzeugt. Auch hier verläuft die Schienentrasse in einer leichten Kurve, was die Überlappungen der Bahnsteige an der Kurveninnenseite (grün markiert) und die Entstehung von Spalten an der Kurvenaußenseiten (rot markiert) zur Folge hat. Die Modellierung der Bahnsteige mittels der Regel *Bahnsteig.cga* erfolgt jedoch nicht für alle Bahnsteige fehlerlos. Das Problem besteht darin, dass die Bahnsteige nicht immer auf die richtige Seite der Schienen modelliert werden. Die Abbildung 6.8 zeigt einen solchen Fall. Ebenso wie bei der vorherigen Abbildung handelt es sich um eine zweispurige Schienentrasse, wobei die erste Hälfte des Bahnsteigs richtigerweise parallel zur Schienentrasse modelliert wird, wohingegen die zweite Hälfte des Bahnsteigs fälschlicherweise auf der Schienentrasse und nicht parallel dazu erzeugt wird. Das Problem konnte im Rahmen dieser Arbeit nicht gelöst werden.

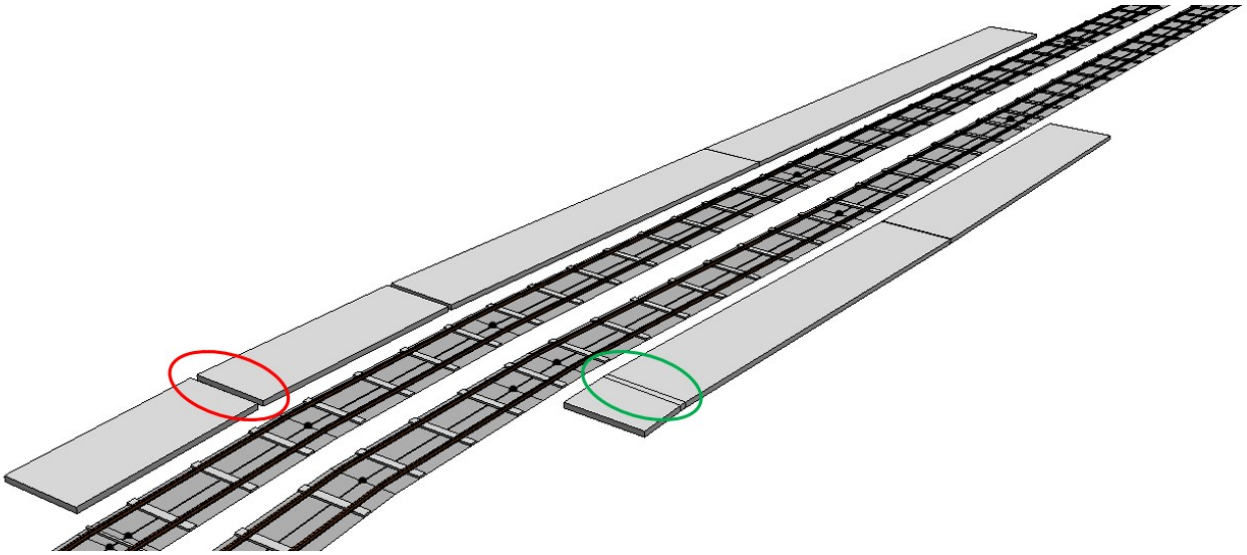

Abbildung 6.7 – Zweispurige Schienentrasse mit Bahnsteigen

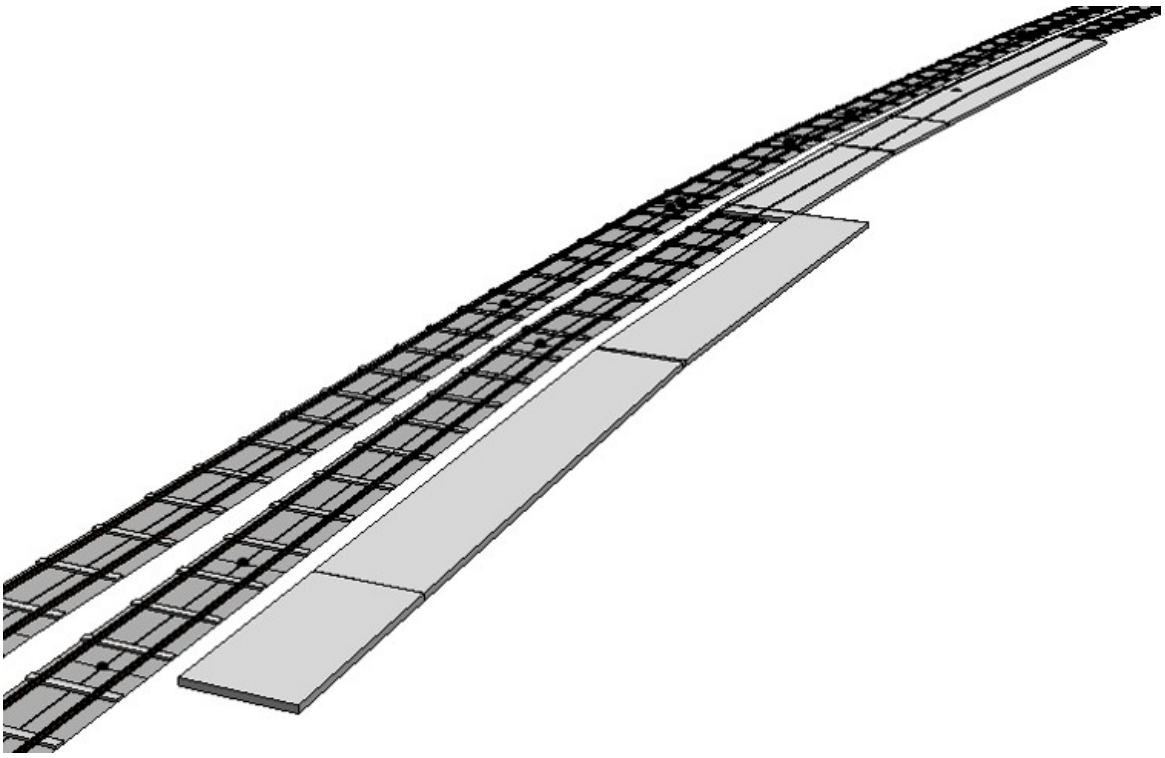

Abbildung 6.8 – Zweispurige Schienentrasse mit falsch modelliertem Bahnsteig

In den Abbildungen 6.9 und 6.10 sind die Bahnsteigpolygone gemeinsam mit der zweispurigen Schienentrasse zu sehen. Die Bahnsteigpolygone werden als eigenständige Objekte und nicht gemeinsam mit der Schienentrasse modelliert, somit ist deren Lage vorgegeben und nicht variabel. Die Schienentrasse wird mit einer Breite von 2,40 m modelliert. In beiden Abbildungen kommt es bei den rechts liegenden Bahnsteigen zu einer geringen Überlappung der Bahnsteige und der Schienentrasse. Die Abbildung 6.9 zeigt einen manuell korrigierten Bahnsteig (vgl. Kapitel 5.1.3).

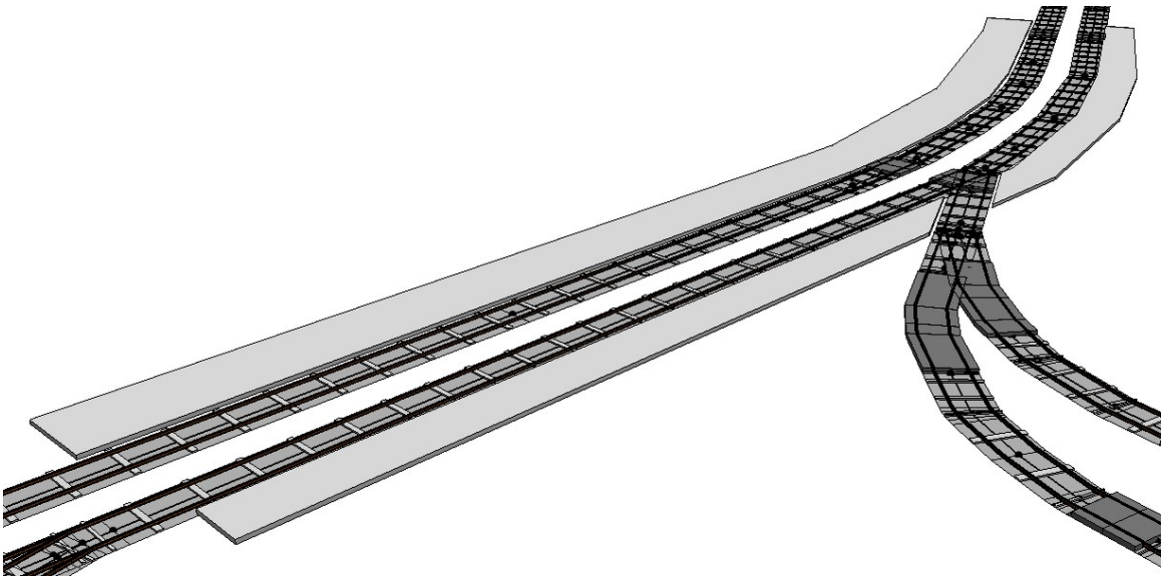

Abbildung 6.9 – Schienentrasse mit geteiltem Bahnsteigpolygon

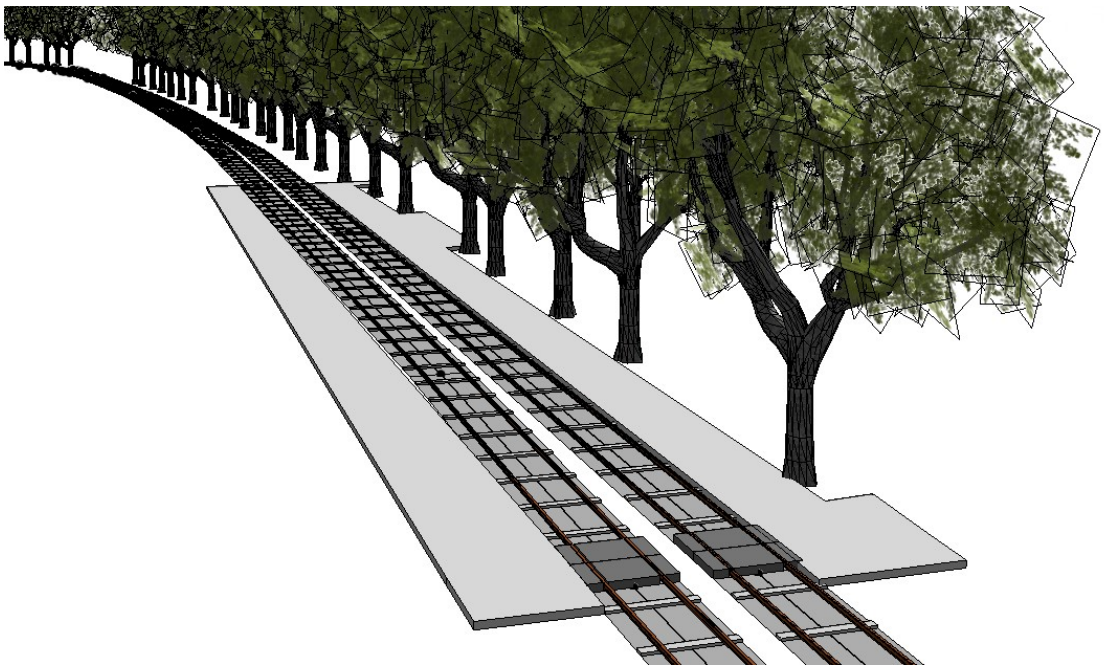

Abbildung 6.10 – Schienentrasse mit Bahnsteigpolygonen

Um die Lage und Länge der einzelnen Bauelemente in der Schienentrasse zu erkennen, werden diese in den nachfolgenden Abbildungen 6.11 und 6.12 farbig visualisiert. Die Darstellung der normalen Schienen erfolgt in grün; die Weichen und Gleiskreuzungen werden in rot dargestellt, die Bahnübergänge in blau und die Bahnsteige in gelb. Die Abbildung 6.11 zeigt eine Weiche und eine Gleiskreuzung, die sehr nah aufeinander folgen. Die beiden Elemente werden durch einen kurzen Abschnitt normaler Schienen verknüpft, der neben den Schienen aus einer sehr schmalen Bahnschwelle besteht. In der Abbildung 6.12 sind zwei Bahnübergänge und ein Bahnsteig enthalten, die durch normale Schienen verbunden werden.

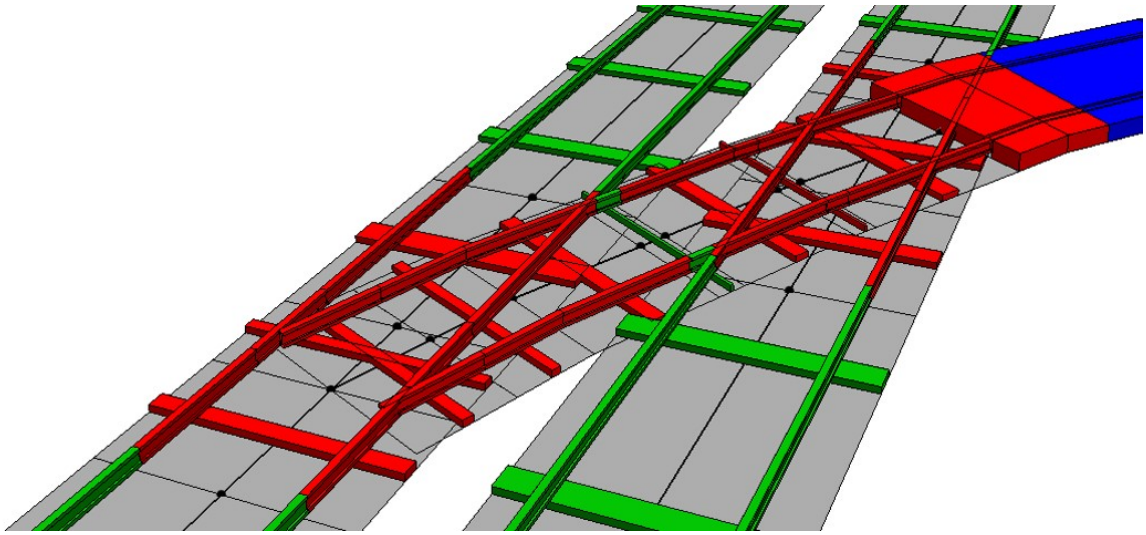

Abbildung 6.11 – Schienentrasse mit Bahnsteigpolygonen

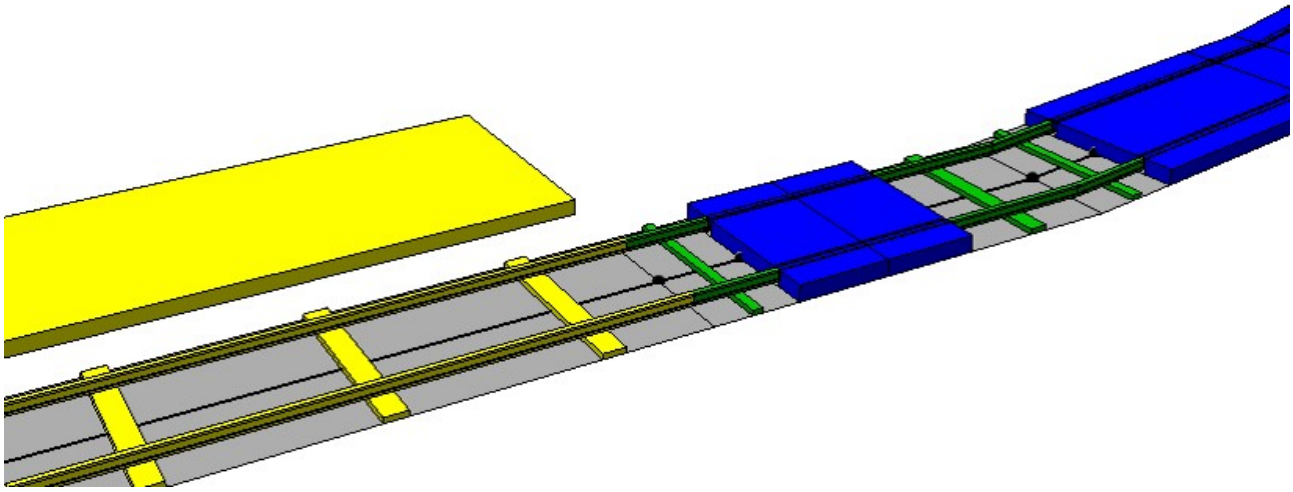

Abbildung 6.12 – Schienentrasse mit Bahnsteigpolygonen

Für die Darstellung der Bahnübergänge in den Abbildungen 6.13 und 6.14 wird das Modell der Bahnübergänge um die Modelle der kreuzenden Straßen ergänzt. In der Abbildung 6.13 kreuzen sich die Schienen- und die Straßentrasse nahezu in einem rechten Winkel. Der Übergang zwischen Schiene und Straße verläuft hier annähernd fließend. Wohingegen der Schnittwinkel zwischen Schienen- und Straßentrasse in der Abbildung 6.14 deutlich geringer ist. Die Folge ist ein unrealistischer Übergang zwischen Schiene und Straße, der große Lücken und starke Überlagerungen der Straße mit dem Element Bahnübergang aufweist.

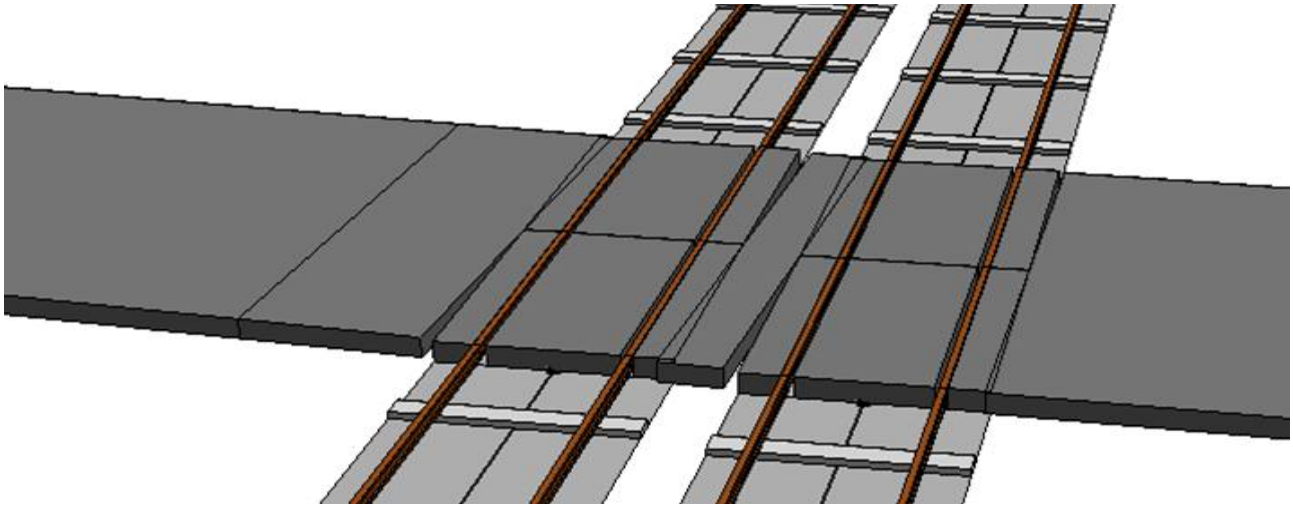

Abbildung 6.13 – Nahezu rechtwinklige Kreuzung der Straßen- und Schienentrasse

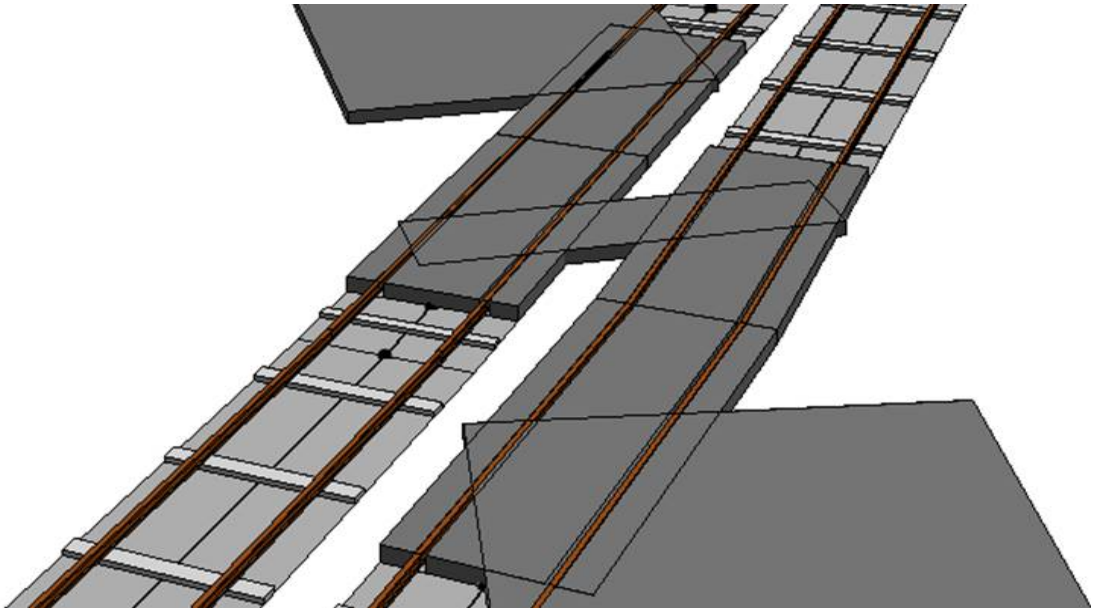

Abbildung 6.14 – Zweispurige Schienentrasse mit Bahnübergängen und kreuzender Straße

6.2 Vergleich

In diesem Kapitel wird die in dieser Arbeit entwickelten Methode der manuellen Erzeugung eines 3D-Modells für einen Straßenbahnsimulator der Müller Systemtechnik GmbH gegenübergestellt. Hierzu findet auch ein Vergleich der 3D-Modelle beispielhaft für drei Szenarien der Teststrecke statt.
Den ersten Unterschied zwischen den beiden Methoden stellt die Datengrundlage dar. Der Firma Müller Systemtechnik GmbH dienen unter anderem Orthophotos, topografische Karten, Stadtpläne, Begehungen und 3D-Stadtmodelle als Datengrundlage (vgl. Kapitel 1.3), wohingegen die vorliegende Arbeit ausschließlich auf OSM-Daten basiert. Die Müller Systemtechnik GmbH verwendet zur Erzeugung ihres 3D-Modells eine 3D-Modellierungssoftware, z.B. den Creator der Firma Presagis und verfügt zudem über eigene Software. Die vorliegende Arbeit nutzt die Software FME des Unternehmens Safe Software Inc. zur Aufbereitung der Daten und die CityEngine von ESRI zur Erzeugung des 3D-Modells sowie ArcMap von ESRI für manuelle Korrekturen. Weiterhin erzeugt die Müller Systemtechnik GmbH die für das Modell benötigte Geometrie selbst, die vorliegende Arbeit verwendet die in den OSM-Daten hinterlegte Geometrie. In beiden Modellen wird auf die Verwendung eines DGMs verzichtet. Den größten Unterschied stellt das Verfahren der Erzeugung dar. Die Firma Müller Systemtechnik GmbH erstellt ihr 3D-Modell semi-manuell, wohingegen die Erzeugung des Modells in dieser Arbeit automatisiert und mithilfe prozeduraler Modellierung erfolgt.
Die nachfolgenden Abbildungen 6.15, 6.16 und 6.17 zeigen drei Ausschnitte der Teststrecke. Jede Abbildung setzt sich aus drei Graphiken zusammen: einem Foto, dass die realen Gegebenheiten darstellt, der dazugehörigen Szene aus dem 3D-Modell der Müller Systemtechnik GmbH und dem entsprechenden Ausschnitt des in dieser Arbeit erzeugten Modells.
Ein 1:1-Vergleich der 3D-Modelle ist hier jedoch nicht möglich, da im Rahmen dieser Arbeit das

3D-Modell der Schienentrasse im Fokus steht. Um den bildlichen Vergleich für den Betrachter aufzuwerten, wird das Modell der Schienentrasse um Gebäudemodelle und Straßen ergänzt. Dabei handelt es sich um generische Gebäudemodelle, die auf den in den OSM-Daten enthaltenen Gebäudegrundrissen erzeugt werden. Die zugrunde liegende CGA-Regel stammt aus der in CityEngine integrierten ESRI-Bibliothek. Die Darstellung der Straßen erfolgt durch die Pufferung der in OSM gegebenen Straßenlinien (Straßenbreiten siehe Kapitel 5.1.2), wobei die Straßen über keine Straßenmarkierungen oder sonstigen Details verfügen.
In der Abbildung 6.15 ist eine Weiche mit darauffolgendem Bahnübergang dargestellt. Beide Modelle enthalten den Weichenverlauf und den darauffolgenden Bahnübergang. Im Modell der Firma Müller Systemtechnik GmbH werden die Schienen- und Straßentrasse durch Texturen dargestellt, die vorliegende Arbeit verwendet mittels prozeduraler Modellierung erzeugte Geometrien. Die dargestellte Weiche verläuft im Modell der Müller Systemtechnik GmbH geschwungener, wie es auch in der Realität der Fall ist. Wohingegen die Weiche des in dieser Arbeit erstellten Modells „eckiger“ erscheint, was auf die fehlende Fähigkeit der CityEngine zur Darstellung von Kurven zurückgeht.
Die Abbildung 6.16 zeigt eine Kreuzung mit zweispuriger Schienentrasse und linksabbiegenden Weichen. Wieder wird der Verlauf der Schienentrasse in beiden Modellen korrekt dargestellt. Aufgrund der vielen kreuzenden Straßen enthält das in dieser Arbeit erstellte Modell eine Vielzahl an Bahnübergängen, wohingegen der gesamte Untergrund des Schienenmodells der Firma Müller Systemtechnik GmbH asphaltiert dargestellt wird.
In der Abbildung 6.17 sind eine einspurige Schienentrasse und ein Bahnübergang dargestellt. Die Schiene läuft in eine leichte Rechtskurve. Beide Modelle bilden dies ab.

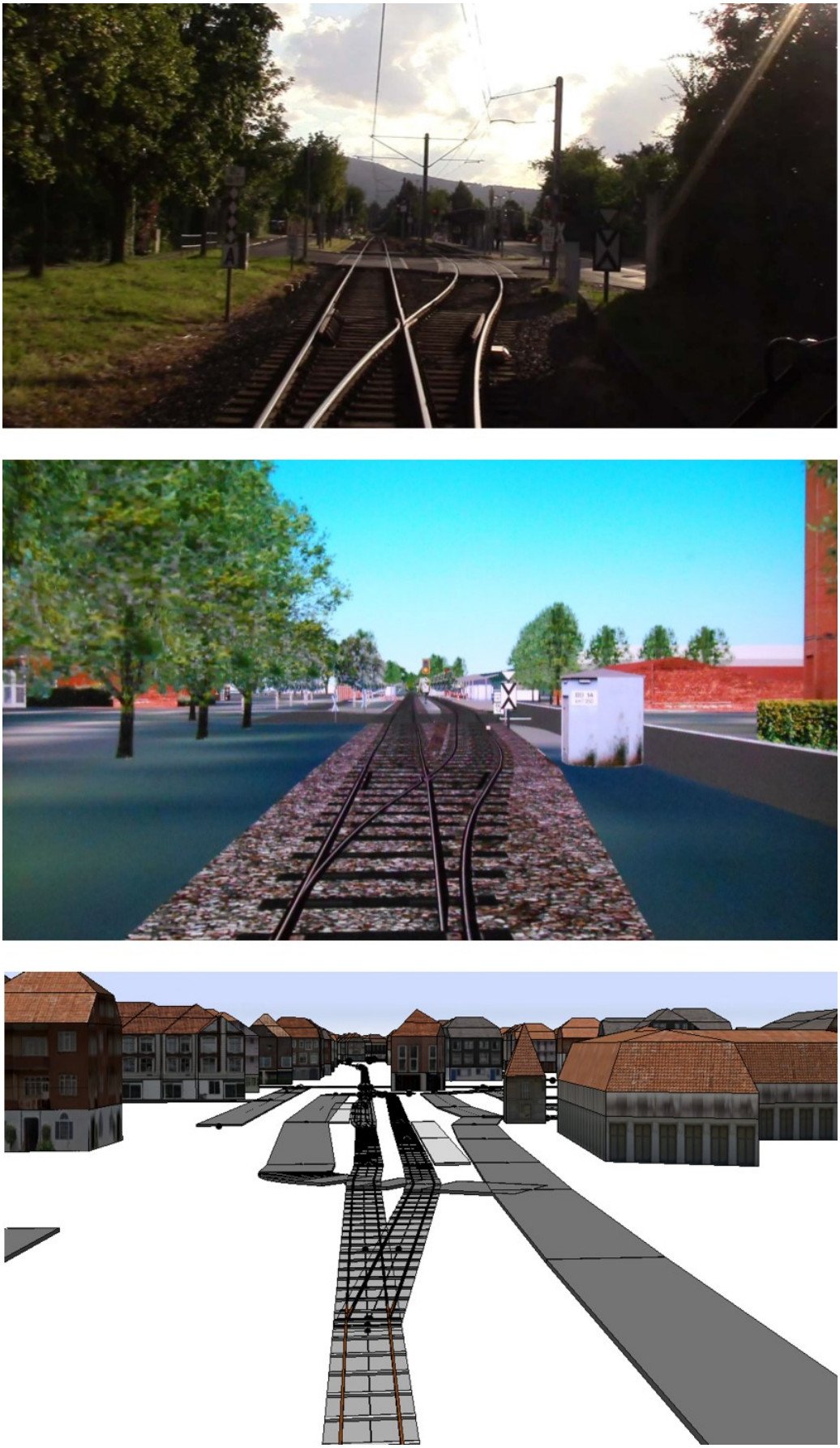

Abbildung 6.15 – Foto einer Weiche (oben) [Müller, 2015a], Weiche im Modell der Müller Systemtechnik GmbH (mittig) [Müller, 2015a] und Weiche im Modell dieser Arbeit (unten)

Abbildung 6.16 – Foto einer zweispurigen Schienentrasse mit Weiche (oben) [Müller, 2015a], zweispurige Schienentrasse mit Weiche im Modell der Müller Systemtechnik GmbH (mittig) [Müller, 2015a] und zweispurige Schienentrasse mit Weiche im Modell dieser Arbeit (unten)

Abbildung 6.17 – Foto einer einspurigen Schienentrasse (oben) [Müller, 2015a], einspurige Schienentrasse im Modell der Müller Systemtechnik GmbH (mittig) [Müller, 2015a] und einspurige Schienentrasse im Modell dieser Arbeit (unten)

6.3 Verbesserungsmöglichkeiten

Die Vollständigkeit der Datengrundlage ist für die realitätsnahe Erzeugung eines 3D-Modells von großer Bedeutung.
So würde die flächendeckende Erfassung der Beschaffenheit der Straßenbahntrasse (z.B. Holz- oder Betonschwelle, asphaltierter Untergrund der Schienen oder Schotteruntergrund) die Qualität des Modells deutlich verbessern.
In Bezug auf die hier „untersuchte“ Schienentrasse würde das Modell einerseits an Realitätsnähe gewinnen, andererseits könnten bestehende Probleme behoben werden. So könnte die Problematik der „zackigen“ Übergänge bei Weichen und Gleiskreuzungen zwischen Bahnübergang und Schiene gelöst werden.
Wichtig ist zudem die Vervollständigung der Bahnsteige. Die fehlenden Objekte und Attribute könnten entweder in den OSM-Daten ergänzt werden oder die OSM-Daten könnten durch alternative Datenquellen komplettiert werden. Dies gilt nicht nur bezüglich der Straßenbahntrasse, sondern für das gesamte Modell, zu dem unter anderem Straßen, Gebäude, Verkehrsampeln und Vegetation zählen.

Des Weiteren sollte die Erstellung des Bauelements Bahnsteig in FME verbessert werden. Diese sollte so angepasst werden, dass auch die beidseitige Erzeugung von Bahnsteigen auf einspurigen Straßenbahntrassen möglich ist.

Ferner sollte der Übergang zwischen den Straßen und Schienen, im Bereich der Bahnübergänge, fließender werden. Hierfür könnte eine Art Duplikat der Schienentrasse für die Breite der Straße erstellt werden. Dieses könnte von den Straßen subtrahiert werden, sodass der Ausschnitt in den Straßen genau dem Verlauf der Schienentrasse entspricht.

Eine deutliche Verbesserung wäre zudem, wenn in der CityEngine Kurven darstellbar wären. Die geringen Polygone und die darauf entstehenden Modelle (z.B. Schwellen mit sehr geringer Breite) stellen aufgrund ihrer mangelnden Realitätsnähe ein Negativpunkt dar. Eine Möglichkeit wäre, die Anzahl der Knotenpunkte vor der Modellierung in CityEngine zu reduzieren. Eine Option hierzu wäre, die Elemente einer „merge“-Operation zu unterziehen. Weiterhin ist die Vermeidung kurzer Abschnitte normaler Schienen zwischen den einzelnen Elementen (z.B. zwischen Weiche und Gleiskreuzung) erstrebenswert. Ein direkter Übergang zwischen diesen wäre von Vorteil.

7 Fazit und Ausblick

Ziel dieser Arbeit war es, die vorwiegend manuelle Generierung des 3D-Modells für einen Straßenbahnsimulator der Firma Müller Systemtechnik GmbH mithilfe prozeduraler Modellierung soweit wie möglich zu automatisieren und zu untersuchen, inwieweit eine Automatisierung sinnvoll ist. Dies erfolgte am Beispiel der Tramlinie 5 in Kassel zwischen den Haltestellen „Großenritte“ und „Auestadion“. Als Datengrundlage wurden OSM-Rohdaten genutzt, die in der Software FME aufbereitet wurden. Des Weiteren wurde die Schienentrasse mittels FME in Bauelemente zerlegt. Anschließend erfolgte die Generierung des 3D-Modells der Schienentrasse in der Software CityEngine. Hierfür wurde für jedes Bauelement ein Modellierungskonzept in Form einer CGA-Regel erstellt.

Neben den Vorteilen, die eine Datengrundlage wie OSM mit sich bringt - z.B. die Aktualität der Daten oder die teils detailreiche Beschreibung der Objekte -,ergeben sich auch Schwierigkeiten. So hat die Datenaufbereitung in FME unerwartet viel Zeit in Anspruch genommen. Grund dafür waren das Einarbeiten in die OSM-Daten und deren Umstrukturierung. Überdies stellt die partielle Unvollständigkeit der Daten eine Schwierigkeit dar. Alles in allem stellte OSM für die Teststrecke, insbesondere für die Schienentrasse, eine solide Datengrundlage dar. Für eine weiterführende Modellierung des Modells (Gebäude, Straßen, Ampeln, Straßenbahnsignale etc.) ist eine Verbesserung der Datengrundlage zwingend notwendig. Dies kann entweder durch Komplettierung der OSM-Daten selbst geschehen oder durch das Nutzen alternativer Datenquellen.
Die Zerlegung der Schienentrasse in Bauelemente wurde ebenfalls mit FME durchgeführt. Hierbei wurden positive Ergebnisse für die Bauelemente normale Schiene, Bahnsteigpolygon, Weiche und Gleiskreuzung sowie Bahnübergang und Fußbahnübergang erzielt. Lediglich die Erstellung des Bauelements Bahnsteig sollte verbessert werden, sodass zusätzlich die Erstellung beidseitiger Bahnsteige an eingleisigen Schienentrassen möglich ist.

Im Rahmen dieser Masterarbeit lag das Hauptaugenmerk der 3D-Modellierung mit CityEngine auf der Schienentrasse. Hierfür wurden Modellierungskonzepte in Form von CGA-Regeln für die Bauelemente erstellt. Insgesamt wurden in CityEngine fünf CGA-Regeln erstellt: für die normalen Schienen, die Bahnübergänge, die Weichen und Gleiskreuzungen, die Bahnsteige und die Bahnsteigpolygone. Die Anwendung der CGA-Regeln auf die Schienentrasse liefert anschauliche Ergebnisse. Die Modellierung der Bahnsteige - durch Anwendung der Regel für Bahnsteige - auf eine scheinbar beliebige Seite konnte im Rahmen der Masterarbeit nicht behoben werden. Eine weitere Schwierigkeit stellt die Modellierung der Schienentrasse in Kurven dar. In der CityEngine werden Kurven derzeit durch Geraden angenähert. Für eine realitätsnähere Darstellung ist eine Anpassung der Software CityEn-

gine wünschenswert.
Die Software CityEngine ist eine komplexe Software, deren vollständige Erfassung in kurzer Zeit eine Herausforderung darstellt. Selbes gilt für die Erstellung der CGA-Regeln.

Das resultierende 3D-Modell der Schienentrasse liefert insgesamt ein anschauliches Ergebnis. Der Ansatz der prozeduralen Modellierung stellt eine interessante Alternative zur manuellen Modellierung dar. Dabei ist zu beachten, dass die Datenaufbereitung und das Erstellen der CGA-Regeln verhältnismäßig viel Zeit in Anspruch nehmen. Wenn diese jedoch erstellt sind, erfolgt die 3D-Modellierung innerhalb kürzester Zeit. Zudem muss berücksichtigt werden, dass die Erstellung der Regeln einmalig erfolgt und diese anschließend wieder verwendet werden können. Einen weiteren Vorteil bietet die Option, Korrekturen verhältnismäßig schnell durchführen zu können. Beispielsweise kann die Spurbreite mit dem Anpassen des entsprechenden Parameters innerhalb kurzer Zeit für die gesamte Schienentrasse durchgeführt werden. Die Realitätsnähe kann durch die Verwendung einer vollständigen Datengrundlage gesteigert werden (Beschaffenheit der Schienentrasse, Darstellung aller Bahnsteige etc.).
Eine vollständige Automatisierung ist zur Zeit jedoch nicht möglich. Sinnvoll wäre eine partielle Automatisierung mit der Möglichkeit gezielten manuellen Eingreifens. So könnten z.B. die Schienentrasse und die Straßentrasse prozedural erzeugt werden, deren Zusammenführung jedoch manuell erfolgen, um somit ein realitätsnahes Ergebnis zu erhalten.

Abschließend kann gesagt werden, dass das Ziel dieser Masterarbeit erreicht wurde. Anhand einer Teststrecke wurde mittels prozeduraler Modellierung ein 3D-Modell erstellt und geprüft, inwieweit eine Automatisierung sinnvoll ist. Weiterführend gilt es Lösungsansätze für die bestehenden Probleme zu entwickeln. Zukünftig könnte beispielsweise der Übergang zwischen Schienen- und Straßentrasse untersucht werden, die prozedurale Modellierung des Modells mit einer verbesserten Datengrundlage durchgeführt werden und weitere CGA-Regeln für die Vervollständigung des 3D-Modell erstellt werden.

Literaturverzeichnis

[Bildstein 2005] BILDSTEIN, F: 3D City Models for Simulation and Training–Requirements on Next Generation 3D City Models. In: *Proc. of the Int. ISPRS Workshop on Next Generation 3D City Models in Bonn, Germany. EuroSDR publication* Bd. 2, 2005

[Esri R&D Center Zurich 2016] ESRI R&D CENTER ZURICH: *CityEngine Overview*. `http://cehelp.esri.com/help/index.jsp`. Version: 2016. – Stand: 22.07.2016

[con terra GmbH 2015] GMBH con t. (Hrsg.): *FME Desktop - Das deutschsprachige Handbuch für Einsteiger und Anwender*. Berlin : Wichmann, 2015

[Grippenkoven u. a. 2014] GRIPPENKOVEN, Jan ; SCHNIEDER, Ing L. ; NAUMANN, Anja ; JÄGER, Ing B.: Aufbau eines modularen Straßenbahnführerstandes für simulationsbasierte Arbeitsplatzuntersuchungen / Deutsches Zentrum für Luft- und Raumfahrt e.V., Institut für Verkehrssytemtechnik. 2014. – Forschungsbericht

[Hölscher 2015a] HÖLSCHER, Carsten: *Zusi Bahnsimulatoren - Gebäudeeditor*. `http://www.zusi.de/pages/dehaupt/zusi-2/programm/komponenten/gebE4udeeditor.php`. Version: 2015. – Stand: 08.12.2015

[Hölscher 2015b] HÖLSCHER, Carsten: *Zusi Bahnsimulatoren - Konzept*. `http://www.zusi.de/pages/dehaupt/zusi-2/programm/konzept.php`. Version: 2015. – Stand: 08.12.2015

[Immler 2014] IMMLER, Walter: *Das OpenStreetMap Handbuch*. Franzis Verlag GmbH, 2014

[Kelly u. McCabe 2006] KELLY, George ; MCCABE, Hugh: A survey of procedural techniques for city generation. In: *Institute of Technology Blanchardstown Journal* 14 (2006), S. 87–130

[Lindenmayer 1968] LINDENMAYER, Aristid: Mathematical models for cellular interactions in development Parts I and II. In: *Journal of theoretical biology* 18 (1968), S. 280–315

[Měch u. Prusinkiewicz 1996] MĚCH, Radomír ; PRUSINKIEWICZ, Przemyslaw: Visual models of plants interacting with their environment. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* ACM, 1996, S. 397–410

[Müller 2015a] MÜLLER, Klaus R.: Anforderungen zur semi-automatischen Generierung eines Stadtmodells für Straßenbahnsimulatoren. (2015). – interne Diskussionspapiere

[Müller 2015b] MÜLLER, Klaus R.: Simulationstrainer - Das Ausbildungsmittel für den Lok- und Triebwagenführer. (2015). – interne Diskussionspapiere

[Müller 2016] MÜLLER, Klaus R.: *Gespräch und E-Mail-Kontakt mit Herrn Müller. Müller Systemtechnik GmbH*. 2016

[Müller u. a. 2006] MÜLLER, Pascal ; WONKA, Peter ; HAEGLER, Simon ; ULMER, Andreas ; VAN GOOL, Luc: Procedural modeling of buildings. In: *ACM Transactions On Graphics (Tog)* 25 (2006), Nr. 3, S. 614–623

[OpenStreetMap 2016] OPENSTREETMAP: *OpenStreetMap - Deutschland - FAQs*. `https://www.openstreetmap.de/faq.html`. Version: 2016. – Stand: 19.08.2016

[OpenStreetMap-Mitwirkende 2016] OPENSTREETMAP-MITWIRKENDE: *OpenStreetMap*. `https://www.openstreetmap.org`. Version: 2016. – Stand: 16.08.2016

[Parish u. Müller 2001] PARISH, Yoav I. ; MÜLLER, Pascal: Procedural modeling of cities. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* ACM, 2001, S. 301–308

[Prusinkiewicz u. a. 1994] PRUSINKIEWICZ, Przemyslaw ; JAMES, Mark ; MĚCH, Radomír: Synthetic topiary. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* ACM, 1994, S. 351–358

[Prusinkiewicz u. Lindenmayer 1990] PRUSINKIEWICZ, Przemyslaw ; LINDENMAYER, Aristid: The algorithmic beauty of plants (the virtual laboratory). (1990)

[Ramm u. Topf 2010] RAMM, Frederik ; TOPF, Jochen: *OpenStreetMap*. 3. Berlin : Lehmanns Media, 2010

[Smelik u. a. 2014] 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, 2014, S. 31–50

[Wonka u. a. 2003] WONKA, Peter ; WIMMER, Michael ; SILLION, François ; RIBARSKY, William: *Instant architecture*. ACM, 2003

[Würzburger Institut für Verkehrswissenschaften GmbH 2014a] WÜRZBURGER INSTITUT FÜR VERKEHRSWISSENSCHAFTEN GMBH: *Anwendungsbereiche*. `https://wivw.de/de/silab/anwendungsbereiche`. Version: 2014. – Stand: 08.12.2015

[Würzburger Institut für Verkehrswissenschaften GmbH 2014b] WÜRZBURGER INSTITUT FÜR VERKEHRSWISSENSCHAFTEN GMBH: *Fahrsimulation und SILAB*. `https://wivw.de/de/silab`. Version: 2014. – Stand: 08.12.2015

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

2D	zweidimensional
3D	dreidimensional
DLR	Deutsche Zentrum für Luft- und Raumfahrt e.V.
DGM	digitales Geländemodell
ESRI Inc.	Environmental Systems Research Institute
FME	Feature Manipulation Engine
ODbL	Open Database Licence
OSM	OpenStreetMap
WIVW	Würzburger Institut für Verkehrswissenschaften GmbH
XML	Extensible Markup Language

Anhang

Siehe Daten-CD