Entwicklung und Erprobung eines interaktiven 3D - Stadtmodells am Beispiel des Personennahverkehrsnetzwerks der Stadt Frankfurt

Bachelorarbeit

zur Erlangung des akademischen Grades eines

Bachelor of Science

im Studiengang Informatik

vorgelegt von

Alen Smajić

angefertigt am
Lehrgebiet Texttechnologie
der Goethe-Universität Frankfurt am Main

Betreuer: Prof. Dr. Alexander Mehler

27.05.2020

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Ebenso bestätige ich, dass diese Arbeit nicht, auch nicht auszugsweise, für eine andere Prüfung oder Studienleistung verwendet wurde.

Frankfurt am Main, den 27.05.2020

_____

Unterschrift

Zusammenfassung

Das Ziel dieser Arbeit ist die realitätsgetreue Entwicklung eines interaktiven 3D-Stadtmodells, welches auf den ÖPNV zugeschnitten ist. Dabei soll das Programm anhand von Benutzereingaben und mit Hilfe einer Datenquelle, automatisch eine dreidimensionale Visualisierung der Gebäude erzeugen und den lokalen ÖPNV mitintegrieren. Als Beispiel der Ausarbeitung diente das ÖPNV-Netz der Stadt Frankfurt. Hierbei wurde auf die Problematik der Erhebung von Geoinformationen und der Verarbeitung von solchen komplexen Daten eingegangen. Es wurde ermittelt, welche Nutzergruppen einen Mehrwert durch eine derartige 3D Visualisierung haben und welche neuen Erweiterungs- und Nutzungspotenziale das Modell bietet.

Dem Leser soll insbesondere ein Einblick in die Generierung von interaktiven 3D-Modellen aus reinen Rohdaten verschafft werden. Dazu wurde als Entwicklungsumgebung die Spiele-Engine *Unity* eingesetzt, welche sich als sehr fähiges und modernes Entwicklungswerkzeug bei der Erstellung von funktionalen 3D-Visualisierungen herausgestellt hat. Als Datenquelle wurde das OpenStreetMap Projekt benutzt und im Rahmen dieser Arbeit behandelt. Anschließend wurde zur Evaluation, das Modell verschiedenen Nutzern bereitgestellt und anhand eines Fragebogens evaluiert.

Danksagung

Ich würde mich bei all jenen bedanken, welche mich während der Ausarbeitung dieser Arbeit unterstützt und begleitet haben.

Ich bedanke mich bei meinem Themenbetreuer, Herrn Prof. Dr. Alexander Mehler, welcher mir ermöglicht hat mein Wunschthema auszuarbeiten. Des Weiteren bedanke ich mich bei Herrn Giuseppe Abrami für seine Unterstützung und Ermutigung in der Entwicklungsphase.

Ein besonderer Dank geht ebenfalls an Herrn Sloan Kelly, dessen Algorithmus zur Datenerhebung und Datenverarbeitung in diesem Projekt angewandt wurde. Bei der Firma Mapbox bedanke ich mich für die Bereitstellung der kostenfreien Schnittstelle zur 3D-Gebäudeintegration. Darüber hinaus bedanke ich mich beim OpenStreetMap Projekt und bei allen Teilnehmern, welche sich aktiv am Projekt beteiligen. Die daraus resultierenden Daten haben die Realisierung dieses Projekts erst möglich gemacht.

Anschließend ein großes Dankeschön an den YouTube-Kanal „ÖPNV Simulationen“ und an dessen Community, mit deren Kooperation die Evaluation der Anwendung durchgeführt wurde.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

3D	Dreidimensional
API	Application programming interface
AR	Augmented reality
AVV	Aachener Verkehrsverbund
CityGML	City Geography Markup Language
DB	Deutsche Bahn
DGM	Digitales Geländemodell
GDI-NRW	Geodateninitative Nordrhein-Westafalen
GML3	Geography Markup Language 3
ISO	International Organization for Standardization
LOD	Level of detail
OSM	OpenStreetMap
ÖPNV	Öffentlicher Personennahverkehr
RMV	Rhein-Main-Verkehrsverbund
SDK	Software development kit
SIG 3D	Special Interest Group 3D
UI	User interface
UMUX	Usability Metric for User Experience
VR	Virtual reality
VRS	Verkehrsverbund Rhein-Sieg
XML	Extensible Markup Language

Kapitel 1
Einleitung

1.1 Motivation

Die Visualisierung von virtuellen 3D-Stadtmodellen hat in den letzten Jahrzehnten einen bedeutsamen Zuwachs sowohl in der Anwendung als auch in der Qualität verzeichnet und Dank der Digitalisierung und dem technologischen Fortschritt eine bedeutsame Rolle in der Umsetzung von diversen Projekten eingenommen. Virtuelle 3D-Stadtmodelle bieten dem Anwender eine realitätsgetreue Perspektive, mit dessen Hilfe bauliche Vorhaben sowie andere Problemstellungen durch eine Simulation visualisiert werden können. Dabei sind einige Anwendungsbereiche ohne derartige Modelle fast schon undenkbar. Beispielsweise ermöglicht die Rekonstruktion von historisch verschwundenen Bauwerken einen einzigartigen Einblick in die Geschichte der Menschen. Moderne Navigationssysteme enthalten vermehrt 3D-Stadtmodelle zur erleichterten Orientierung. Auch die Unterhaltungsbranche profitiert von dieser Technologie enorm. „Realfilme nutzen regelmäßig computergenerierte Städte als Kulisse. So besitzt man alle Freiheiten hinsichtlich der Beleuchtung und Effekte. Auch Kamerafahrten, die in Wirklichkeit nahezu unmöglich wären, lassen sich so darstellen“ [17]. Ähnliches gilt für die Videospielindustrie.

Die Entwicklung von virtuellen Städten befindet sich noch immer in ihrer Anfangsphase und verzeichnet jedes Jahr erstaunliche Fortschritte. Immer mehr Daten und Datenquellen werden den Entwicklern zur Verfügung gestellt und auch der Anwendungsbereich nimmt weiter zu. Diese Arbeit soll einen Beitrag zur Weiterentwicklung in diesem Gebiet leisten und die Anwendung von derartigen Modellen speziell für den ÖPNV veranschaulichen.

1.2 Aufgabenstellung und Zielsetzung

Im Zuge dieser Arbeit wird auf die Problematik der Entwicklung eines interaktiven 3D-Stadtmodells für den ÖPNV und dessen Evaluation eingegangen, wobei das ÖPNV-Netzwerk der Stadt Frankfurt als Beispiel der Ausarbeitung dienen soll. Der Anwender soll sich innerhalb eines 3D-Stadtmodells frei bewegen und mit den einzelnen Stationsobjekten interagieren können, um lokale ÖPNV-Informationen zu erhalten. Dabei soll sich der Nutzer möglichst gut innerhalb der Anwendung orientieren können. Des Weiteren sollen einzelne Simulationselemente integriert werden, um die Nutzungspotenziale und die Erweiterbarkeit der Anwendung zu demonstrieren.

Die Anforderungen an die Anwendung lauten:

- Benutzerfreundlich

- Dynamisch (unterstützt eine Vielzahl von Orten)
- Optimale Abstrahierung und Übersichtlichkeit (dem Nutzer nur das zeigen, was er sehen will)
- Hohes Maß an Orientierung und somit eine gute Alternative zu den aktuell vorhandenen Orientierungshilfsmitteln
- Verschiedene Interaktionsmöglichkeiten (z.B. Nutzer kann schnell an gewünschte Informationen herankommen)
- 3D-Modellierung der bewohnten Orte
- Effizienz
- Akkurate Wiederspiegelung der Wirklichkeit
- Aktualisierungsfähig (die Veränderungen der Welt, bedingt durch Zeit, werden ins Modell übernommen)
- Raum für Ausbaumöglichkeiten und Erweiterungen
- Simulierbares Modell

Das Ziel ist es eine qualitative und in sich abgeschlossene Anwendung zu entwickeln, welche dem aktuellen Stand der Technik entspricht und in ihrer Ausprägung einen Mehrwert für verschiedene Nutzergruppen bieten kann. Dabei soll verglichen werden, welche Vorteile ein 3D-Stadtmodell gegenüber einem klassischen 2D-Modell bietet, hinsichtlich der Orientierungs- und Visualisierungseigenschaften. Des Weiteren wird auf die interaktiven Funktionalitäten eingegangen und der Vergleich zu klassischen statischen Modellen/Karten gezogen. Natürlich stellt sich auch die Frage, welche Datenquelle genutzt werden soll, unter Berücksichtigung, dass keine finanziellen Ressourcen für die Umsetzung des Projekts vorliegen und vollständig auf kostenfreie Angebote zurückgegriffen werden muss. Diese Datenquelle muss neben der Kostenfreiheit auch andere Aspekte erfüllen und für eine effiziente Datenverarbeitung optimiert sein. Dies führt zu einem weiteren wichtigen Aspekt: Die Arbeit soll einen Einblick in die Komplexität der Datenverarbeitung und der Wiedergabe eines interaktiven 3D-Stadtmodells aus reinen Daten verschaffen. Dabei wird insbesondere auf die Rechenleistung und Effizienz gesetzt. Am Ende soll das Modell durch verschiedene Nutzer evaluiert werden, wodurch einige Nutzungspotenziale und Nutzergruppen identifiziert werden sollen.

1.3 Struktur der Arbeit

In Kapitel 1 gibt es eine Einleitung in die Thematik der Arbeit. Dabei wird die Motivation des Projekts vorgestellt und daraus die Aufgabenstellung abgeleitet. Als Zielsetzung wird die Entwicklung eines interaktiven 3D-Stadtmodells festgelegt, welches auf den ÖPNV zugeschnitten ist. Dieses soll neben den funktionalen Eigenschaften auch weitere Möglichkeiten zur Erweiterung bieten und auf verschiedene Nutzergruppen ausgelegt werden. Des Weiteren werden grundlegende Anforderungen an die Anwendung formuliert. Im 2. Kapitel gibt es dann eine Analyse über verwandte Werke. Hierbei wird

insbesondere auf Werke, welche *OSM* als Datengrundlage nutzen, eingegangen, aber auch sonstige Werke, welche sowohl ein 3D-Stadtmodell enthalten als auch den lokalen ÖPNV visualisieren. Kapitel 3 fungiert als Grundlagen-Kapitel und soll eine Einführung in die Thematik der 3D-Stadtmodellierung bieten. Zugleich beinhaltet dieses Kapitel auch eine Übersicht der wichtigsten Entwicklungsressourcen, welche im Implementierungsprozess genutzt wurden. Im Zuge dessen wurde *OSM* als Datenquelle ausgewählt und im Rahmen dieser Arbeit behandelt. Des Weiteren gibt es eine kurze Einführung in die Funktionsweise der genutzten Entwicklungsumgebung, der Spiele-Engine *Unity*. Für die Generierung der 3D-Gebäude wurde ein durch die Firma *Mapbox* bereitgestelltes Softwarepaket verwendet. Dieses wird in Form einer Unity-Schnittstelle kostenfrei angeboten und am Ende des 3. Kapitels ebenfalls analysiert. Ab Kapitel 4 wird die Implementierung der Software vorgestellt. Dabei wird jeweils auf die verschiedenen Funktionalitäten der Anwendung eingegangen und der entwickelte Algorithmus analysiert. Dieser verarbeitet automatisch die übergebenen OSM-Daten und erzeugt mit dessen Hilfe das 3D-Stadtmodell. Im 5. Kapitel wird die Evaluation der Simulationsanwendung beschrieben. Dazu werden die Ergebnisse des Fragebogens ausgewertet und analysiert. Anschließend werden die daraus resultierenden Erkenntnisse für die Beschreibung der Nutzungspotenziale und der Weiterentwicklungsmöglichkeiten in Kapitel 6 verwendet und abschließend eine Zusammenfassung und Schlussbemerkung formuliert.

Kapitel 2
Verwandte Werke

2.1 OpenStreetMap-basierte Werke

Eine Vielzahl von 3D-Stadtmodellen nutzt vollständig oder zum Großteil *OpenStreetMap* als Datengrundlage. Bei *OpenStreetMap* handelt es sich um ein Open-Data Projekt, welches Geoinformationen kostenfrei an seine Nutzer bereitstellt und in Kapitel 3.2.1 näher behandelt wird. Die im Folgenden aufgeführten Werke sind auch auf der offiziellen Wiki-Seite von *OSM* dokumentiert und dienen als Übersicht über aktuelle Nutzungspotenziale hinsichtlich der Implementierung von 3D-Visualisierungen basierend auf *OSM* Daten [35].

2.1.1 blender-osm

Bei dieser Software [45] handelt es sich um eine kostenpflichtige Erweiterung für die 3D-Modellierungssoftware *Blender*. Diese nutzt *OpenStreetMap*-Daten, welche durch den Nutzer eingegeben werden können und verarbeitet die entsprechenden Informationen zu einer 3D-Darstellung, welche die geografische Landschaft und die Gebäude wiederspiegelt. Dabei werden dem Nutzer verschiedene Anpassungsmöglichkeiten und Funktionalitäten (im Bezug zur Landschaft und den Gebäuden) angeboten. Des Weiteren kann der Nutzer mit Hilfe der *Blender Tools* weitere individuelle Anpassungen an dem geladenen Modell vornehmen und dieses nach eigenen Wünschen bearbeiten.

2.1.2 OSM Buildings

Bei *OSM Buildings* [29] handelt es sich um einen kostenfreien Web Service, welcher von Jan Marsch entwickelt wurde. Dieser ermöglicht dem Anwender eine schnelle 3D-Visualisierung der Gebäude in Form einer Webkarte. Der Nutzer kann innerhalb der Webanwendung die Gebäude auswählen und bekommt die verschiedenen *OSM*-Eigenschaften und -Attribute angezeigt. Dies ist insbesondere für die Fehleranalyse von *OSM*-Daten hilfreich.

2.1.3 F4 Map

F4 Map ist ebenfalls eine kostenfreie Webanwendung, welche dem Nutzer eine 3D-Visualisierung der Gebäude in Form einer Webkarte bereitstellt. Diese wird als Demo Version [9] angeboten, um die Anwendung vorzuführen. Ein besonderes Merkmal hierbei

sind die Animationen, welche innerhalb des 3D Modells integriert wurden, wie etwa fahrende Schiffe, sprudelnde Springbrunnen, sich bewegende Baustellenkrane und vieles mehr. Entwickelt wurde dies mit Hilfe der WebGL Technologie, welche zum Rendern (Erstellung einer Grafik aus Rohdaten) von 2D/3D-Grafik innerhalb eines Webbrowsers verwendet wird. Spezifische Gebäude und Modelle wurden aus anderen (nicht *OSM*) Quellen importiert. Entwickler der Anwendung ist die *F4 Group*, welche kommerzielle Kartendienste anbietet und diese nutzerindividuell anpassen lässt [8].

Abb. 1: 3D Visualisierung der Stadt Frankfurt in F4 Map, Bildschirmaufnahme der Webanwendung [9]

2.1.4 Mapbox

Mapbox ist ein US-amerikanisches Unternehmen, welches kommerzielle Karten-Dienste anbietet. Unter den zahlreichen Kunden von *Mapbox* zählen unter anderem große Unternehmen wie *Facebook*, *Adobe*, *Lonely Planet*, *The Washington Post*, *Snap Inc.*, *The New York Times* und viele andere [23]. Den namenhaften Kundenstamm verdankt die Firma *Mapbox* ihren benutzerdefinierten Karten-Diensten, welche als Entwicklerwerkzeuge bereitgestellt werden. Zu den Datenquellen zählt insbesondere *OpenStreetMap*, aber auch andere Quellen, wodurch eine möglichst akkurate Kartierung ermöglicht werden soll. Neben den diversen Diensten, die angeboten werden, ist im Rahmen dieser Arbeit vor allem das „*Mapbox SDK für Unity*“ [27] von Interesse. Dieses wurde für die Erstellung der 3D-Gebäude in dieser Arbeit verwendet und wird in Kapitel 3.4 näher dokumentiert und behandelt.

2.2 Sonstige Werke

Derzeit existieren keine auf *OpenStreetMap* basierten 3D-Stadtmodelle, welche den ÖPNV mitintegrieren oder in irgendeiner Form unterstützen. Im Folgenden sind weitere Werke aufgeführt, welche (teilweise) sowohl 3D-Stadtmodelle als auch den ÖPNV visualisieren

bzw. unterstützen und auf einer anderen Datenquelle als *OpenStreetMap* beruhen. Diese sollen als Leitfaden für die Realisierung dieses Projekts dienen.

2.2.1 HoloLens 3D Visualisierung ÖPNV

Eine grafisch ansprechende Darstellung des ÖPNV bietet die *HoloLens 3D Visualisierung des ÖPNV* der Firma *Duality*. Dazu wurde für ein Modellprojekt der komplette Datenbestand eines großen ÖPNV-Anbieters analysiert und zu unternehmensdienlichen Informationen umgewandelt. Zu den internen Informationen wurden auch externe Informationen (Wetterinformationen, Qualität von Straßen usw.) hinzugefügt. Diese Daten wurden dann mit Hilfe der Spiele-Engine *Unity* und *Microsofts HoloLens Brille* in AR visualisiert (siehe Abb. 2) [6]. „Ziel des Projekts ist es, neue Erkenntnisse basierend auf allen vorhandenen Daten zu erhalten, statistische Schlussfolgerungen zu identifizieren, die Möglichkeit der Vorhersage aufzuzeigen und dem Benutzer zu ermöglichen, Entscheidungen basierend auf Echtzeitinformationen zu treffen“ [6]. Als Kritik kann aufgeführt werden, dass das Projekt lediglich für einen ÖPNV-Anbieter prototypisch verwirklicht wurde und für eine flächendeckende Implementierung ein zu großer Bedarf an nicht allgemein zugänglichen Informationen besteht.

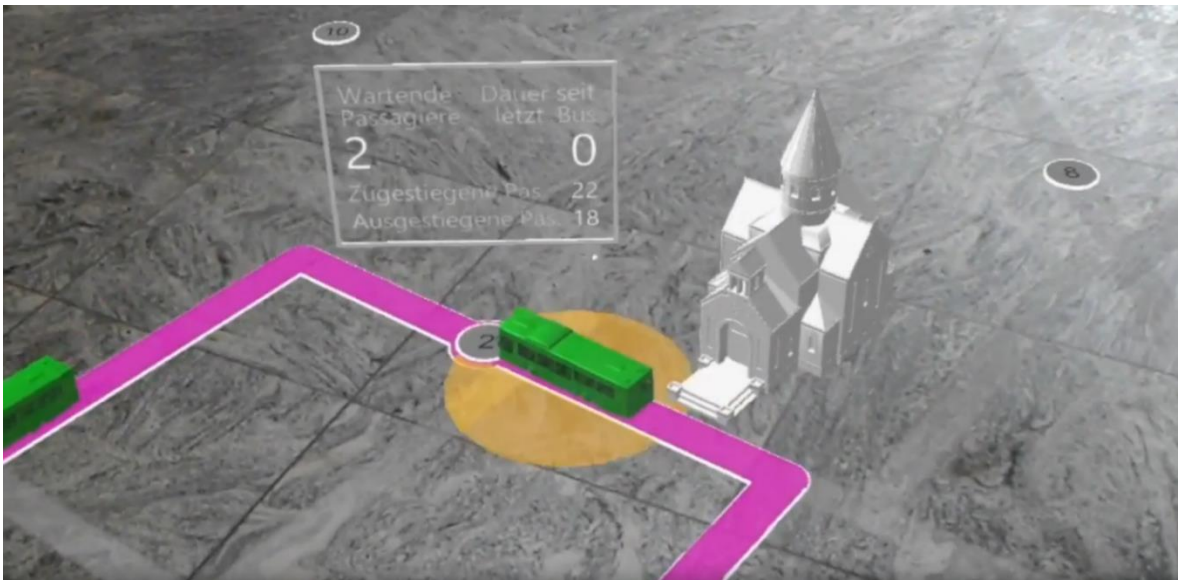


Abb. 2: 3D Visualisierung des ÖPNV in AR, Bildschirmaufnahme des YouTube-Videos [7]

2.2.2 Google Maps & Google Earth

Das US-amerikanische Unternehmen *Google* bietet die wahrscheinlich grafisch beste und akkurateste Visualisierung eines 3D-Stadtmodells, welche aktuell auf dem Markt verfügbar ist. Vorreiter hierbei ist der kostenfreie Online-Kartendienst *Google Maps* [13] und das kostenfreie Programm *Google Earth* [12]. Für die 3D-Darstellung werden unter anderem

2D-Satellitenbilder und -Luftaufnahmen aus Flugzeugen (ähnlich zu den eingesetzten Autos von *Google Street View*) aufgenommen und aufeinander abgestimmt. Mit Hilfe der Flugzeugaufnahmen aus verschiedenen Winkeln werden Tiefeninformationen ausgelesen und auf die Satellitenaufnahmen angewandt. Anschließend werden die 3D-Objekte mit Hilfe dieser Aufnahmen texturiert. Laut *Googles* Entwicklern werden die Satellitenbilder alle paar Jahre erneut aufgenommen. Flugzeugaufnahmen größerer Städte werden sogar jährlich aktualisiert. Dieser Vorgang der Aktualisierung ermöglicht es auch, neue Erkenntnisse über die Veränderungen des Planeten zu erforschen (z.B. Entwaldung, Schmelzen der Polkappen usw.) [4].

Abb. 3: 3D Visualisierung der Stadt Frankfurt in Google Earth, Bildschirmaufnahme des Programms Google Earth [12]

Eine in Hinblick auf die Thematik dieser Arbeit interessante Funktionalität von *Google Maps* ist *Google Transit*. Dabei handelt es sich um ein Projekt, welches von den beiden *Google*-Mitarbeitern Avichal Garg und Chris Harrelson im Jahr 2005 ins Leben gerufen wurde. Die beiden Entwickler nutzten diejenigen 20 Prozent ihrer Arbeitszeit, die man als *Google*-Mitarbeiter für eigene Projekte verwenden darf, um eine Erweiterung von *Google Maps* zu entwickeln. Der Dienst, den sie entwickeln, visualisiert den lokalen ÖPNV einer Stadt und zeigt deren jeweilige Stationen an. Seit einiger Zeit schon befindet sich das Unternehmen in Verhandlungen mit zahlreichen Städten und Verkehrsverbünden, um deren Daten nutzen zu können und innerhalb des Kartendienstes zu integrieren. Im Jahre 2007 wurde der Dienst in *Google Map*s aufgenommen und dadurch ein offizieller Bestandteil des Angebots von *Google* [52]. Die *Deutsche Bahn* hat sich ebenfalls am Projekt beteiligt und im September 2012 eine Kooperation mit *Google* verkündet [5]. Mittlerweile unterstützen viele Großstädte in Deutschland *Google Transit* und bei manchen werden sogar Echtzeitdaten für Busse und Bahnen angezeigt.

Tabelle 1: Unterstützung des Google Transit Dienstes in deutschen Großstädten im Bezug zu den öffentlichen Verkehrsmitteln, Daten am 8.3.2020 durch Selbstüberprüfung in Google Maps hinzugefügt

Stadt	S-Bahn	U-Bahn	Tram	Bus	Echtzeitdaten
Berlin	ja	ja	ja	ja	ja
Hamburg	ja	ja	/	ja	ja
München	ja	ja	ja	ja	ja
Köln	ja	/	ja	ja	ja
Frankfurt am Main	ja	nein	nein	nein	ja
Stuttgart	ja	ja	/	ja	ja
Düsseldorf	ja	ja	ja	ja	ja

Durch eine Überprüfung des Dienstes anhand von sieben deutschen Großstädten (siehe Tabelle 1) wurde festgestellt, dass Frankfurt aktuell die einzige Stadt ist, für die keine Unterstützung im *Google Transit*-Dienst besteht. Dies ist darauf zurückzuführen, dass der lokale Verkehrsverbund *RMV* seine Rohdaten nicht für *Google Maps* bereitstellt. Diese Erkenntnis weist eine große Schwachstelle des Dienstes und des gesamten Projekts auf: Es besteht eine große Abhängigkeit von den lokalen Verkehrsverbünden und nicht allgemein zugänglichen Daten.

2.2.3 Apple Maps

Apple Maps ist ein von *Apple* entwickelter Karten-Dienst, welcher im September 2012 zusammen mit dem hauseigenen Betriebssystem *iOS6* veröffentlicht wurde und somit *Google Maps* als Standard Karten-Anwendung für *Apple*-Systeme ersetzte. Der Dienst ist nur auf *Apple*-Plattformen oder als Webanwendung über die Suchmaschine *DuckDuckGo* verfügbar. Die Geoinformationen bezieht *Apple* von der Firma *TomTom* aber auch von anderen Quellen, unter denen *Automotive Navigation Data (AND), Hexagon AB*, *Intermap Technologies*, *OpenStreetMap* und *Waze* zählen [51]. Eine 3D-Visualisierung der Gebäude wird über *Apple*‘s *flyover Mode* für einige urbane Gebiete angeboten. Dabei handelt es sich um einen Dienst, welcher ursprünglich von der schwedischen Firma *C3 Technologies* entwickelt wurde. Es werden Flugzeugaufnahmen von Gebäuden mit verschiedenen Kameras aus verschiedenen Winkeln aufgenommen (ähnlich wie bei *Google*s 3D-Anwendung). Danach werden die Aufnahmen mit Hilfe eines Algorithmus analysiert und in 3D-Modelle umgewandelt [43]. Aktuell wird *flyover* in folgenden deutschen Städten unterstützt: Augsburg, Berlin, Bielefeld, Braunschweig, Bremen, Köln, Dresden, Hamburg, Hannover, Karlsruhe, Kiel, Leipzig, Mannheim, München, Münster, Nürnberg und Stuttgart [1].

Ein weiterer Dienst von *Apple Maps* ist die Funktion *Transit*. Dabei werden ÖPNV-Routen visualisiert und mit Echtzeitdaten den Benutzern bereitgestellt. Zusätzlich wird die derzeitige Position des Busses bzw. Zuges auf der Route visualisiert [43]. Nach offiziellen Angaben wird aktuell in Deutschland nur die Stadt Berlin unterstützt [1].

Kapitel 3
Grundlagen und Entwicklungsressourcen

3.1 Einführung in 3D-Stadtmodelle

3D-Stadtmodelle dienen zur Visualisierung einer Beschreibung des Geländes, der Straßen, der Gebäude und der Vegetation in einem dreidimensionalen Maßstab. Es gibt zahlreiche Anwendungsbereiche für derartige 3D-Stadtmodelle, insbesondere für 3D Autonavigationssysteme, virtuelle Tourismusinformationssysteme, Visualisierungen für die Stadt und Gebäudeplanung sowie für diverse Architekturwettbewerbe. Dabei haben die jeweiligen Anwendungen verschiedene Anforderungen in Bezug auf den Detaillierungsgrad oder die Visualisierungsmethode [3]. Eine mögliche Visualisierungsart ist der „virtuelle Spaziergang“, bei dem sich der Nutzer frei innerhalb des Modells bewegen kann. Diese Visualisierungsart wird insbesondere bei tourismusbezogenen Modellen genutzt. Die Rekonstruktion von historisch verschwundenen Szenarien ist ebenfalls eine Visualisierungsart, welche immer mehr Anwendung findet, und mittlerweile als ein weitverbreitetes Konzept zur Beschreibung von historischen Bauwerken dient [21]. Ein Beispiel einer solchen Anwendung ist die "*Stolperwege App*" [30]. Diese hat als Ziel die Vermittlung der Geschichte des Holocaust im Kontext der allgemeinen Umgebung und bietet in diesem Zusammenhang 3D-Animationen von zerstörten oder nicht begehbaren Objekten, welche auf einer modernen Karte synchronisiert werden. Dabei können mit Hilfe der Softwarekomponente "*VAnnotatoR*" [31, 48] diverse Texte und Bilder zu den virtuell begehbaren Gebäuden in VR annotiert werden und somit ein einzigartiger Einblick in die menschliche Geschichte ermöglicht werden. Großflächige Bereiche zu visualisieren ist eine weitere Anwendungsart, bei der versucht wird eine möglichst realitätsgetreue Situation nachzubilden [21]. „So kann man sich in ihr orientieren, Sichtlinien prüfen oder aber diese Darstellung als Basis für Simulationen (Integration beweglicher Objekte) verwenden. Diese Anwendung der Visualisierung wird als dreidimensionale Karte bezeichnet“ [21]. Im Rahmen dieser Arbeit wird versucht, eine dreidimensionale Karte für den ÖPNV nachzubilden.

Aufgrund der perspektivischen Vorteile eines 3D-Stadtmodells und dank des technologischen Fortschritts, welcher heutzutage eine effiziente Umsetzung ermöglicht, gibt es einen immer größer werdenden Bedarf an derartigen Modellen sowohl für die Forschung als auch in der Wirtschaft. Die *Geodateninitative Nordrhein-Westfalen* (*GDI NRW*), welche eine Initiative der *Special Interest Group 3D* (*SIG 3D*) ist, hat 2002 ein Austauschformat entwickelt, welches die Übertragung raumbezogener Objekte, Relationen und Geometrien ermöglicht. *CityGML*, welches auf den ISO-Standards ISO191xx basiert,

wurde mit Hilfe der auf XML basierten Auszeichnungssprache *Geography Markup Language* (*GML3*) realisiert [19]. „Als Multiskalen-Repräsentation konzipiert erlaubt CityGML die verlustfreie Speicherung und Übertragung der 3D-Geometrie, Topologie und Thematik von 3D-Stadtmodellen in fünf Detaillierungsgraden (LOD 0 bis LOD 4). Es wird zwischen Gelände, Gebäude, Vegetation, Gewässer und Verkehr unterschieden“ [19]. In Abb. 4 und Tabelle 2 werden die jeweiligen Detailstufen veranschaulicht und näher erklärt.

Tabelle 2: Detailstufen von CityGML mit Angabe von Genauigkeiten und Mindestgrößen, Quelle: [19]

	Beschreibung	Genauigkeit Lage	Genauigkeit Höhe	Mindestgröße Grundfläche
LOD 0	2.5D digitale Geländemodelle (DGM) mit überlagertem Luftbild oder Karte	-	-	-
LOD 1	Blockmodelle ohne Dachstrukturen oder Texturen	≥ 5m	≥ 5m	6m x 6m
LOD 2	Modelle mit Dachstrukturen und Texturen + Vegetationsobjekte	2m	1m	4m x 4m
LOD 3	Architekturmodelle mit detaillierten Fassaden- und Dachstrukturen (z.B. Balkone, Erker, Vorsprünge) mit hoch aufgelösten Texturen	0.5m	0.5m	2m x 2m
LOD 4	Begehbare Innenraummodelle (Räume, Türen, Treppen, Möbel usw.)	≤ 0.2m	≤ 0.2m	-

Abb. 4: Visualisierung der 5 Detaillierungsgrade LOD0 bis LOD4, Quelle: [2]

Im Rahmen dieser Arbeit wird für das zu entwickelnde 3D-Stadtmodell ein Detaillierungsgrad von LOD 1 angestrebt, da dieser eine effiziente Ausführung ermöglicht und die 3D-Modelle lediglich zur besseren Orientierung und Visualisierung der geografischen Gegebenheiten dienen sollen.

3.2 OpenStreetMap als Datenquelle

Bei der Auswahl der Datenquelle waren folgende Anforderungen zu beachten:

- Freie Nutzung und Verarbeitung der Daten (lizenztechnisch)
- Kostenfreiheit
- Gute Datenlage (Straßen, Gebäude und ÖPNV)
- Gutes Datenformat für die Datenverarbeitung
- Daten werden aktualisiert (passen sich den Veränderungen der Welt an)
- Gute Zukunftsaussichten (Datenquelle wird durch den Entwickler auch in den kommenden Jahren unterstützt und gepflegt)

Als Datenquelle wurde *OpenStreetMap* ausgewählt, da diese allen Anforderungen entspricht und für ähnliche Projekte (siehe Kapitel 2.1) bereits öfters verwendet wurde. Im nächsten Unterkapitel wird das *OSM*-Projekt vorgestellt und jeweils auf die einzelnen Anforderungspunkte eingegangen.

3.2.1 Das OpenStreetMap-Projekt

„OpenStreetMap (OSM) ist heute das weltweit größte Open-Data-Projekt im Bereich digitaler Geoinformationen und eines der größten Open-Data-Projekte überhaupt“ [46]. Dabei handelt es sich um ein im Jahre 2004 gegründetes Projekt, welches als Ziel die Erstellung einer freien Weltkarte hat. Im Zuge dessen werden mit Hilfe von Crowdsourcing Daten zu Straßen, Eisenbahnen, Flüsse, Wälder, Häuser und vieles mehr gesammelt und auf einer Karte visualisiert. Die Daten, welche durch *OpenStreetMap* an die Benutzer bereitgestellt werden, sind lizenzkostenfrei und dürfen von jedem eingesetzt und weiterverarbeitet werden. Dies ist darauf zurückzuführen, dass die Daten von *OSM* selbst bzw. deren Community erhoben werden und keiner lizenzgebundenen Quelle entstammen. Somit sind die lizenztechnischen Aspekte und die Kostenfreiheit durch *OpenStreetMap* gewährleistet. Im Gegensatz zu *Google Maps* werden bei *OSM* kaum Bedingungen an die Benutzung der Karten geknüpft. Dementsprechend sind *Google*s Karten kostenlos nutzbar, aber nicht frei. Der größte Nachteil allerdings von *Google Maps* im Vergleich zu *OSM* ist, dass *Google* nicht die entsprechenden Geoinformationen an die Anwender bereitstellt [34].

Die Daten selbst erhält *OpenStreetMap* durch die zahlreichen Nutzer, die sich aktiv am Projekt beteiligen. Dabei ist der größte Anteil dieser Nutzer mit dem Mapping beschäftigt, also dem Sammeln von Kartendaten mit Hilfe von GPS-Geräten und der Eingabe in *OpenStreetMap*. Öfters werden Luftbilder abgezeichnet oder auch alte Stadtpläne, deren Copyright abgelaufen ist [34]. Dabei stellt sich die Frage, wie hoch die Qualität solcher Crowdsourcing-Daten ist:

Bezüglich der Datenqualität von OpenStreetMap gibt es mittlerweile zahlreiche Studien. Muki Haklay hat 2010 einen Vergleich des Straßenbestandes in *OpenStreetMap* mit den amtlichen Daten des nationalen englischen Landesvermessungsamts *Ordnance Survey*

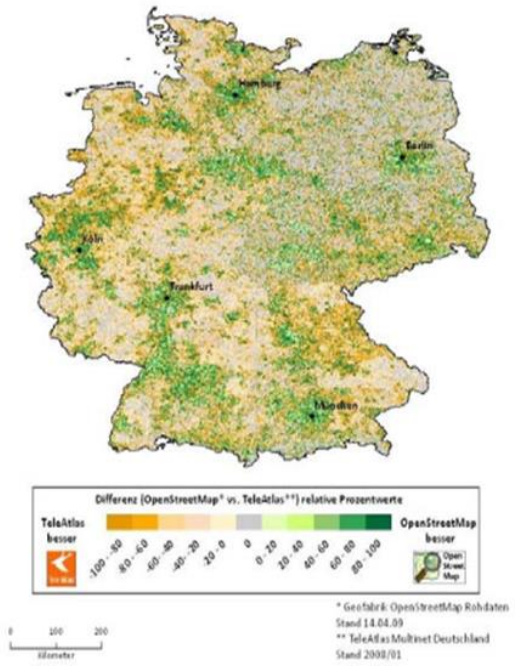


Abb. 5: Differenz des Gesamtwegenetzes zwischen OSM und TeleAtlas pro km² in %, Quelle: [54]

durchgeführt [16], bei dem festgestellt wurde, dass die Autobahnen in *OSM* bis zu 80% mit den amtlichen Daten übereinstimmten. Insgesamt wurde durch den Vergleich abgeleitet, dass urbane und dicht besiedelte Gebiete eine sehr gute Datenlage aufweisen, während der Stadtrand und Gebiete mit einer geringeren Bevölkerungsdichte ein starkes Gefälle hinsichtlich der Vollständigkeit verzeichnen. Analog zu diesem Vergleich wurde im gleichen Jahr eine entsprechende Untersuchung für das Gebiet von Deutschland durch Zielstra & Zipf [54] durchgeführt, die ähnliche Ergebnisse lieferte. Beim Vergleich von *OSM* und dem kommerziellen Anbieter *TeleAtlas* wurde festgestellt, dass in Groß- und Mittelstädten die Gesamtstraßenlänge, welche in *OSM* kartiert wurde, die Gesamtstraßenlänge von *TeleAtlas* übertrifft. Bewegt man sich allerdings in weniger dicht besiedelten Regionen, ist ebenfalls eine geringere Kartierung seitens *OSM* zu erkennen (siehe Abb. 5) [47].

Marcus Götz hat 2012 eine quantitative Bewertung des Gebäudebestands in *OSM* vorgenommen. Im Intervall von 2007 bis 2012 wurde ein stetiger Zuwachs an Gebäudedaten festgestellt (siehe Abb. 6). Dabei übertraf die Anzahl der Gebäude erstmals im Jahre 2012 die Anzahl der Straßen. „Jede Woche werden etwa 450 000 neue Gebäudegrundrisse erfasst und der Abstand zwischen Gebäuden und Straßen wird zunehmend größer“ [14].

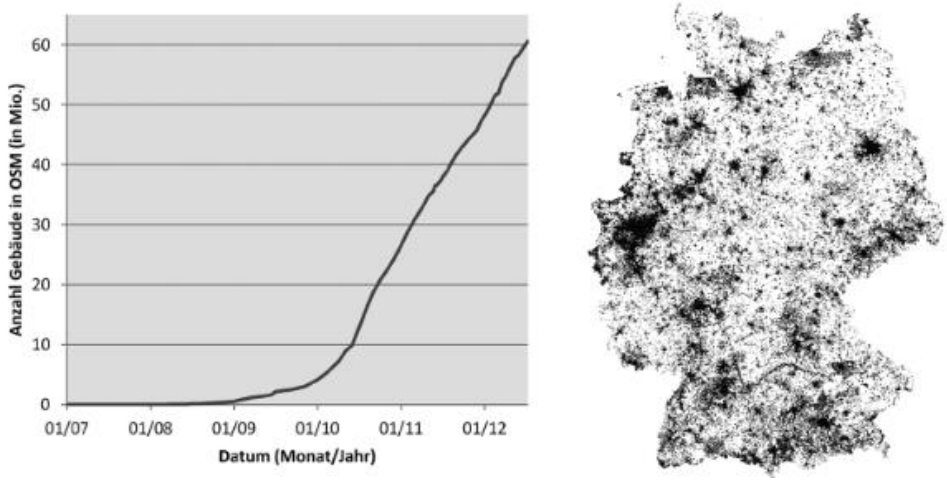


Abb. 6: (links) Entwicklung des globalen Gebäudebestands in OSM seit 2007, (rechts) Visualisierung der OSM-Gebäudepolygone in Deutschland, Quelle: [14]

Der Nahverkehr in Deutschland weist in *OpenStreetMap* eine hohe Datenqualität auf. Dies liegt teilweise auch daran, dass sich seit 2011 der Verkehrsverbund *Rhein-Sieg* (*VRS*) und der *Aachener Verkehrsverbund* (*AVV*) aktiv bei *OSM* engagieren [11]. Des Weiteren hat der Entwickler Melchior Moos eine Onlinekarte [32] bereitgestellt, welche zu 100% *OpenStreetMap*-Daten nutzt, um den ÖPNV auf einer 2D-Karte zu visualisieren (siehe Abb. 7). Diese hatte ebenfalls einen positiven Einfluss auf die Entwicklung des Datenbestandes in *OSM* (in Bezug auf den Nahverkehr), da es nun möglich war die Nahverkehrsdaten aus *OSM* direkt zu prüfen und mit anderen Quellen zu vergleichen. Der deutschlandweite Nahverkehr wird in *OpenStreetMap* auf einer eigenen Wiki-Seite dokumentiert und behandelt, bei der zwischen den verschiedenen Verkehrsverbünden und Städten differenziert wird. Laut dieser Dokumentation [39] ist das U-Bahn- und Tram-Netz der Stadt Frankfurt vollständig in *OSM* enthalten (letzte Überprüfungen gab es 2015 und 2018). Beim Bus-Netz kann mit geschätzten 95% Überdeckung ebenfalls von einer sehr guten Datenlage, gesprochen werden. Insgesamt lässt sich für das Gebiet der Stadt Frankfurt sagen, dass die Datenlage und Datenqualität sehr hoch und nahezu vollständig ist.

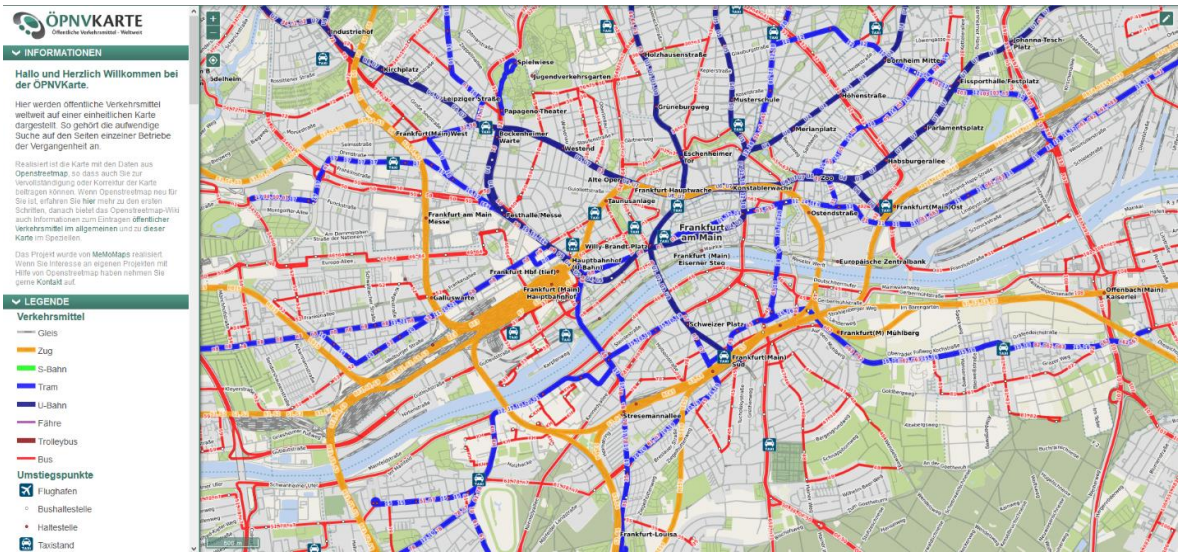


Abb. 7: Visualisierung des ÖPNV der Stadt Frankfurt anhand der OSM Daten, Bildschirmaufnahme der Webanwendung [32]

Da sehr viele Nutzer die Geoinformationen aus *OSM* für eigene Projekte verwenden, wird durch *OSM* ein XML-basiertes Austauschformat angeboten (OSM-XML), welches zur Übertragung der Daten von OSM zur eigentlichen Anwendung hin dient. „In diesem Format sind die Basiselemente Node, Way, Relation und Tag abgebildet, es ist vollständig dokumentiert und einfach weiterzuverarbeiten“ [46]. Mit Hilfe von Konvertern kann eine OSM-XML Datei in ein anderes Datenformat umgewandelt werden, falls dies gewünscht ist [46].

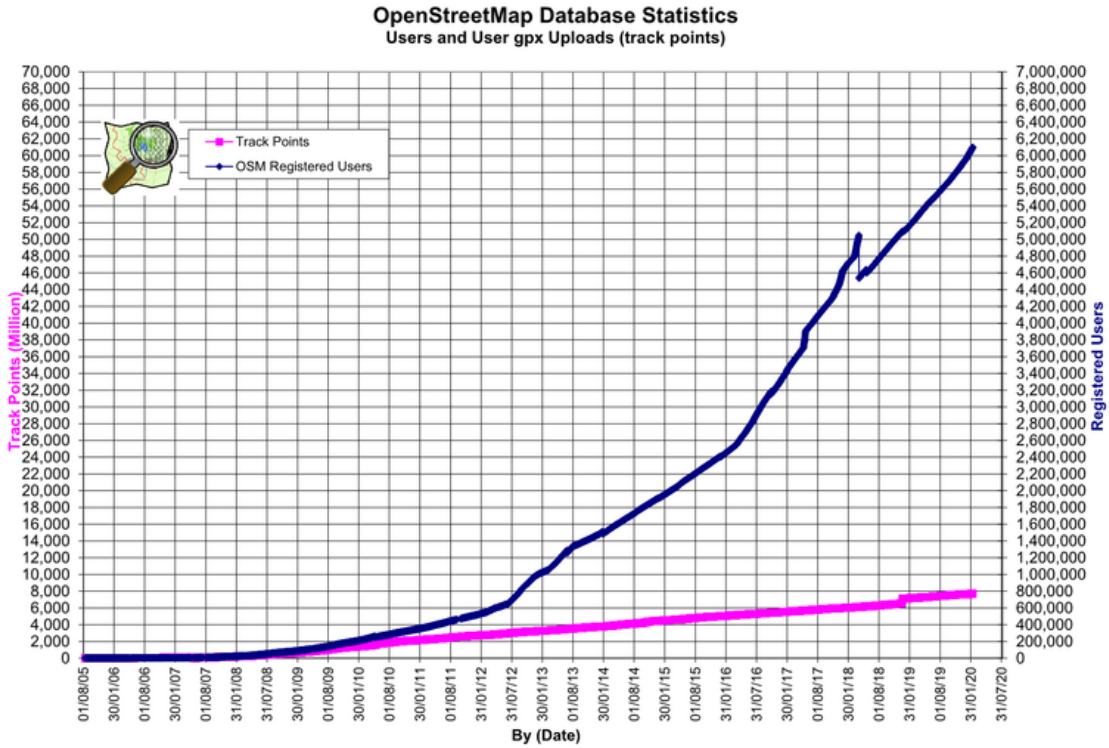


Abb. 8: Anzahl der registrierten Nutzer und die Anzahl der Track Points in OSM seit 2005, Quelle: [40]

OSM hat über die Jahre einen stetigen Zuwachs bezüglich der Nutzerzahlen verzeichnet (siehe Abb. 8). Aktuell hat *OpenStreetMap* über 6 Millionen registrierte Nutzer (Stand: 07.02.2020) [40]. Aus den Statistiken [40], die *OSM* auf ihrer Wiki-Seite veröffentlicht, wurde die Abb. 9 entnommen, mit der schnell ersichtlich wird, dass die Bearbeitung von bereits erfassten Datensätzen und dessen Aktualisierung ebenfalls einen stetigen Zuwachs aufweist.

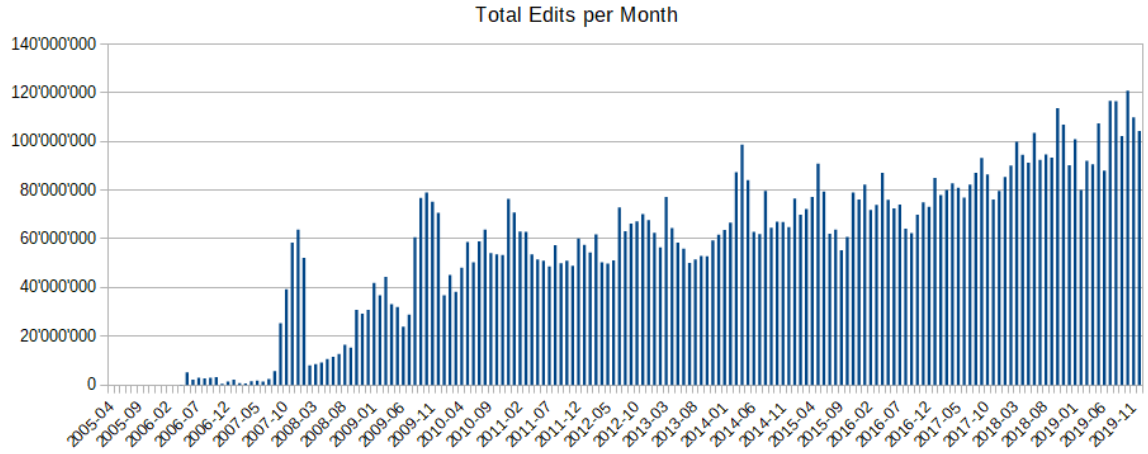


Abb. 9: Anzahl der monatlichen Anpassung von bereits eingefügten Daten in OSM seit 2005, Quelle: [40]

„OpenStreetMap wird zu Recht oft mit Wikipedia verglichen. Beide Plattformen lassen Laien gleichberechtigt mit Experten Wissen zusammentragen, wachsen rasant und sind in ihren Feldern mittlerweile allgemein anerkannte Akteure. OpenStreetMap weist darüber hinaus viele Schwächen der Wikipedia und anderer User-Generated Content Projekte nicht auf. Da geographische Gegebenheiten zum ganz überwiegenden Teil Fakten sind, kann darüber auch nur schwer unter den Teilnehmern gestritten werden“ [46]. Unter Berücksichtigung der bereits vorgewiesenen Datenqualität und der rasant steigenden Nutzerzahlen, kann allgemein angenommen werden, dass *OpenStreetMap* auch in den kommenden Jahren eine sehr gute Quelle für Geoinformationen sein wird und durch die Entwickler bzw. die Community weitergepflegt wird.

3.2.2 Aufbau und Struktur einer OSM XML-Datei

Ein wichtiger Kernpunkt bei der Datenverarbeitung ist das Datenformat. Bei OSM XML handelt es sich um ein XML-basiertes Datenformat, welches für den Austausch von Geoinformationen genutzt wird. Die Daten werden innerhalb einer Textdatei nach bestimmten Kriterien hierarchisch strukturiert. Dies führt am Ende dazu, dass die Daten sowohl von Menschen als auch von Maschinen lesbar und interpretierbar sind [53]. Dabei ist es wichtig, die Kriterien und die Hierarchie zu kennen und zu verstehen. In diesem Unterkapitel wird nur auf die Daten eingegangen, welche in dem realisierten Projekt genutzt wurden. Für eine weitere Vertiefung diesbezüglich wird auf die offizielle Dokumentation [38] von *OSM* verwiesen.

Grundsätzlich besteht eine osm-Datei aus den vier Grund-Datensatztypen **Bounds**, **Nodes**, **Ways** und **Relations**. Mit Hilfe des **Tag**-Datensatzes können Nodes, Ways und Relations um weitere Eigenschaften angereichert werden (siehe Abb. 10).

Ganz am Anfang einer osm-Datei befindet sich der **Bounds**-Datensatz. Dieser spezifiziert die vier Grenzkoordinaten des Gebietes, welches in der jeweiligen osm-Datei beschrieben wird. Dabei beschreiben die Variablen “*minlat*” und “*minlon*” die minimale geografische Breite bzw. Länge und die Variablen “*maxlat*” und “*maxlon*” die maximale geografische Breite bzw. Länge, welche in dem ausgewählten Gebiet vorkommen.

Der **Node**-Datensatz ist unter anderem der wichtigste Datensatz innerhalb der osm-Datei, da er die geografischen Objekte darstellt und die Positionskoordinaten enthält. Dieser wird durch die Variablen “*lat*” und “*long*” beschrieben, welche für die geografische Breite und Länge stehen (geografische Position des Objekts). Dabei symbolisiert ein Node einen geometrischen Punkt z.B. eine Ampel und kann durch verschiedene Tags mit weiteren Information versehen werden. Jedem Node wird mit Hilfe der Variable “*id*” eine eindeutige Identifikationsnummer zugewiesen, um das jeweilige Objekt zu benennen.

Ein **Way**-Datensatz wird ebenfalls durch eine eindeutige Identifikationsnummer “*id*” benannt. Jeder Way besteht aus mehreren Node-Datensätzen, welche den jeweiligen Way darstellen. Dabei werden die zugehörigen Nodes mit der Bezeichnung „*nd*“ und der

entsprechenden Identifikationsnummer unter der Variable "*ref*" hinterlegt. Ein Way setzt sich so zusammen, dass die Nodes der Reihe nach verbunden werden und eine Linie entsteht. Auf diese Weise können Straßen, Flüsse, Eisenbahnlinien u.v.m. modelliert werden. Eine Besonderheit hierbei ist, dass ein Node mehreren Ways zugeordnet sein kann und diese sich z.B. an Kreuzungen berühren können. Ways können ebenfalls durch Tags weitere Information enthalten. Im Spezialfall, dass der Anfangs- und Endpunkt eines Way Datensatzes identisch ist, wird eine geschlossene Linie gebildet, welche z.B. ein Gebäude darstellen kann. Solche geschlossenen Linien bilden Flächen und werden meistens durch Attribute näher beschrieben, z.B. *tag k="building".*

Der **Relation**-Datensatz umfasst Punkte, Linien oder auch andere Relationen und stellt eine Beziehung zwischen diesen Objekten dar. Dabei werden die Objekte als Mitglieder der Relation benannt und durch den Datensatz "*member*" gespeichert. Die Variable "type" spezifiziert den Grund-Datensatztyp und die Variable "ref" beschreibt die jeweilige Identifikationsnummer des Mitglieds der Relation. Unter der Variable "*role*" können nähere Informationen hinterlegt sein, welche die Rolle des jeweiligen Objekts innerhalb der Relation beschreiben. Die Informationen über die Relationseigenschaften werden durch die Tag-Datensätze hinterlegt. Eine Relation enthält auch eine eindeutige Identifikationsnummer "*id*", da diese ebenfalls als ein Mitglied in anderen Relationen vorkommen kann.

Der **Tag**-Datensatz ist kein alleinstehender Datensatz und wird immer in Bezug zu einem Node, Way oder einer Relation genutzt. Er dient hauptsächlich für die Speicherung von weiteren Informationen und wird durch einen Schlüssel "*k*" und einen Wert "*w*" beschrieben. Die Variable "*k*" beschreibt dabei eine Informationsart und die Variable "*w*" den dazugehörigen Informationswert.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
 <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900" maxlon="12.2524800"/>
 <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO" uid="46882" visible="true" version="1" changeset="676636" timestamp="2008-09-21T21:37:45Z"/>
 <node id="261728686" lat="54.0906309" lon="12.2441924" user="PikoWinter" uid="36744" visible="true" version="1" changeset="323878" timestamp="2008-05-03T13:39:23Z"/>
 <node id="1831881213" version="1" changeset="12370172" lat="54.0900666" lon="12.2539381" user="lafkor" uid="75625" visible="true" timestamp="2012-07-20T09:43:19Z">
  <tag k="name" v="Neu Broderstorf"/>
  <tag k="traffic_sign" v="city_limit"/>
 </node>
 ...
 <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO" uid="46882" visible="true" version="1" changeset="676636" timestamp="2008-09-21T21:37:45Z"/>
 <way id="26659127" user="Masch" uid="55988" visible="true" version="5" changeset="4142606" timestamp="2010-03-16T11:47:08Z">
  <nd ref="292403538"/>
  <nd ref="298884289"/>
  ...
  <nd ref="261728686"/>
  <tag k="highway" v="unclassified"/>
  <tag k="name" v="Pastower Straße"/>
 </way>
 <relation id="56688" user="kmvar" uid="56190" visible="true" version="28" changeset="6947637" timestamp="2011-01-12T14:23:49Z">
  <member type="node" ref="294942404" role=""/>
  ...
  <member type="node" ref="364933006" role=""/>
  <member type="way" ref="4579143" role=""/>
  ...
  <member type="node" ref="249673494" role=""/>
  <tag k="name" v="Küstenbus Linie 123"/>
  <tag k="network" v="VVW"/>
  <tag k="operator" v="Regionalverkehr Küste"/>
  <tag k="ref" v="123"/>
  <tag k="route" v="bus"/>
  <tag k="type" v="route"/>
 </relation>
 ...
</osm>
```

Abb. 10: Beispiel einer osm-Datei, welche im OSM-XML-Datenformat vorliegt, Quelle: [38]

3.3 Unity als Entwicklungsumgebung

Die Visualisierung von virtuellen 3D-Stadtmodellen bzw. Stadtsimulationen ist ein aufwendiges Unterfangen und bedarf oft der Zusammenarbeit von mehreren Entwicklern. Zudem müssen viele Aspekte beachtet und diverse Funktionalitäten implementiert werden. Als Alternative werden heutzutage Entwicklungsumgebungen in Form von Spiele-Engines angeboten, welche sich durch zahlreiche Funktionalitäten auszeichnen und die Entwicklung von virtuellen 2D- oder 3D-Welten erleichtern sollen. Solche Spiele-Engines enthalten diverse Basismodule wie Grafik-Engines, vorgefertigte Steuerungen, Physik, Soundsysteme u.v.m., was in der Entwicklung derartiger Anwendungen von großem Nutzen sein kann [22].

Für die Entwicklung dieses Projekts wurde die Spiele-Engine *Unity* verwendet. Diese wird vom Unternehmen *Unity Technologies* betrieben und ist aktuell eine der bekanntesten kostenfreien Spiele-Engines auf dem Markt. Die Hauptbestandteile der *Unity*-Entwicklungsumgebung sind die zwei Modi: Editor-Mode und Play-Mode.

Im Editor-Mode wird die eigentliche Entwicklung der Anwendung vorgenommen. Es existieren eine oder mehrere Szenen, welche einen virtuellen 2D- oder 3D-Raum darstellen und diverse Objekte enthalten können. Mit Hilfe einer hierarchischen Struktur werden die sich in der Szene befindlichen Objekte aufgelistet. Dabei können diese mit anderen Objekten gruppiert und einander zugewiesen werden, um somit eine Parent-Child-Beziehung zu bilden. Die Objekte können sowohl dreidimensionale physikalische als auch nicht-physikalische Objekte, wie etwa Licht-Effekte oder Benutzeroberflächen sein. Jedem Objekt wird eine Position innerhalb des virtuellen Raumes durch ein *Unity*-internes Koordinatensystem zugewiesen. Des Weiteren können die Objekte verschiedene Komponenten enthalten bzw. mit C#-Skripten ausgestattet werden. Die C#-Skripte implementieren dabei die Logik und stellen den funktionalen Teil der Anwendung dar. Öfters werden vorgefertigte Objekte in Form von Schablonen (Prefabs) erstellt, um diese dann mit Hilfe der C#-Skripte zur Laufzeit zu instanziieren. Es ist ebenfalls möglich, mehrere Szenen innerhalb einer Anwendung zu verknüpfen und mit Hilfe von C#-Befehlen von einer Szene zur nächsten zu wechseln.

Im Play Mode wird die tatsächliche Laufzeit des Programms wiedergegeben. Dieser ermöglicht dem Entwickler eine schnelle Nutzeransicht und wird hauptsächlich für das Testen und die Buganalyse genutzt. Die Nutzersicht im Play Mode wird durch ein Kamera-Objekt festgelegt, welches sich standardmäßig innerhalb der Szene befindet. Hierbei ist ein weiterer Vorteil von *Unity*, dass Anpassungen innerhalb des Editor-Modes zur Laufzeit des Programmes durchgeführt werden können. Außer dem Kameraobjekt befinden sich standardmäßig auch eine Lichtquelle und eine Himmel-Visualisierung (die sogenannte „Skybox“) innerhalb der Szene.

Diverse weitere Funktionalitäten werden den Entwicklern durch die Menüpunkte innerhalb der *Unity*-Anwendung bereitgestellt. Außerdem ermöglicht *Unity* eine

Exportierung des fertigen Programmes auf eine Vielzahl von Systemen. Einige von diesen sind: iOS, Android, Windows, Mac, Linux, Playstation 4, VR, AR, Smart TVs usw.

Abb. 11 veranschaulicht die Unity-Entwicklungsumgebung und enthält obendrein eine Erläuterung der jeweiligen Anwendungskomponenten.

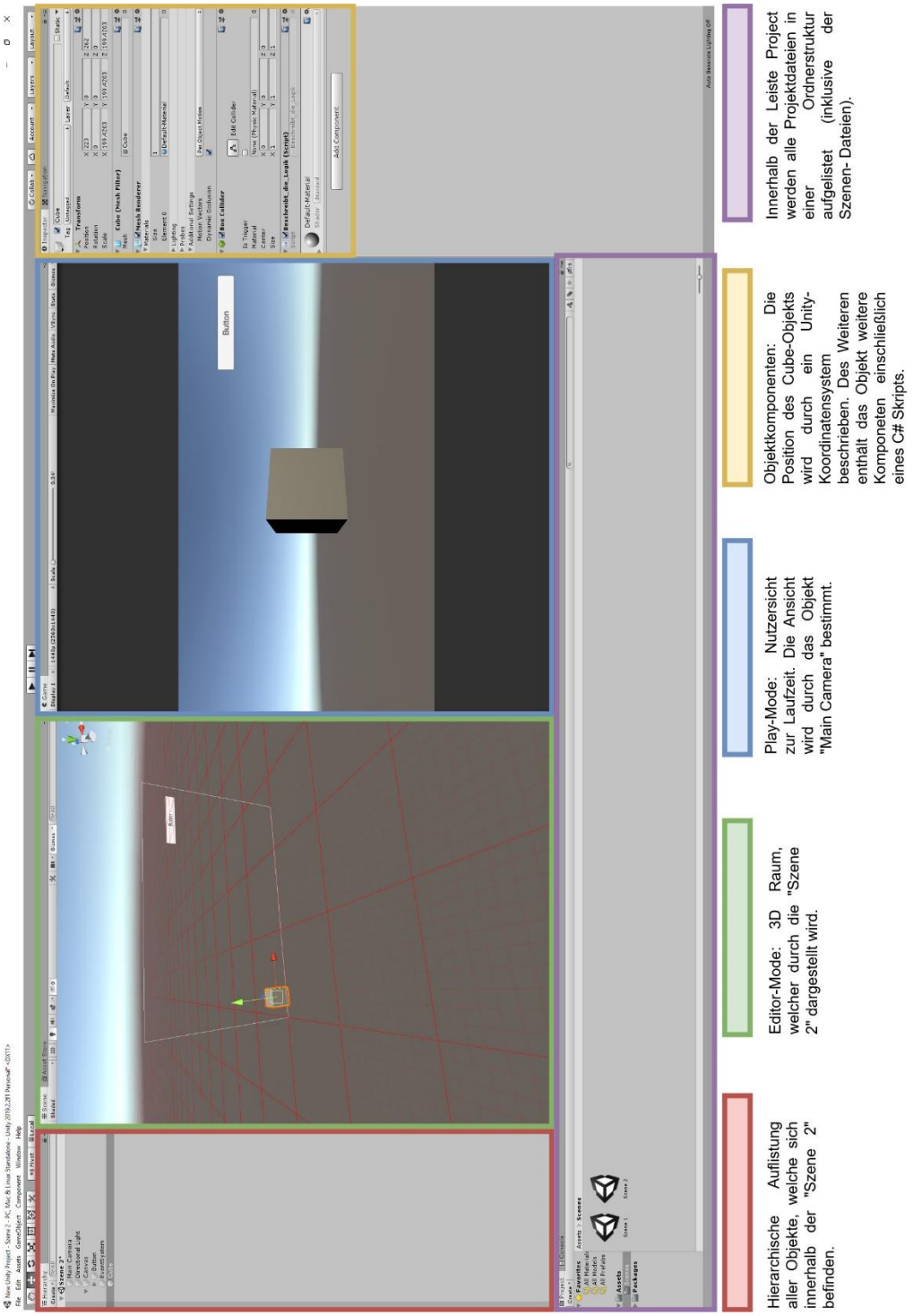


Abb. 11: Bildschirmaufnahme der Unity-Entwicklungsumgebung inklusive Erläuterung der Anwendungskomponenten

3.4 Mapbox SDK für Unity zur Gebäudegenerierung

Die Firma *Mapbox* ist ein US-Amerikanisches Unternehmen, welches (kommerzielle) Kartendienste und damit verbundene Werkzeuge an Entwickler bereitstellt. In diesem Zusammenhang werden diverse Quellen für die Datenerhebung hinzugezogen und entsprechend verarbeitet. Das finale Produkt wird den Entwicklern in Form von APIs und SDKs angeboten, welche auf verschiedenen Plattformen integrierbar und ausführbar sind. Dabei basieren die Anwendungen auf sogenannten Kachelsätzen, welche die Karte darstellen. Die Kachelsätze enthalten die jeweiligen Raster- oder Vektordaten und sind als einheitliches Gitter aus quadratischen Kacheln dargestellt. Diese werden insbesondere für die effiziente Visualisierung der Kartenobjekte benötigt und sind für einen schnellen Lade- und Zwischenspeicherprozess optimiert worden [25]. Es werden insgesamt 4 Grund-Kachelsatztypen angeboten:

- **Mapbox Streets** visualisiert Straßen, Gebäude, Verwaltungsbereiche, Wasseroberflächen und Landflächen.
- **Mapbox Terrain** beinhaltet Landbedeckung und globale Höheninformationen.
- **Mapbox Satellite** enthält globale Satellitenbilder.
- **Mapbox Traffic** umfasst regelmäßig aktualisierte Verkehrsdaten, welche auf den Mapbox-Straßen angezeigt werden.

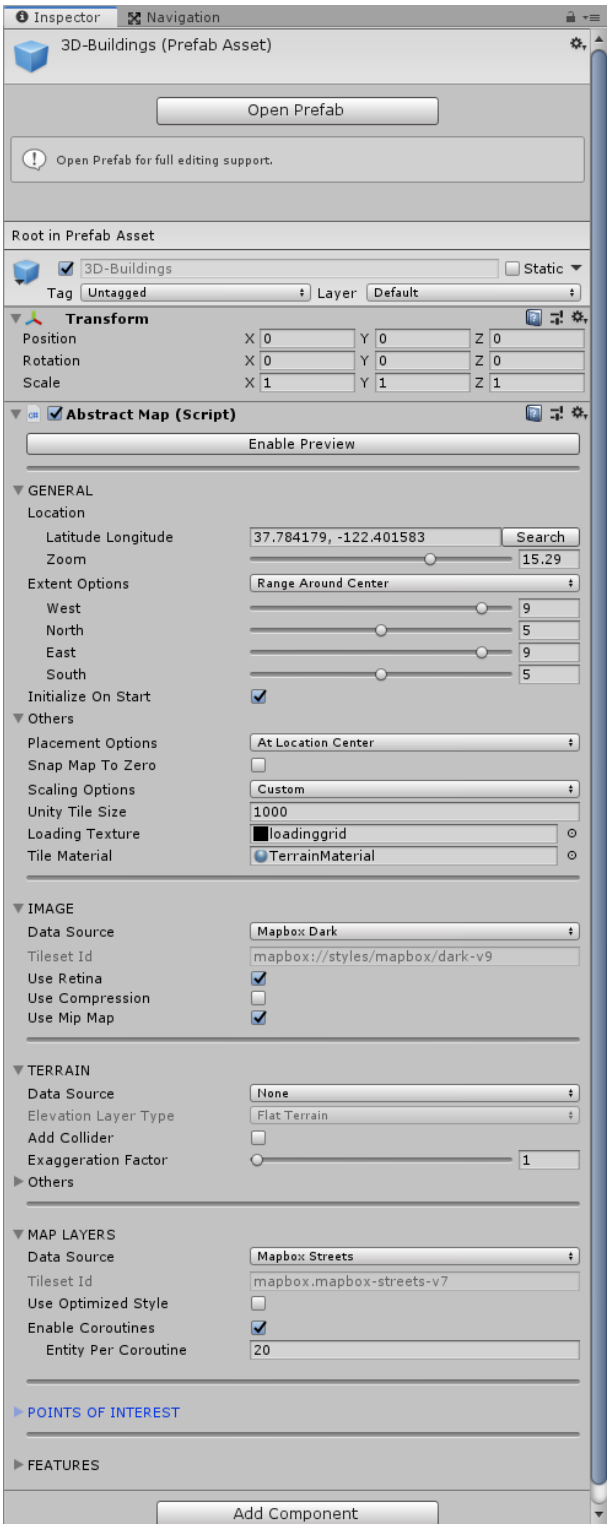
Die von Mapbox genutzten Datenquellen für die jeweiligen Kachelsatztypen sind unter der Literaturnummer [24] einsehbar.

Mapbox ermöglicht Webentwicklern eine einfache Integration der Karten innerhalb eigener Webseiten. Es wird die Einbindung auf mobile Endgeräte (iOS und Android) unterstützt, aber auch das Einbinden innerhalb *Unity*-Programme. Letzteres ermöglicht eine Simulation der realen Welt und bietet neue Entwicklungsmöglichkeiten. Neben den herkömmlichen Programmformaten werden auch AR-Anwendungen unterstützt.

Die *Mapbox SDK für Unity* ist ein Werkzeugsatz, welcher auf Basis von reellen Kartendaten die Erstellung und Integration von *Unity*-Programmen ermöglicht. Im Zuge dessen wird eine Kommunikation mit der Mapbox-Webanwendung aufgebaut, um die Daten aufzurufen und eine Konvertierung der Kartendaten zu *Unity Gameobjects* ausgeführt. Dies ermöglicht wiederum das Rendern von Kartendaten zur Laufzeit und die Visualisierung des aktuellsten Datenbestandes von Mapbox. Ein weiterer Vorteil diesbezüglich ist, dass nur eine Teilmenge der Daten bei Mapbox angefragt werden muss, wodurch eine effiziente Ausführung der Anwendung gewährleistet ist. Die *Mapbox SDK für Unity* kann kostenfrei heruntergeladen und mit Hilfe einer von Mapbox bereitgestellten Dokumentation in *Unity* eingebunden werden [28].

Für die Einbindung der SDK werden sogenannte „Access tokens“ benötigt. Diese werden bei der Registrierung auf der Mapbox Webseite generiert. Mit Hilfe der „Access tokens“ kann Mapbox die Anzahl der API-Anfragen mit dem jeweiligen Account verlinken und

dadurch eine monatliche Gebühr in Rechnung stellen. Für Anwendung, welche die Unity SDK einbinden, gilt: Falls die Anzahl der monatlichen Nutzer nicht über 25 000 steigt, ist die Nutzung der SDK komplett kostenfrei [26].



Nachdem die SDK erfolgreich in Unity eingebunden wurde, kann eines der verfügbaren Mapbox-Objekte in die Szene instanziiert werden. Für dieses Projekt wurde das Objekt „3D-Buildings“ verwendet. Die C#-Klasse „Abstract Map“ ist dabei mit dem Objekt verknüpft und dient als Entwickler-UI. Hier können die Koordinaten des gewünschten Ortes und die Anzahl der Kachelsätze festgelegt werden, um das jeweilige Modell zu generieren. Des Weiteren kann der Entwickler weitere Einstellungen und diverse Anpassungen an dem Modell vornehmen (siehe Abb. 12).

Abb. 12: Entwickler-UI des Mapbox-Objekts „3D-Buildings“, Bildschirmaufnahme in Unity

Kapitel 4
Implementierung

4.1 Einführung in die Struktur der Anwendung

Die Anwendung wurde mit Hilfe von zwei Szenen implementiert, welche verschiedene Objekte enthalten. Die erste Szene „*StartScreen*“ dient als Hauptmenü des Programms und wird bei der Ausführung der Anwendung als erstes angezeigt. Hier können diverse programmbezogene Informationen eingesehen bzw. Einstellung für das auszuführende Modell angepasst werden. Des Weiteren kann der Nutzer einen Dateipfad (die Datenquelle) angeben, um die Simulation zu starten. Es folgt ein Übergang zur nächsten Szene „*Simulator*“. Dort wird anhand der Daten ein virtuelles 3D-Stadtmodell generiert, welches weitere Benutzeroberflächen und (interaktive) Objekte enthält. Der Nutzer kann sich innerhalb dieses Stadtmodells frei bewegen und diverse Aktionen durchführen. Es ist ebenfalls zu jedem Zeitpunkt möglich die Simulation abzubrechen und zurück ins Hauptmenü (die Szene „*StartScreen*“) zu gelangen. Dieser Prozess wird in Abb. 13 visualisiert.

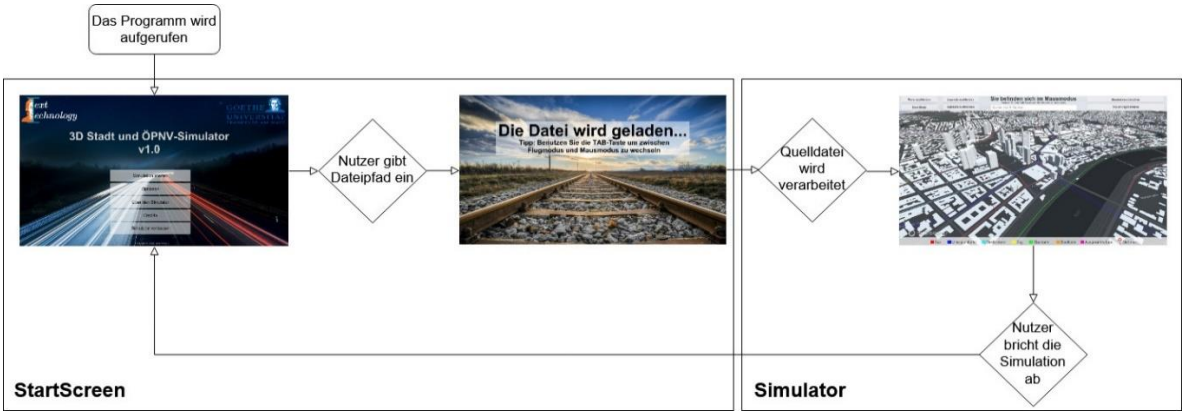


Abb. 13: Interaktionsdiagramm der beiden Szenen StartScreen und Simulator

Um die Simulation zu erzeugen, braucht die Anwendung eine Datenquelle. Die Daten müssen auf folgende Weise dem Programm übergeben werden: Der Nutzer muss innerhalb der *OpenStreetMap* Webanwendung ein Gebiet auswählen, um die Daten von der Webseite in Form einer Quelldatei zu exportieren. Als nächstes muss die Quelldatei in eine „*.txt*“ Datei umgewandelt werden (dies erfolgt meistens durch hinzufügen der Endung „*.txt*“ am Ende des Dateinamens). Als Letztes muss nur noch der Dateipfad kopiert und dem Programm übergeben werden. Das Programm greift daraufhin auf die Quelldatei (unter dem jeweiligen Dateipfad) zu und beginnt den Prozess der Datenverarbeitung und der

Erzeugung des 3D-Stadtmodells. Dabei ist es wichtig, dass die zu verarbeitende Quelldatei im OSM XML Format vorliegt. Eine entsprechende Anleitung diesbezüglich befindet sich innerhalb des Programms.

In den hier folgenden Unterkapiteln erfolgt eine Beschreibung der Implementierung der Szenen „*StartScreen*“ und „*Simulator*“.

4.2 Implementierung der Szene StartScreen

Die Szene „*StartScreen*“ fungiert als Hauptmenü (siehe Abb. 14) der Anwendung und soll dem Nutzer einen benutzerfreundlichen und einfachen Einstieg in das Programm verschaffen. Die Benutzeroberflächen wurden mit Hilfe von *Unity*-UI-Objekten erstellt. Diese generieren die grafischen Oberflächen, wie z.B. Tasten, während die Logik dahinter mit Hilfe von C#-Funktionen implementiert wird. Die Ausführung des Hauptemenüs wird durch die C#-Klasse „*FileLoader*“ gesteuert, während die Klasse „*UserPreferences*“ für die Speicherung der ausgewählten Optionen zuständig ist. Die Hintergrundbilder der Menüseiten entstammen der lizenzfreien Quelle *Pixabay* [44].

Abb. 14: Bildschirmaufnahme des Hauptmenüs innerhalb der Anwendung

Ein wichtiger Aspekt des Menüs sind die Auswahlelemente innerhalb des Fensters „*Optionen*“. Diese ermöglichen dem Nutzer vor Ausführung der Simulation festzulegen, welche Elemente geladen und angezeigt werden sollen. Bei der Datenverarbeitung und Generierung des Modells greift die Anwendung nur auf die Datenelemente zurück, welche zuvor in den Optionen ausgewählt wurden. Dies ermöglicht wiederum auch auf

hardwareschwächeren Geräten eine effiziente Ausführung des Programms. Des Weiteren können größere Quelldateien der Anwendung übergeben werden, da nicht alle Elemente zwischengespeichert werden müssen. Innerhalb der Optionen kann der Nutzer folgende Elemente auswählen: ÖPNV Schienen, Busstraßen, Alle Straßen, Stationen, 3D Gebäude, Untergrundbahnen, Straßenbahnen, Züge, Eisenbahnen Stadtbahnen, Busse. Diese Auswahlmöglichkeiten werden in Form von Kontrollkästchen aufgeführt und sind standardmäßig alle ausgewählt. Die ausgewählten Optionen werden innerhalb der C#-Klasse „*UserPreferences*“ in Form von statischen Variablen gespeichert, welche später bei der Datenverarbeitung ausgelesen werden können.

Der Kernpunkt des Hauptmenüs ist das Fenster „*Simulation starten*“. Hier muss innerhalb des Eingabefensters ein Dateipfad als Datenquelle angegeben werden, um die Simulation zu starten. Dabei überprüft das Programm, ob der Dateipfad existiert bzw. ob es sich bei der Datei um eine „*.txt*“-Datei handelt. Ist dies nicht der Fall, so wird eine entsprechende Fehlermeldung ausgegeben. Textdateien, welche nicht im XML Format vorliegen, können ebenfalls erkannt werden. Diese werden im nächsten Schritt in der Szene „*Simulator*“ beim Versuch der Datenverarbeitung als „nicht XML-Struktur“ erfasst und ebenfalls in Form einer Fehlermeldung zurückgewiesen.

Während der Datenverarbeitung wird dem Nutzer ein Ladefenster angezeigt, welches sich innerhalb der Szene „*StartScreen*“ befindet. Sobald alle Daten erfolgreich verarbeitet wurden und die benötigten Datenstrukturen bereit sind, erfolgt der Übergang zum Modell.

Die Abb. 15 visualisiert die Prozesse innerhalb der Szene „*StartScreen*“ und die Schnittstellen zur Szene „*Simulator*“.

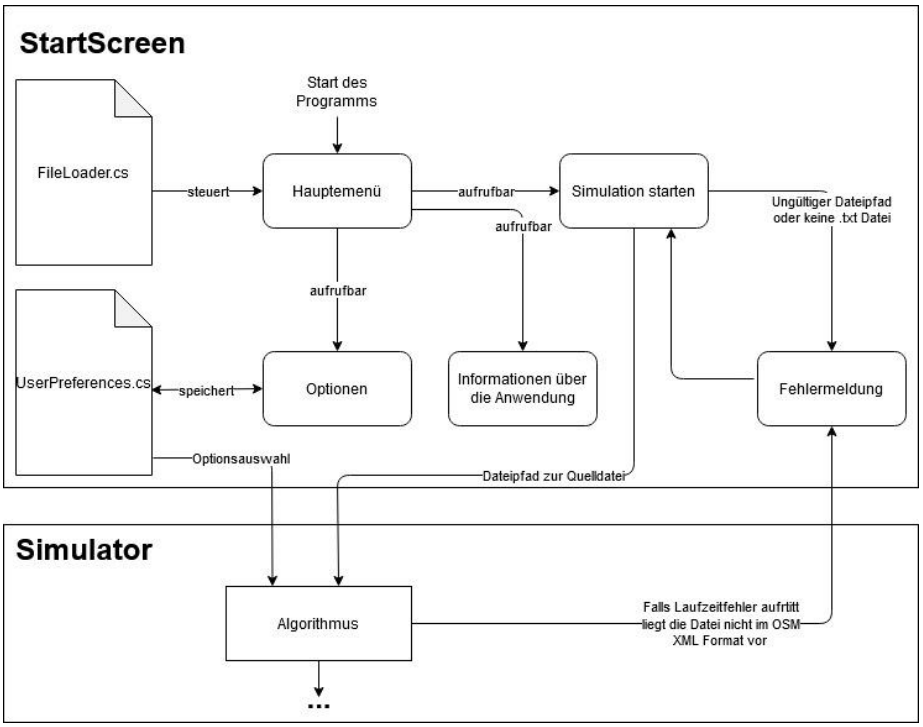


Abb. 15: Struktur der Szene StartScreen und Schnittstellen zur Szene Simulator

4.3 Implementierung der Szene Simulator

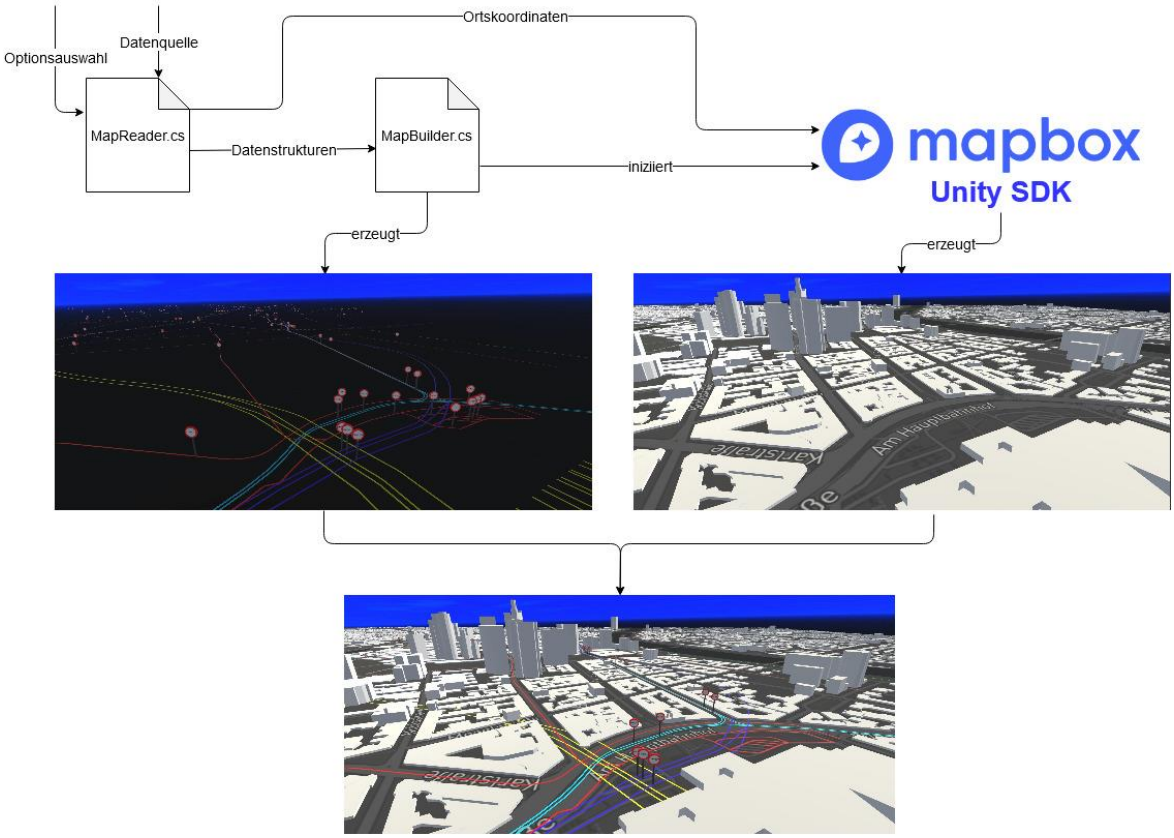


Abb. 16: Generierung der Objekte durch MapBuilder.cs und Mapbox Unity SDK

Nachdem ein gültiger Dateipfad zu einer OSM XML Quelldatei angegeben wurde, beginnt die Datenverarbeitung durch die Klasse „*MapReader*“. Dabei werden nur die Daten verarbeitet, welche in den Optionen ausgewählt wurden. Sobald sich alle benötigten Informationen innerhalb der Datenstrukturen befinden, beginnt der Prozess der Objekterzeugung. Standardmäßig werden folgende Objekte durch die Klasse „*MapBuilder*“ generiert:

- Straßen, welche durch Buslinien befahren werden
- Schienen, welche durch den Schienenpersonennahverkehr befahren werden (Untergrundbahnen, Straßenbahnen, Züge, Eisenbahnen und Stadtbahnen)
- Alle zugehörigen Stationen samt UI mit den Nahverkehrsinformationen

Dabei gilt es zu beachten, dass die Straßen und Schienen abhängig vom Verkehrsmittel in verschiedenen Farben visualisiert werden. Des Weiteren verfügen die Stationsobjekte über eigene Benutzeroberflächen und zeigen dem Nutzer die jeweiligen ÖPNV-Linien an, welche die Station anfahren.

Parallel dazu werden die 3D-Gebäude und die zugrundeliegende Karte mit allen Straßenmarkierungen durch die *Mapbox SDK für Unity* generiert. Diese erhält durch die Klasse „*MapReader*“ die benötigten Koordinaten, um die Daten und Objekte des jeweiligen Ortes zu laden. Im Zuge dessen musste der Quellcode der *Mapbox SDK* umgeschrieben werden, um eine dynamische Generierung zur Laufzeit des Programmes zu ermöglichen.

Abb. 16 visualisiert den Schritt, von der Datenverarbeitung zur Objekterzeugung und zeigt die entsprechenden Objekte, welche durch „*MapBuilder*“ und *Mapbox Unity SDK* generiert werden.

Standardmäßig wird bei der Ausführung der Szene „*Simulator*“ ein UI geladen. Dieses enthält unter anderem Steuerungshinweise, eine Legende mit der Farberläuterung, eine Suchfunktion und weitere Aktionsmöglichkeiten. Der Nutzer kann z.B. die Gebäude ausblenden (für eine bessere Übersicht) oder zwischen Tag- und Nachtmodus wechseln (siehe Abb. 17).

Der Höhepunkt des Modells sind die interaktiven Stationsobjekte. Diese werden in Form von Straßenschildern mit der Überschrift „Station“ dargestellt und können durch den Nutzer ausgewählt werden. In Folge dessen erscheint ein Stations-UI mit dem jeweiligen Stationsnamen und einer Dropdown-Auswahl der ÖPNV-Linien, welche die Station befahren. Nachdem der Nutzer eine ÖPNV-Linie aus der Liste ausgewählt hat, wird der dazugehörige Weg markiert und es aktivieren sich alle Stationen entlang der Strecke (es erscheint die UI mit dem Stationsnamen und der ÖPNV-Linie wie in Abb. 17). Je nach Verkehrsmittel werden ebenfalls Zug- bzw. Busmodelle geladen und befahren die entsprechende Strecke. Diese werden an der Anfangsstation generiert und halten an jeder Station entlang des Weges kurz an.

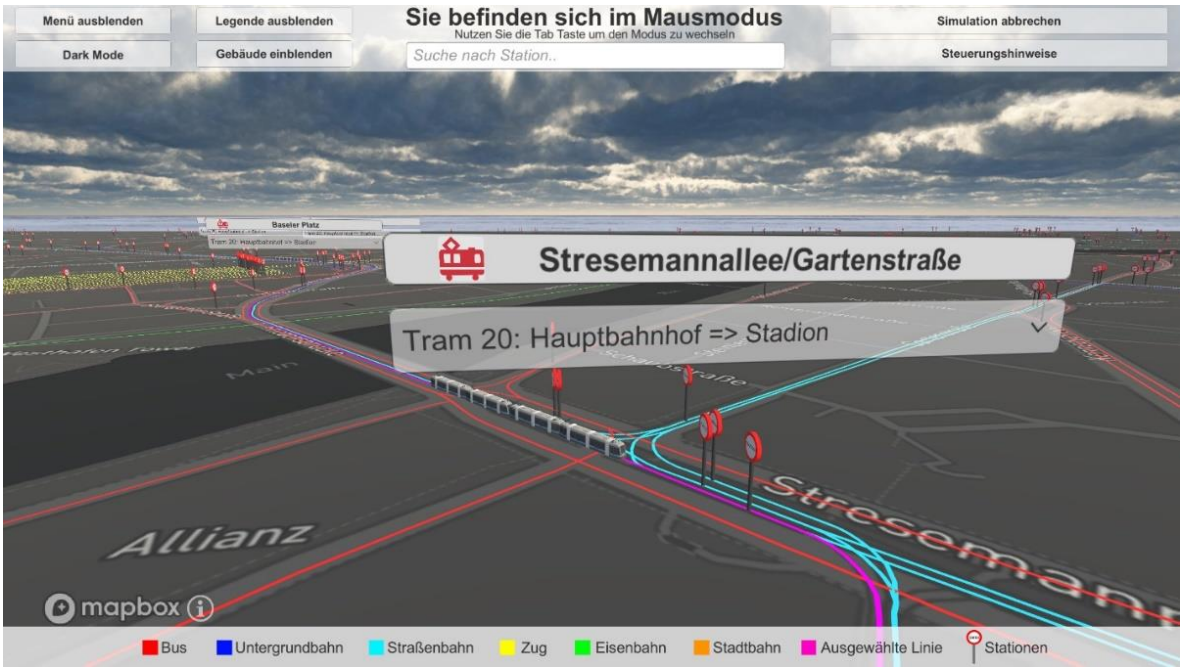


Abb. 17: Stations-UI mit einer ausgewählten ÖPNV-Linie, Bildschirmaufnahme innerhalb der Anwendung

4.3.1 Analyse und Verarbeitung der OSM XML Quelldatei

Die Analyse und Verarbeitung der Quelldatei wird vollständig durch das Skript „*MapReader*“ gesteuert. Dabei werden die Grund-Datensatztypen (Bounds, Node, Way und Relation) aus der Quelldatei ausgelesen und mit Hilfe der Klassen innerhalb des „Serialization“-Ordners in Instanzen der jeweiligen Klasse umgewandelt. Danach werden die Instanzen in Dictionarys und Listen abgelegt, um später bei der Objekterzeugung darauf zuzugreifen (siehe Abb. 18).

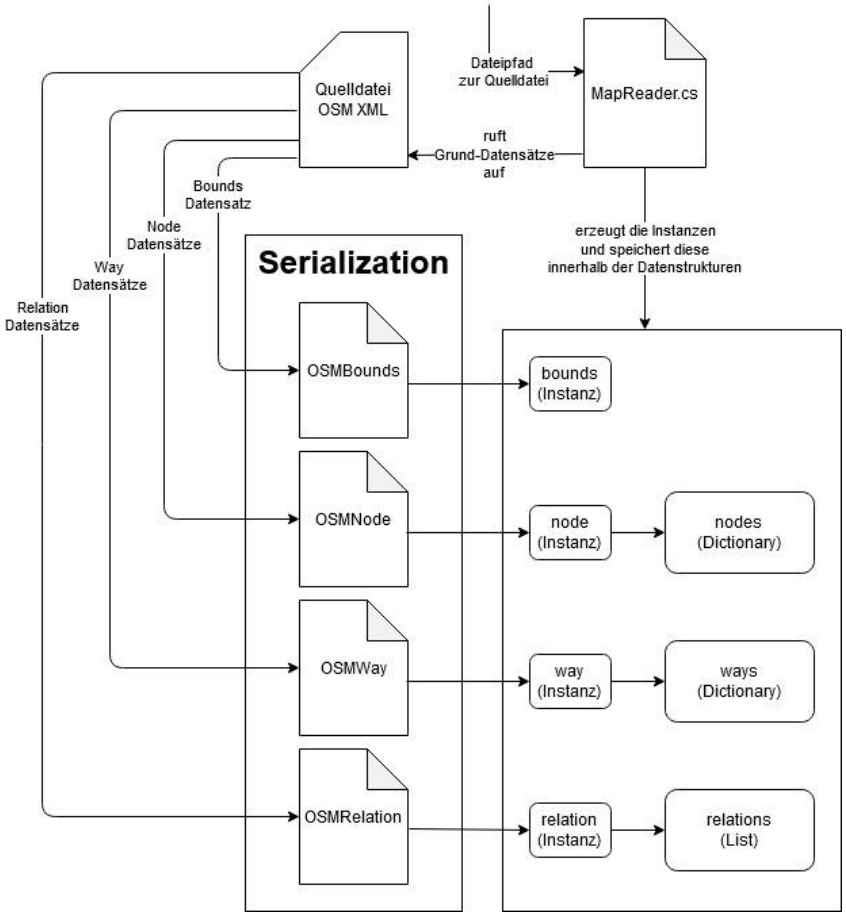


Abb. 18: Visualisierung der Datenerhebung und Verarbeitung durch MapReader.cs

Die Basisstruktur für die Datenverarbeitung und die Objekterzeugung wurde aus dem Werk „*Scene Based Real World Map Data*“ [20] von Sloan Kelly entnommen. Dieses beinhaltet die Klassen „*OSMBounds*“, „*OSMNode*“, „*OSMway*“ aus dem Serialization-Ordner, die Klasse „*MapReader*“ für die Datenverarbeitung und die Klasse „*MapBuilder*“ für die Objekterzeugung. Die Klassen wurden entsprechend den Anforderungen dieses Projekts weiterentwickelt und angepasst, jedoch blieb die Vorgehensweise und Funktionalität unberührt. Für die Datenanalyse wurde die offizielle *OSM*-Dokumentation [36] hinzugezogen, auf welcher die Rechtlinien für die Darstellung des ÖPNV mit Hilfe der

Grund-Datensatztypen verzeichnet sind. Anhand dieser Dokumentation konnte abgeleitet werden, nach welchen Daten gezielt innerhalb der Quelldatei gesucht werden muss.

Als erstes wird der Bounds-Datensatz geladen. Es wird eine Instanz der Klasse „*OSMBounds*“ erzeugt, welche die 4 Grenzmarkierungen („*minlat*“, „*minlon*“, „*maxlat*“, „*maxlon*“) aus dem Datensatz zwischenspeichert. Anhand dieser Daten wird das Zentrum berechnet (die Zentrums-Koordinaten) und dient später als Referenzpunkt bei der Objekterzeugung durch die Klasse „*Mapbuilder*“ und die *Mapbox SDK*. In Kapitel 4.3.2 wird das Verfahren der Objekterzeugung durch die Zentrumskoordinaten näher beschrieben. Des Weiteren wird anhand der Grenzmarkierungen die Größe des ausgewählten Ortes berechnet. Diese Daten werden ebenfalls für die Objekterzeugung durch die *Mapbox SDK* benötigt und spezifizieren die Anzahl der Kachelsätze, welche geladen werden sollen. Abb. 19 visualisiert die Klasse „*OSMBounds*“ und die zu verarbeitenden Daten.

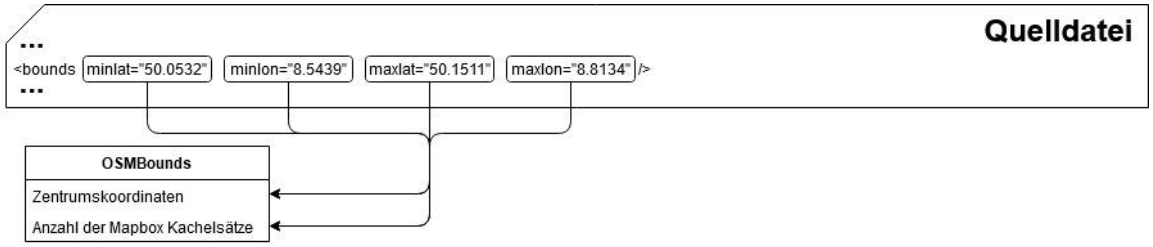


Abb. 19: Verarbeitung des Bounds-Datensatzes durch die Klasse OSMBounds.cs

Als Nächstes werden die Node-Datensätze aus der Quelldatei geladen. Diese werden der Reihe nach abarbeitet um die dazugehörigen Instanzen der Klasse „*OSMNode*“ zu erzeugen. Im Zuge dessen wird die ID und die geografische Position („*lat*“ und „*long*“) hinterlegt. Ein Node kann neben der Zusammensetzung eines Ways in Form einer Schiene/Straße auch individuell in Form einer Station vorkommen. Für solche Fälle werden die Tags der Node-Datensätze nach dem Schlüssel *k=“name“* durchsucht. Der dazugehörige Wert wird als Stationsname gespeichert. Jede Instanz der Klasse „*OSMNode*“ wird in ein Dictionary abgelegt, wobei ihre ID als Schlüssel dient, um auf die Instanz später zuzugreifen. Abb. 20 visualisiert einen Node-Datensatz als Beispiel der Datenerhebung und Verarbeitung.

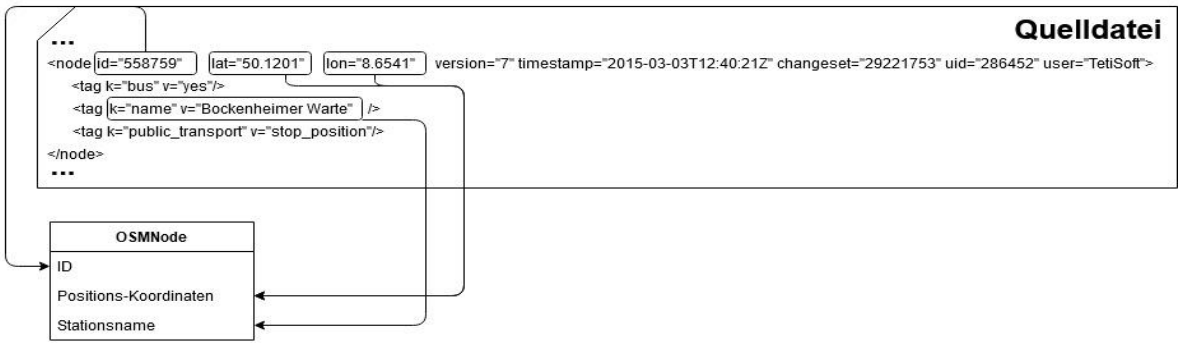


Abb. 20: Vearbeitung der Node-Datensätze durch die Klasse OSMNode.cs

Ähnlich zu den Node-Datensätzen werden auch die Way-Datensätze verarbeitet. Hierbei wird die Klasse „*OSMWay*“ für die Erzeugung der Instanzen verwenden und ebenfalls die ID für jede Instanz übernommen. Als nächstes werden die Tags nach dem Schlüssel *k=„railway“* bzw. *k=„highway“* durchsucht. Wird ein solcher Tag gefunden, wird die Instanz als Schiene bzw. Straße markiert. Des Weiteren werden dann die dazugehörigen *nd*-Datensätze geladen und deren „*ref*“-Werte (Node IDs) innerhalb einer Liste abgelegt. Am Ende werden nur Instanzen mit der Markierung als Schiene bzw. Straße in das Dictionary *ways* gespeichert. Die ID der Instanz dient hier ebenfalls als Schlüssel für das Dictionary. In Abb. 21 wird ein Way-Datensatz, welcher eine Schiene darstellt, aufgeführt und entsprechend verarbeitet.

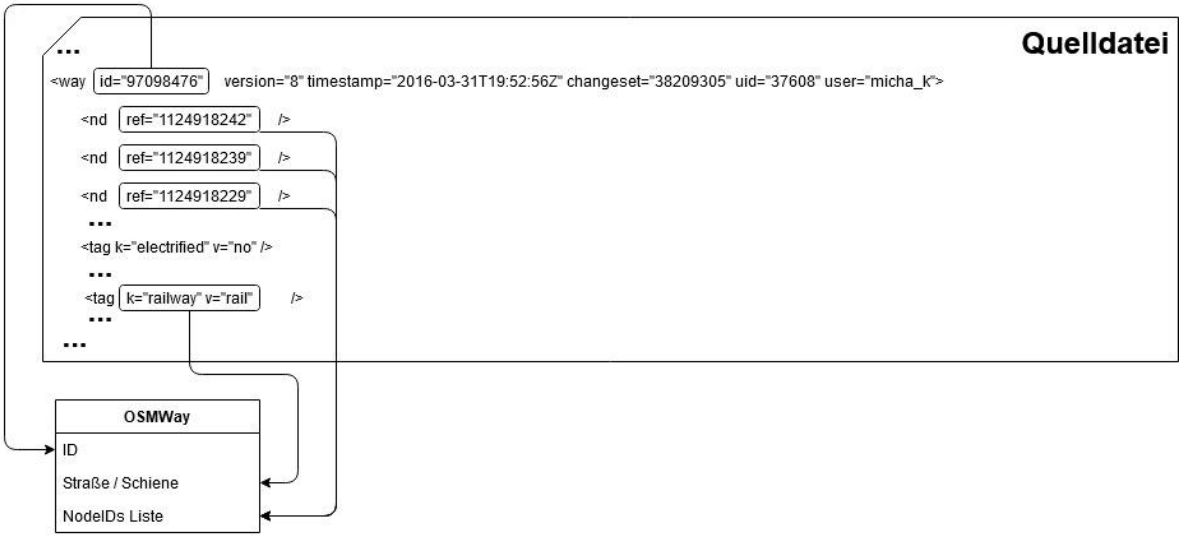


Abb. 21: Verarbeitung der Way-Datensätze durch die Klasse OSMWay.cs

Als Letztes gilt es noch die Relation-Datensätze zu verarbeiten. Diese sind insbesondere für die Darstellung der ÖPNV-Linien wichtig. Es wird innerhalb der Datensätze nach einem Tag mit dem Schlüssel *k=“Route“* gesucht. Wird ein solcher Tag gefunden, wird der entsprechende Wert geladen und nach einem der folgenden Wörter durchsucht: *v=“subway“, v=“tram“, v=“train“, v=“railway“, v=“light_rail“, v=“bus“*. Entspricht der Wert des Tags ebenfalls einem dieser Wörter, wird die Instanz als ÖPNV-Linie hinterlegt und das entsprechende Verkehrsmittel innerhalb der Instanz vermerkt. Um den Namen der ÖPNV-Linie zu speichern, werden dann ebenfalls die Tags durchsucht. Am Ende müssen nur noch die Mitglieder der ÖPNV-Relation gesichert werden. Bei diesen werden nur Nodes mit der Rolle „*stop*“ gespeichert, da diese die Stationen der ÖPNV-Linie darstellen. Des Weiteren werden nur Ways ohne die Rolle „*Platform*“ gesichert, da diese die Schienen bzw. Straßen darstellen. Hierbei ist zu beachten, dass nur mit Instanzen, welche als ÖPNV-Linie markiert sind, gearbeitet wird. Diese werden im nächsten Schritt in die „*relations*“-Liste abgelegt.

Nachdem eine Relations-Instanz erzeugt wurde (und als ÖPNV-Linie markiert ist), werden die entsprechenden ÖPNV-Daten der Instanz an ihre Mitglieder weitergeleitet. Somit wird

z.B. in jeder Node-Instanz ersichtlich, zu welcher ÖPNV-Linie es gehört. Hierbei ist ebenfalls zu beachten, dass alle Way-Instanzen aus der Liste WayIDs als ÖPNV-Straße bzw. ÖPNV-Schiene vermerkt werden. Die Verarbeitung eines beispielhaften Relation-Datensatzes wird in Abb. 22 visualisiert.

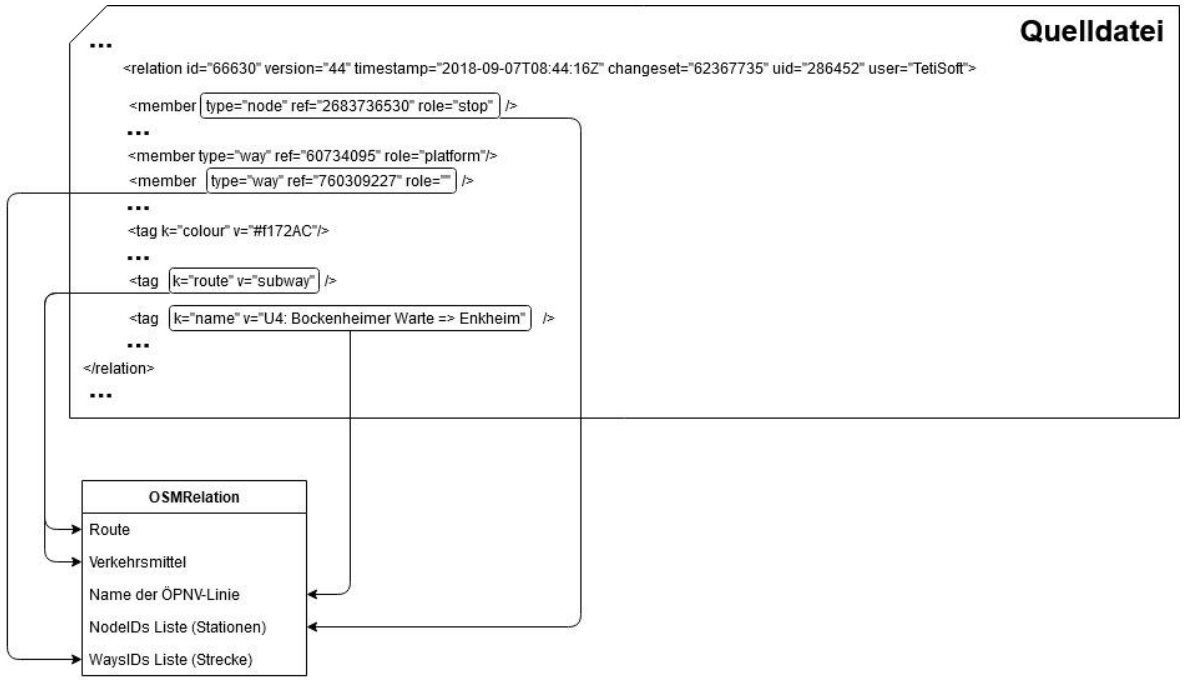


Abb. 22: Verarbeitung der Relation-Datensätze durch die Klasse OSMRelation.cs

Nachdem alle Daten verarbeitet wurden, werden alle Way-Instanzen innerhalb des *ways*-Dictionary, welche nicht als ÖPNV-Schiene oder ÖPNV-Straße hinterlegt sind, nicht mehr benötigt und können bei der Objektgenerierung ignoriert werden.

4.3.2 Generierung der Objekte

Die Objekterzeugung wird vollständig durch die Klasse „*MapBuilder*“ gesteuert und erfolgt in drei Phasen. In der ersten Phase werden die Stations-Objekte geladen und an den jeweiligen Positionen der ÖPNV-Linien platziert. Dabei wird für jedes Stations-Objekt ein entsprechendes UI geladen und mit den dazugehörigen Daten versehen. In der zweiten Phase werden die 3D-Gebäude und die zugrundeliegende Karte mit den Straßenmarkierungen durch die *Mapbox SDK* generiert. Die dritte Phase ist die zeit- und ressourcenaufwendigste Phase. Hierbei werden mit Hilfe des *ways*-Dictionary die Schienen und Busstraßen erzeugt. Parallel dazu wird ein Ladefenster mit dem aktuellen Fortschritt der Objektgenerierung angezeigt.

Um die Objektposition der einzelnen Elemente zu bestimmen, müssen deren geografische Breite und Länge aus der Quelldatei mit Hilfe der Node-Datensätze ausgewertet werden.

Da es sich hier um das Koordinatensystem einer Sphäre handelt, ist es wichtig diese Daten in ein Koordinatensystem umzuwandeln, welches einer geraden Fläche entspricht. Hierfür wird die *Mercator Projection* genutzt, um eine Konvertierung in *Unity* Koordinaten durchzuführen. Eine entsprechende Implementierung der *Mercator Projection (*für *Unity)* wurde von der offiziellen OSM-Webseite [37] entnommen. Nachdem die Konvertierung erfolgt ist, besteht noch immer das Problem, dass diese Koordinaten sehr groß sein können und dadurch sehr weit entfernt vom *Unity* Zentrum (Koordinaten (0, 0, 0)) ausfallen. Für eine optimale Ausführung sollten alle Objekte, unabhängig davon wo sie sich auf der Weltkarte befinden, um das *Unity* Zentrum herum generiert werden. Dazu wird ein einfaches Verfahren angewandt. Anhand des bounds-Datensatzes aus der Quelldatei werden die Zentrumskoordinaten des jeweiligen Gebiets berechnet. Nun werden bei der Generierung der Objekte die entsprechenden Koordinaten des Objekts durch die Zentrumskoordinaten subtrahiert. Dadurch fallen die neu berechnet Koordinaten deutlich geringer aus und werden um das *Unity*-Zentrum herum angeordnet.

Im Folgenden werden die drei Phasen der Objektgenerierung näher erläutert.

Phase 1 beginnt nachdem die Klasse „*MapReader*“ die Datenstrukturen vollständig verarbeitet hat und ein Startsignal an die Klasse „*MapBuilder*“ versendet hat. Als Erstes wird die Liste *relations* durchiteriert, um auf alle ÖPNV-Linien zuzugreifen. Dabei wird für jede einzelne ÖPNV-Linie pro Eintrag in der NodeIDs Liste (Stationen) ein Stationsobjekt erzeugt. Anhand der Node ID wird die jeweilige Stationsposition aus dem Dictionary *nodes* aufgegriffen und das Stationsobjekt entsprechend platziert. Des Weiteren wird pro Station ein UI geladen, welches mit dem Stationsobjekt gekoppelt ist und sich im dreidimensionalen Raum oberhalb des Stationsobjekts befindet. Als Nächstes werden nun die Daten an das UI übertragen. Dabei wird der Stationsname in dem UI übernommen, während die ÖPNV-Linien, welche die Station anfahren, in eine Dropdown-Liste abgelegt werden. Als Letztes wird nur noch das Verkehrsmittel abgefragt. Handelt es sich um eine Busstation, wird ein Bussymbol auf der UI aktiviert. Handelt es sich jedoch um eine Schienenstation, erscheint ein Zugsymbol.

In Phase 2 wird die Objekterzeugung durch die *Mapbox SDK* initiiert. Diese bietet dem Entwickler grundsätzlich an, vor der Ausführung des Programms zu spezifizieren, welche Koordinaten geladen und wie groß (Anzahl der Kachelsätze) das Modell ausfallen soll. Die Eingabe erfolgt in Form eines Entwickler-UI und wird bei der Ausführung des Programms entsprechend verarbeitet. Da in dieser Anwendung jedoch die Eingabe der Quelldatei samt Koordinateninformationen durch den Benutzer erfolgt, sprich diese Informationen erst zur Laufzeit bereitstehen, musste der Quellcode der *Mapbox SDK* umgeschrieben werden, um eine dynamische Ausführung zur Laufzeit des Programmes zu ermöglichen. Dazu wurde innerhalb des Quellcodes statt der Informationsübernahme aus dem Entwickler-Eingabefenster, eine Informationsübernahme aus der bounds-Instanz (Klasse „*MapReader*“) implementiert. Sobald Phase 2 beginnt, wird das Mapbox-Objekt erzeugt und das dazugehörige Softwaremodul ausgeführt. Dieses erhält nun zur Laufzeit des Programmes die Zentrumskoordinaten der Quelldatei und die Anzahl der Kachelsätze, welche geladen werden sollen.

In der dritten und letzten Phase gilt es noch die Schienen und Busstraßen zu erzeugen. Dafür wird das *ways*-Dictionary durchiteriert und jede Way-Instanz einzeln ausgewertet. Als Erstes wird ermittelt welches Verkehrsmittel die Way-Instanz nutzt und anhand dessen wird die Färbung des Objekts eingestellt. Als Nächstes wird dann die Liste NodeIDs durchiteriert, um die Strecke entsprechend abzubilden. Dazu wird der Reihe nach jeweils ein Punkt (Node) hinzugezogen und mit dem vorherigen Punkt (Node) verbunden. Damit die Verbindung zwischen zwei Punkten keine dünne Linie ist, wird ein Rechteck mit einer voreingestellten Breite von einem Punkt zum anderen Punkt gezogen. Dies wird dann für alle Punkte der Instanz durchgeführt bis die Strecke fertig abgebildet wurde. Dieser Generierungsprozess ist *Unity*-bedingt nicht optimiert und bedarf deshalb einer deutlich längeren Ladezeit als die vorherigen zwei Phasen. Im Zuge dessen wurde ein Ladefenster implementiert, dass den aktuellen Fortschritt anzeigen soll. Dabei wird in jedem Schritt die Anzahl der bisher durchlaufenen Way-Instanzen durch die Gesamtanzahl der Way-Instanzen im *ways* Dictionary dividiert und als Prozentsatz angezeigt.

Während Phase 3 kann sich der Nutzer die Steuerungshinweise durchlesen und sich innerhalb des Modells frei bewegen, jedoch können keine weiteren Aktionen durchgeführt werden bevor nicht alle Objekte generiert wurden.

Abb. 23 visualisiert die drei Phasen der Objektgenerierung und zeigt dabei die verwendeten Datenstrukturen und die generierten Objekte an.

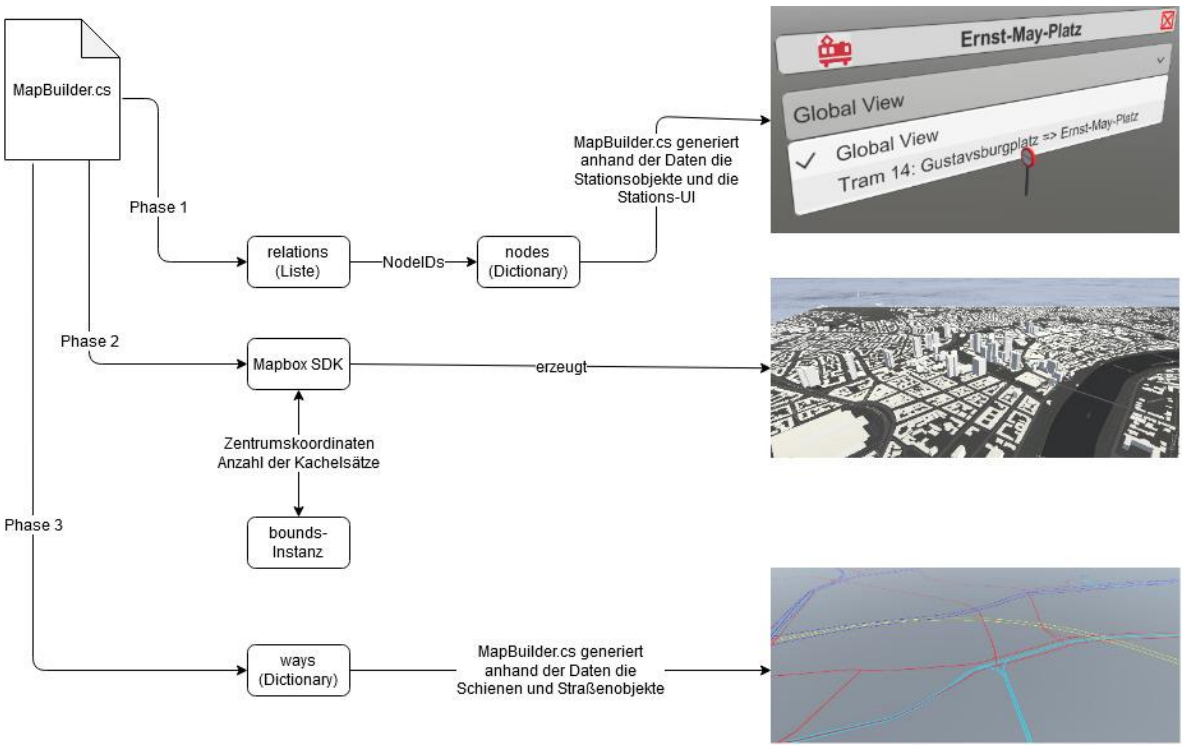


Abb. 23: Visualisierung der drei Phasen der Objektgenerierung und der entsprechenden Datenquellen

4.3.3 Implementierung der Steuerung und der UI

Innerhalb der Simulation wird grundsätzlich zwischen zwei Modi unterschieden: dem Mausmodus, mit dessen Hilfe der Nutzer Elemente aus dem UI auswählen bzw. mit den Stations-Objekten interagieren kann, und dem Flugmodus, welcher ausschließlich zur Erkundigung des Modells dient und die Position des Anwenders innerhalb der Simulation ändert. Hinzu kommt, dass während des Flugmodus die Mausanzeige ausgeblendet wird und stattdessen ein Fadenkreuz erscheint, welches das Navigieren erleichtern soll. Durch Drücken der Tab-Taste, kann zwischen den zwei Modi gewechselt werden.

Um dem Nutzer eine möglichst hohe Orientierung innerhalb der Anwendung zu verschaffen, bedarf es einer Steuerung, welche maximale Bewegungsfreiheit innerhalb des Stadtmodells ermöglichen soll. Diese muss aber zugleich benutzerfreundlich und einfach zu verstehen sein. Im Zuge dessen wurde eine typische Steuerungsvariante für den Flugmodus implementiert, welche bereits bei zahlreichen Spielen und Modellierungsprogrammen angewendet wird. Mit Hilfe der [W] und [S]-Tasten kann sich der Nutzer nach vorne oder nach hinten bewegen, während ihn die Tasten [A] und [D] nach links oder rechts bewegen. Des Weiteren hat die Maus eine zentrale Rolle und dient zur Manipulation des Kamerablickwinkels. Durch diese Steuerungsvariante wird dem Nutzer ein typischer Flugmodus ermöglicht, wobei dieser die Stadt überfliegen kann und komplette Bewegungsfreiheit besitzt. Eine entsprechende Unity-Implementierung wurde durch den github-Nutzer „*gunderson*“ bereitgestellt [15] und in diesem Projekt in Form der Klasse „CameraController“ integriert.

Das Haupt-UI setzt sich aus zwei Abschnitten zusammen. Während die obere Leiste verschiedene Aktionen mit Hilfe von Tasten ermöglicht, wird in der unteren Leiste eine Farblegende eingeblendet, welche die ÖPNV-Verkehrsmittel beschreibt. Innerhalb der oberen Leiste können folgende Aktionen durchgeführt werden: Menü ausblenden, zwischen Tag- und Nachtmodus wechseln, Gebäude ausblenden, Legende ausblenden, Steuerungshinweise anzeigen, die Simulation abbrechen oder nach einer bestimmten Station suchen. Mit Hilfe der Suchfunktion kann der Nutzer einen Suchbegriff (Stationsnamen) eingeben und bekommt eine Liste mit Suchvorschlägen angezeigt. Sobald ein Eintrag aus dieser Liste ausgewählt wurde, wird der Nutzer zur Station teleportiert und das dazugehörige Stations-UI aktiviert.

4.3.4 Funktionslogik der interaktiven Stations-Objekte

Nachdem der Nutzer ein Stations-Objekt auswählt, erscheint das dazugehörige UI mitsamt der Stationsbezeichnung und einer Dropdown-Auswahl der ÖPNV-Linien. Sobald der Anwender einen Eintrag aus dem Dropdown-Menü ausgewählt hat, wird ein Hintergrundprozess ausgeführt, welcher die Strecke der ÖPNV-Linie markiert und alle dazugehörigen Stationen aktiviert. Dieser Prozess soll in diesem Unterkapitel behandelt werden. Des Weiteren wird, sobald der Nutzer eine ÖPNV-Linie ausgewählt hat, ein Bus-

bzw. Zugmodell geladen, welches die entsprechende Strecke befährt. Dieser Prozess wird wiederum im Unterkapitel 4.3.5 behandelt.

Es wird zu Erklärungszwecken angenommen, dass der Anwender zum ersten Mal seit Anwendungsstart eine Dropdownauswahl innerhalb der Stations-UI tätigt. Nachdem dies geschehen ist, steht dem Programm nur der Name der ausgewählten ÖPNV-Linie zur Verfügung. Mit Hilfe dieser Information iteriert der Algorithmus durch alle Objekte innerhalb der Szene und durchsucht die Straßen- bzw. Schienenobjekte nach einem Eintrag der ÖPNV-Linie. Dieser Eintrag befindet sich für jedes Straßen- und Schienenobjekt innerhalb einer Unity-Textkomponente (in den Metadaten des Objekts). Sobald innerhalb eines Objekts ein derartiger Eintrag gefunden wurde, beginnt der Prozess der Wegmarkierung. Um den Weg später wiederherzustellen, wird die aktuelle Färbung und eine Objektreferenz zwischengespeichert. Im nächsten Schritt wird das Objekt umgefärbt und leicht angehoben (Höhe von 0 auf 0.1), um eine bessere Übersicht zu ermöglichen. Parallel dazu wird während der Iteration aller in der Szene sich befindlichen Objekte ebenfalls nach Stationen mit dem entsprechenden ÖPNV-Eintrag gesucht. Wird ein solches Stations-Objekt gefunden, wird seine UI aktiviert und die entsprechende ÖPNV-Linie angezeigt. Die gefundenen Stationsobjekte werden ebenfalls in einer Liste zwischengespeichert, um später die Auswahl rückgängig zu machen.

Um die aktuelle Auswahl (Wegmarkierung) einer ÖPNV-Linie abzubrechen, hat der Anwender zwei Möglichkeiten. Innerhalb jeder Dropdown-Auswahl eines jeden Stations-Objekts befindet sich ein „Global view“-Eintrag. Wird dieser bei einer Station ausgewählt, wird eine entsprechende Reparatur der Wegmarkierung eingeleitet. Diese findet ebenfalls statt, sobald der Nutzer eine andere ÖPNV-Linie auswählt. Die Reparatur der Wegmarkierung erfolgt aufgrund der zwischengespeicherten Information sehr effizient. Alle Straßen bzw. Schienen werden wieder auf die Höhe 0 eingestellt und deren Färbung wird angepasst. Des Weiteren werden die UIs der Stations-Objekte wieder deaktiviert und die Auswahl auf „Global view“ zurückgesetzt.

In Abb. 24 wird der beschriebene Prozess in Form eines Automaten visualisiert wobei der rechte Zustand eine Eigenschleife besitzt.

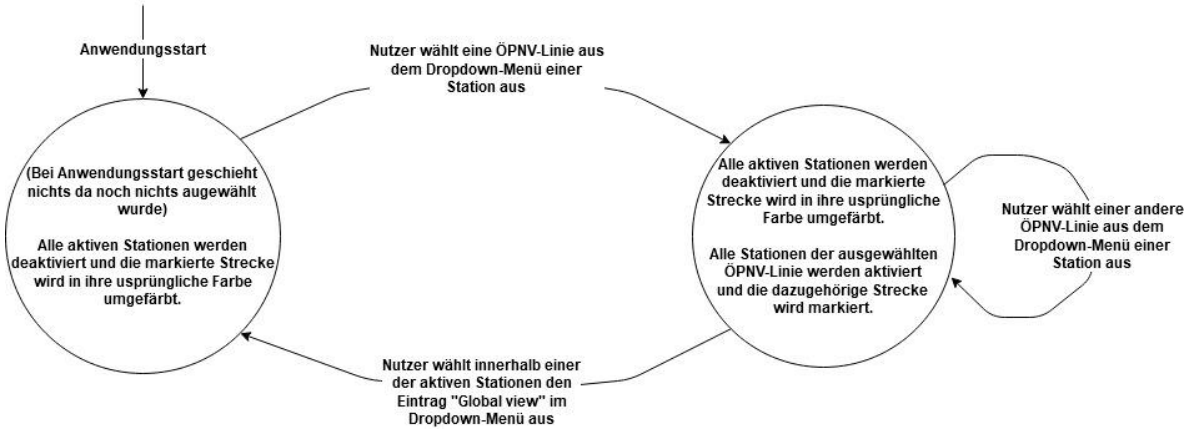


Abb. 24: Visualisierung der Logik der interaktiven Stations-Objekte mit Hilfe eines Automaten

4.3.5 Implementierung der Bus- und Zuganimationen

Im vorherigen Unterkapitel wurde die Funktionslogik der interaktiven Stations-Objekte erläutert. Dabei werden, sobald der Nutzer eine ÖPNV-Linie aus dem Dropdown-Menü ausgewählt hat, die entsprechenden Way-Objekte markiert und zwischengespeichert (für die Wiederherstellung in das ursprüngliche Format). Nun gilt es mit Hilfe der zwischengespeicherten Way-Objekte ein Bus- bzw. Zugmodell auf der entsprechenden Strecke fahren zu lassen. Dazu wird die Eigenschaft der Zusammensetzung eines Ways ausgenutzt. Ein Way besteht aus mehreren Nodes (Punkte mit entsprechenden Koordinaten). Sortiert man die Way-Objekte sodass eine bestimmte Strecke abgebildet wird, kann man diese Eigenschaft ausnutzen, um das Bus- bzw. Zugmodell von Punkt zu Punkt fahren zu lassen. Die Problematik hierbei ist, dass die Strecke unsortiert in der Datenquelle hinterlegt ist und ein entsprechender Algorithmus entwickelt werden musste, um diese Aufgabe des Sortierens zu übernehmen. Dabei musste insbesondere beachtet werden, dass je nach ÖPNV-Linie eine bestimmte Strecke auch in zwei Richtungen verlaufen kann und die sortierte Node-Liste in einer korrekten Reihenfolge erscheinen muss. Eine weitere Problematik bestand darin, dass durch die benutzerdefinierte Auswahl der Karte einige Strecken an einen Rand stoßen und später von einem anderen Rand aus fortgesetzt werden. Dadurch entstanden getrennte Abschnitte einer zusammengehörigen Strecke. Da die Funktionsweise dieses Algorithmus sehr umfangreich ist, wird der Sortieralgorithmus im Rahmen dieser Arbeit nicht weiterverfolgt und es wird auf die Programmdokumentation innerhalb des Quellcodes verwiesen. Nachdem die Strecke sortiert wurde, gilt es nur noch zu ermitteln, welche Farbe die jeweiligen Way-Objekte besaßen (bevor sie durch die Auswahl im Dropdown-Menü markiert und umgefärbt wurden). Anhand dieser Farbe kann das Verkehrsmittel identifiziert werden. Handelt es sich beispielsweise um die Farbe #FF0000 (rot), so wird ein Busmodell instanziiert, welches nun mit Hilfe der sortierten Node-Liste von Node zu Node fährt. Dabei wird beim Erreichen eines jeden Nodes geprüft, ob es sich um eine Station der ausgewählten ÖPNV-Linie handelt. Falls dies zutrifft, wird das jeweilige Bus- bzw. Zugmodell an der Station kurz anhalten. Sobald die Modelle den letzten Node aus der Liste erreichen, werden sie despawned. Abb. 25 veranschaulicht die genutzten Bus- und Zugmodelle. Das erste und dritte Modell (von links betrachtet) entstammen folgender Quelle [33][18], während die übrigen zwei Modelle eigens in Blender entwickelt wurden.

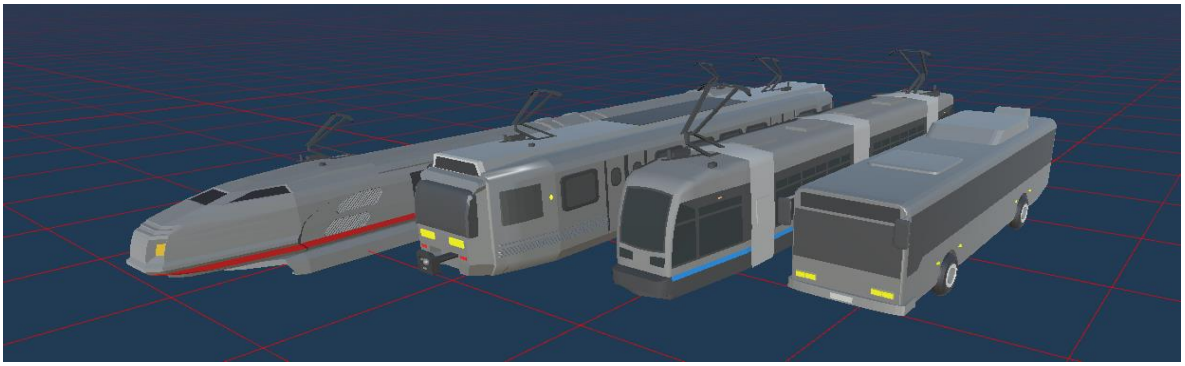


Abb. 25: Bildschirmaufnahme der genutzten Bus- und Zugmodelle innerhalb der Unity-Anwendung

Kapitel 5
Evaluation

5.1 Gestaltung der Evaluation

Die Evaluation wurde in Kooperation mit dem YouTube-Kanal „*ÖPNV Simulationen*“ [41] durchgeführt. Im Zuge dieser Kooperation wurde von den Betreibern des Kanals ein 30-minütiges Präsentationsvideo [42] entwickelt und veröffentlicht, welches die Anwendung vorstellt und die einzelnen Funktionalitäten der Simulation erläutert. Anschließend wurden die Zuschauer gebeten sich an der Evaluation zu beteiligen.

Die Evaluation erfolgte in zwei Teilen. Im ersten Teil sollte die Anwendung über einen Link heruntergeladen und ausgeführt werden. Innerhalb der Anwendung wurden diverse Aufgaben gestellt, damit sich der Nutzer mit allen Funktionalitäten des Programms vertraut machen kann. Im zweiten Teil sollte der Nutzer einen Fragebogen ausfüllen, welcher mit Hilfe der Online-Survey-Plattform *Typeform* [50] bereitgestellt und ausgewertet wurde. Die komplette Evaluation wurde über eine *Github*-Seite [49] administriert, um den Teilnehmern einen möglichst hohen Überblick über alle Ressourcen und Informationen zu verschaffen.

5.2 Inhalt der Evaluation

Innerhalb der Simulationsanwendung konnte der Nutzer standardmäßig eine vorkonfigurierte Karte der Stadt Frankfurt laden, die für die Durchführung der Evaluation benötigt wurde. An die Teilnehmer der Evaluation wurden folgende 5 Aufgaben gestellt:

1. *Starten Sie das Programm, indem sie die zip-Datei zuerst extrahieren und danach auf die Anwendung "OPNV Simulator.exe" klicken. Machen Sie sich mit den einzelnen Menüpunkten vertraut. Wenn Sie bereit sind, können Sie die Simulation durch Drücken der "Simulation starten"-Taste beginnen. Das darauffolgende Eingabefenster lassen Sie bitte leer und betätigen direkt die "START"-Taste.*

2. *Lesen Sie sich die Steuerungshinweise durch und spielen Sie ein bisschen mit der Steuerung herum während die Simulation lädt, um sich daran zu gewöhnen. Sobald die Simulation vollständig geladen wurde, suchen Sie nach der Station "Beuthener Straße". Nutzen Sie dafür die Suchfunktion in der oberen Leiste.*

3. *Wählen Sie an der Station "Beuthener Straße" die Linie "Tram 18: Louisa Bahnhof => Preungesheim" aus.*

4. *Folgen Sie dieser Strecke bis zur Station "Lokalbahnhof/Textorstraße" (7 Stationen Entfernung). Dort müssen Sie zur Buslinie "Bus 36: Hainer Weg => Westbahnhof" wechseln. Die dazugehörige Station befindet sich in unmittelbarer Nähe. Suchen Sie die Busstation und wählen Sie die entsprechende Buslinie aus.*

5. *Nutzen Sie die Menüschalter in der oberen Leiste, um den Dark Mode einzuschalten, und um die Gebäude auszublenden, damit Sie eine bessere Übersicht über die ausgewählte Strecke bekommen. Fahren Sie anschließend mit dieser Buslinie bis zur Station "Uni Campus Westend" (16 Stationen Entfernung).*

   Sie haben soeben den Campus der Goethe-Universität erreicht und somit erfolgreich alle Aufgaben bearbeitet. Bitte fahren Sie mit dem Fragebogen fort.

Der Fragebogen orientierte sich an der standardmäßigen UMUX-Bewertung [10]. Dabei konnten Aspekte wie z.B. die Steuerungskomplexität, aber auch subjektive Meinungen auf einer UMUX-Skala von 1 bis 7 bewertet werden. Insgesamt gab es vier Bewertungsfragen und eine Ja-/Nein-Frage. Anschließend konnte sich der Teilnehmer innerhalb der letzten 3 Fragen mit eigenen Worten äußern und beispielsweise eigene Ideen und Erweiterungsvorschläge angeben.

5.3 Ergebnis der Evaluation

Die Evaluation fand in der Zeit zwischen dem 15. und 19. Mai statt und hat insgesamt 34 Teilnehmer verzeichnet. In den nächsten Unterkapiteln folgt eine Analyse der Evaluationsergebnisse. Die Auswertung der Evaluationsergebnisse wurde durch die Online-Survey-Plattform *Typeform* bereitgestellt. Dabei ist es wichtig zu erwähnen, dass bei den Bewertungsaufgaben die Zahl 1 die Aussage „trifft gar nicht zu“ und die Zahl 7 die Aussage „trifft voll zu“ vertritt.

Da sich die 6. Und 7. Frage mit den Erweiterungsvorschlägen und Nutzungspotenzialen der Simulationsanwendung befassen, wurden die Erkenntnisse aus diesen beiden Fragestellungen jeweils zusammenfassend in Kapitel 6 „Zusammenfassung und Ausblick“ behandelt. Dabei werden die Inhalte aus Frage 6 in Unterkapitel 6.1.1 analysiert, während die Inhalte aus Frage 7 in Unterkapitel 6.1.2 vorkommen.

5.3.1 Bewertung der Steuerungskomplexität

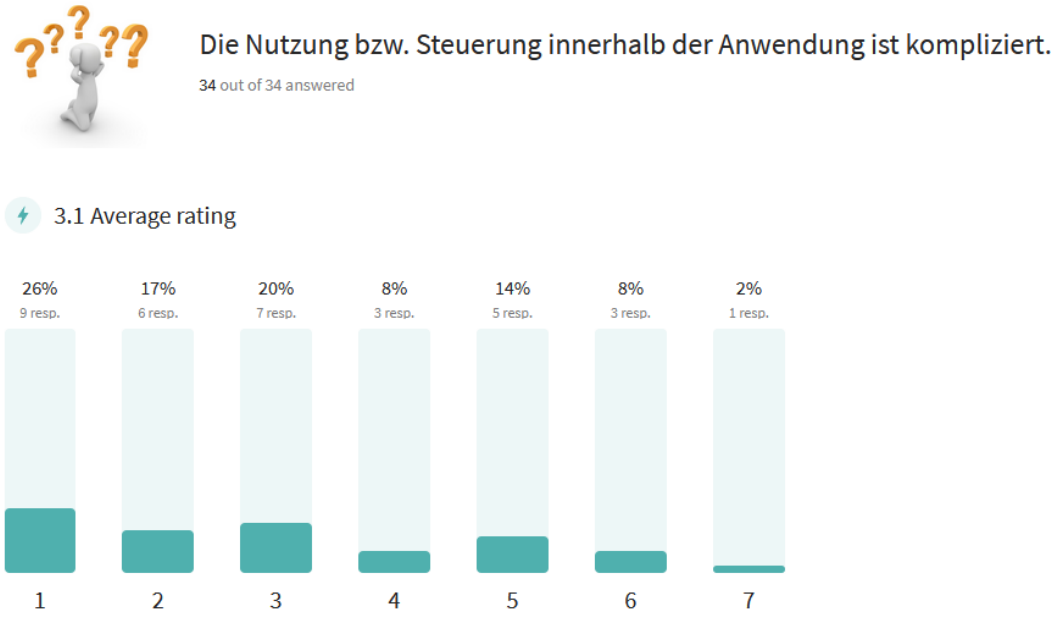


Abb. 26: Balkendiagramm des Evaluationsergebnisses für die 1. Frage, Bildschirmaufnahme der Fragebogenauswertung

Bei der ersten Frage wurde festgestellt, dass die Nutzer die integrierte Steuerung überwiegend für nicht zu komplex hielten, es jedoch Verbesserungswünsche gibt. Dementsprechend könnte eine andere Kombination aus Tastatur- und Mausbelegung die Nutzer zufrieden stellen. Hier müssten weitere Befragungen stattfinden, um die genaue Ursache der Probleme zu beleuchten. Des Weiteren könnten andere Plattformen einen neuen Steuerungsansatz verfolgen und dadurch die Steuerungskomplexität verringern, was am Ende zu einer höheren Nutzerzufriedenheit führen könnte. Beispielsweise könnte eine Handysteuerung, bei der sich der Nutzer durch Drehen und Kippen des Smartphones innerhalb des Modells bewegt, eine bessere Alternative zu der Tastatur- und Mausbelegung bieten. Der Einsatz von Controllern wäre ebenfalls eine mögliche Option.

5.3.2 Vergleich zwischen 3D-Modellen und 2D-Karten

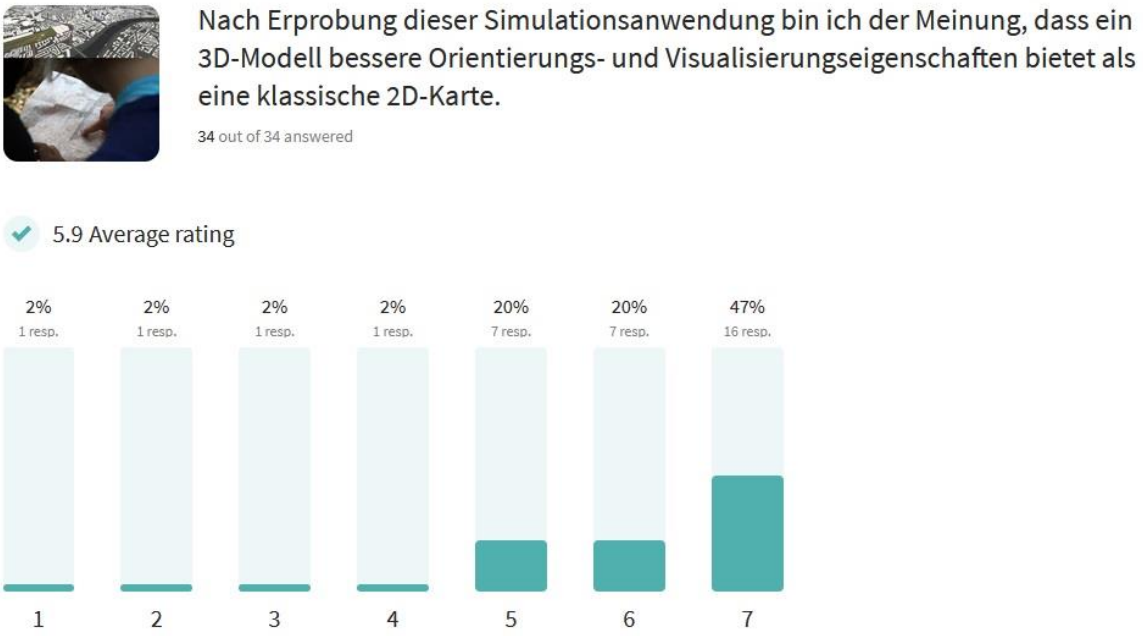


Abb. 27: Balkendiagramm des Evaluationsergebnisses für die 2. Frage, Bildschirmaufnahme der Fragebogenauswertung

Die zweite Frage verdeutlicht durch eine sehr hohe Mehrheitsmeinung die Hypothese, dass ein 3D-Modell bessere Orientierungs- und Visualisierungseigenschaften besitzt als eine herkömmliche 2D-Karte. Hierbei bezog sich die Aussage insbesondere auf das zu evaluierende Projekt. Dementsprechend wurde die Sinnhaftigkeit des Projekts bezogen auf die Entwicklung einer Orientierungsanwendung durch die Nutzer bestätigt. Die Integration der dritten Dimension innerhalb derartiger Simulationsanwendungen bietet dem Nutzer völlig neue Möglichkeiten und Perspektiven und sollte dementsprechend auch in zukünftigen Anwendungen als Standard gelten.

5.3.3 Vergleich zwischen interaktiven und statischen Karten

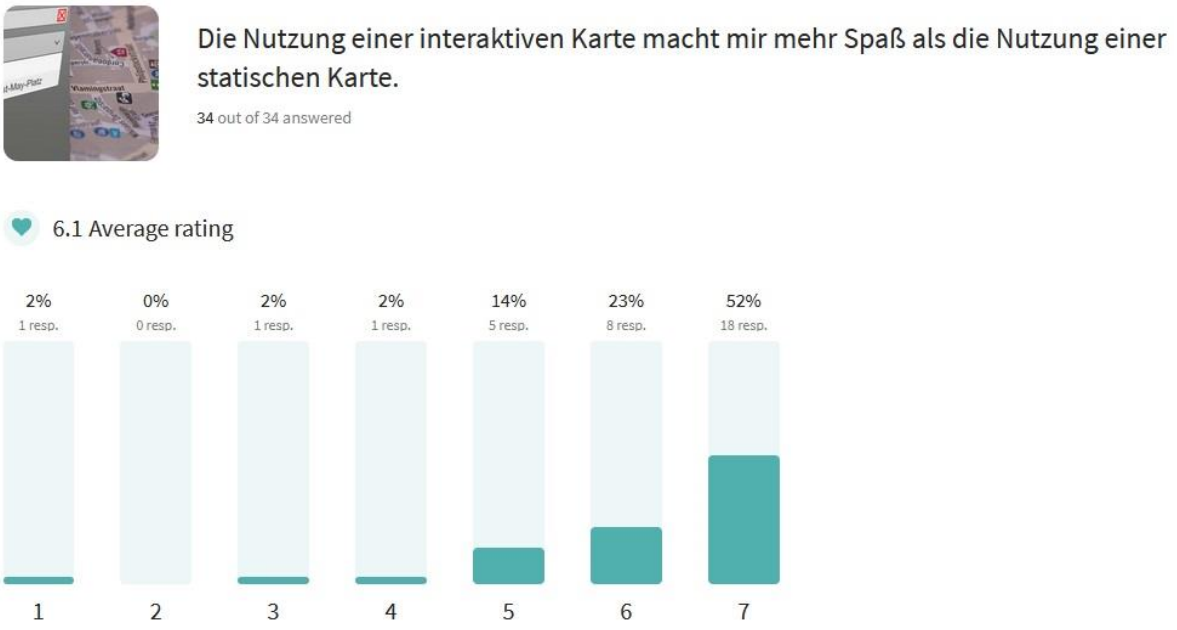


Abb. 28: Balkendiagramm des Evaluationsergebnisses für die 3. Frage, Bildschirmaufnahme der Fragebogenauswertung

In Frage drei wird eine deutliche Aussage seitens der Nutzer vertreten. Die Implementierung von funktionalen Eigenschaften und der dadurch verbundenen Interaktivität der Karte führt letztendlich zu einer besseren Benutzererfahrung. Grund dafür könnte die intelligente Informationsallokation innerhalb solcher interaktiven Karten sein. Da statische Karten keine Interaktionsmöglichkeiten besitzen, müssen alle Informationen auf einen Schlag angezeigt werden, was im Endeffekt zu einer deutlich geringeren Übersichtlichkeit und einer schlechten Benutzererfahrung führt. Interaktive Karten bieten diverse Lösungsansätze für dieses Problem, wobei innerhalb der entwickelten Anwendung das Prinzip der Informationsabstraktion verfolgt wurde. Dabei werden dem Nutzer nur die Informationen angezeigt, welche durch ihn ausgewählt wurden. Diese werden zugleich durch ein optisch ansprechendes 3D-Modell visualisiert und ermöglichen eine lebhafte und realitätsgetreue Abbildung der physikalischen Welt.

5.3.4 Bewertung der erfüllten Anwendungsanforderungen

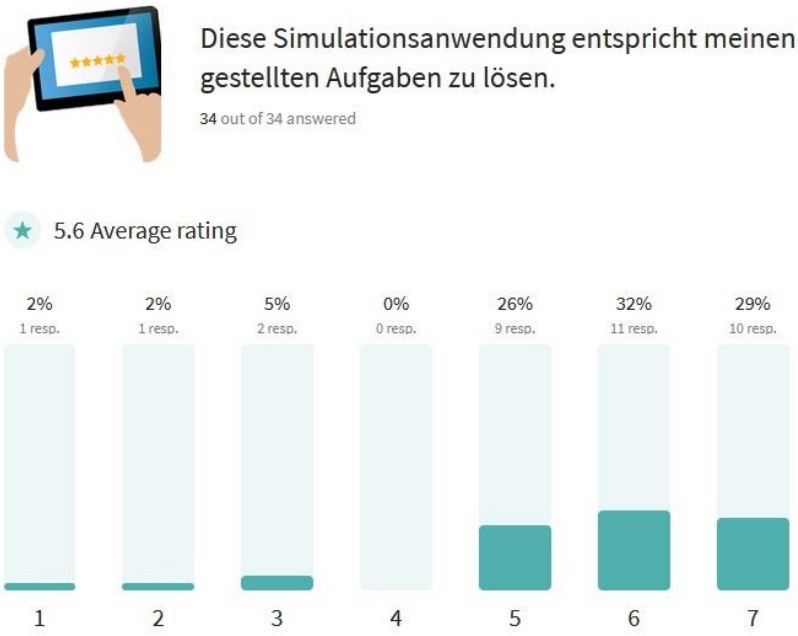


Abb. 29: Balkendiagramm des Evaluationsergebnisses für die 4. Frage, Bildschirmaufnahme der Fragebogenauswertung

Die Bewertung der vierten Frage verlief größtenteils positiv, jedoch konnte festgestellt werden, dass einige Nutzer Probleme mit der Durchführung der gestellten Aufgaben hatten. Der Grund dafür liegt höchstwahrscheinlich im Ergebnis der ersten Frage, der Steuerungskomplexität. Da die Steuerung ein zentrales Element der Anwendung darstellt und die Beherrschung dieser unabdingbar für die Durchführung der Aufgaben ist, kann angenommen werden, dass diese die Ursache für die Komplikationen war. Des Weiteren könnte die Aufgabe 4 einigen Nutzern Probleme bereitet haben, da hier (beabsichtigt) nur eine ÖPNV-Linie genannt wurde und der Nutzer nach der entsprechenden Station suchen musste. Dementsprechend könnte die Anwendung so weiterentwickelt werden, dass die Suche nach ÖPNV-Linien ebenfalls im Suchfenster unterstützt wird (derzeitig wird nur die Suche nach Stationen unterstützt).

5.3.5 OpenStreetMap als Datenquelle für Anwendungen

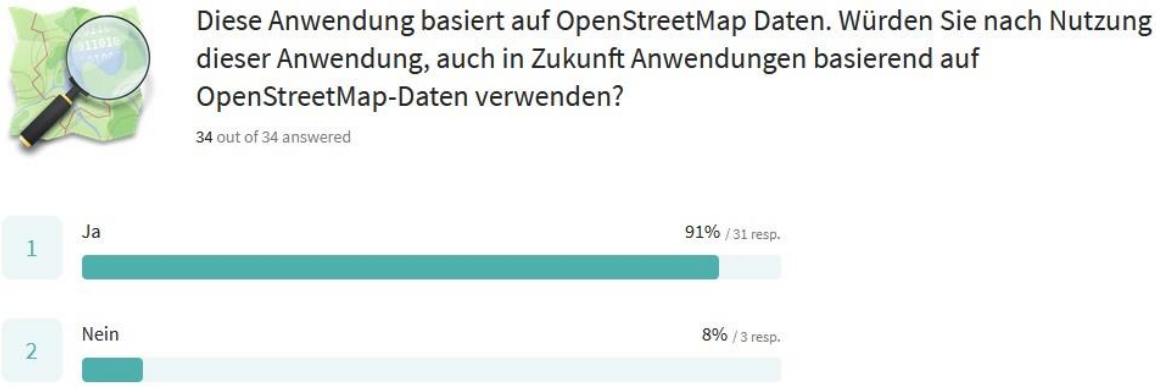


Abb. 30: Balkendiagramm des Evaluationsergebnisses für die 5. Frage, Bildschirmaufnahme der Fragebogenauswertung

Das Ergebnis der Frage fünf zeigt die deutliche Meinung der Nutzer zu OpenStreetMap. OpenStreetMap kann als seriöse und qualitative Quelle für Geodaten angesehen werden, trotz der Crowdsourcing Eigenschaften. Die Entwicklung von OSM-basierten Anwendung ermöglicht es, das Konzept von OSM immer weiter zu verbreiten und die Vorteile derartiger Projekte zu veranschaulichen. Diese könnten für viele Nutzer zugleich eine weitere Motivation sein, sich selbst an dem OpenStreetMap Projekt zu beteiligen. Die zentrale Pflege dieser Daten fördert die Entwicklung von neuen interessanten Anwendungen, wodurch ein Mehrwert für die gesamte Gesellschaft geschaffen wird. Dabei muss immer bedacht werden, dass die Anwendungen nur so gut sind wie die Daten, aus denen sie stammen.

5.3.6 Erweiterungsvorschläge der Nutzer

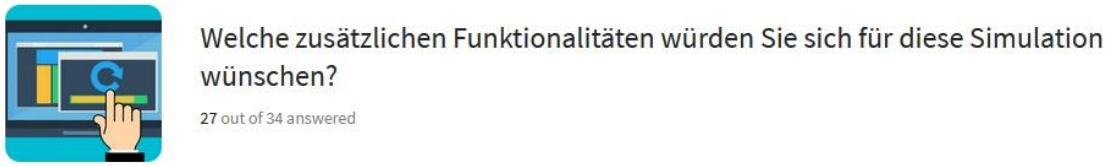


Abb. 31: Bildschirmaufnahme der 6. Fragestellung aus dem Fragebogen

Ab Frage sechs konnten sich die Nutzer mit eigenen Worten äußern. Dabei wurden einige interessante Ideen hervorgebracht. Die Erweiterungsvorschläge aus diesem Abschnitt wurden zusammenfassend in Kapitel 6.1.1 behandelt.

5.3.7 Nutzungspotenziale der Simulationsanwendung

Nennen Sie mindestens 3 Nutzungspotenziale dieser Simulationsanwendung.

27 out of 34 answered

Abb. 32: Bildschirmaufnahme der 7. Fragestellung aus dem Fragebogen

Die siebte Frage befasste sich mit der Identifikation der Nutzungspotenziale und in diesem Zusammenhang auch mit den Nutzergruppen der Simulationsanwendung. Der Inhalt dieses Abschnitts wird in Kapitel 6.1.2 behandelt.

5.3.8 Vorteile der entwickelten Anwendung

Welche Vorteile sehen Sie in dieser Anwendung gegenüber gängigen Anwendung (z.B. Google Maps)?

28 out of 34 answered

Abb. 33: Bildschirmaufnahme der 8. Fragestellung aus dem Fragebogen

Die achte und letzte Frage befasste sich mit dem Vergleich der entwickelten Simulationsanwendung gegenüber gängigen Anwendungen wie z.B. Google Maps. Dabei sollten einige Vorteile durch die Nutzer hervorgebracht werden, an denen sich die zukünftige Entwicklung weiterorientieren könnte. Ein großer Anteil der Nutzer hat die 3D-Komponente offen bevorzugt gegenüber der klassischen 2D-Komponente anderer Anwendungen. Als Grund wurde angegeben, dass sich der dreidimensionale Maßstab sehr gut zur Orientierung und Visualisierung der Orte eignet, was wiederum zu einem besseren Verständnis der ÖPNV-Infrastruktur führt. Fragen wie z.B. was es für alternative Strecken gibt oder was es um die Haltestelle herum noch gibt, können leichter beantwortet werden. Ein weiterer großer Vorteil ist die globale Unterstützung des ÖPNVs innerhalb der Anwendung. Da die Daten aus OSM stammen, können auch kleiner Orte sehr gut abgebildet und visualisiert werden, was beispielsweise bei Anwendungen wie Google Maps nicht mehr unterstützt wird. Der integrierte Flugmodus innerhalb des Modells erlaubt es den Nutzern sich frei zu bewegen und verschiedene Winkel und Perspektiven einzunehmen, welche in der Form bisher von keiner anderen Anwendung unterstützt werden. Des Weiteren verfügt das Modell über animierte Bus- und Zugmodelle, welche ebenfalls in der Form von bisher keiner anderen Anwendung umgesetzt werden. Ein weiteres wichtiges Thema war für viele Nutzer der Datenschutz, der in diesem Zusammenhang genannt wurde. Da die entwickelte Anwendung keinerlei personenbezogene Daten erhebt, ist der Datenschutz im vollen Umfang gewährleistet.

5.4 Zusammenfassung der Evaluationsergebnisse

Die Evaluation hat insgesamt sehr interessante Einblicke in das Nutzerverhalten und die Nutzeranforderungen ermöglicht. Dabei wurden Hypothesen über die Vorteile von interaktiven 3D-Karten überprüft und die Gründe dieser Vorteile näher beleuchtet. Es wurden interessante Nutzervorschläge gesammelt, welche durch ihre Interaktion die Weiterentwicklung der Anwendung beeinflussen. Dies bedeutet, dass das Konzepte wie beispielsweise die Steuerung überdenkt werden müssen, um eine möglichst breite Nutzerzufriedenheit zu gewährleisten. Die Teilnehmer haben die interaktive 3D-Ausführung offen bevorzugt gegenüber herkömmlichen Karten-Ausführungen. Des Weiteren wurde ein breites Interesse an der Weiterentwicklung und Integration von neuen und komplexeren Funktionalitäten festgestellt.

Zusammenfassend kann gesagt werden, dass die Anwendung den Benutzeranforderungen entspricht und durch die modulare Aufbauweise viel Raum für Erweiterungsmöglichkeiten bietet. Durch die Befragung konnte festgestellt werden, dass die Anwendung bereits zum jetzigen Zeitpunkt einige Interessante Vorteile aufweist und durch entsprechende Weiterentwicklung durchaus in näherer Zukunft mit marktführenden Anwendungen konkurrieren könnte.

Kapitel 6
Zusammenfassung und Ausblick

6.1 Perspektiven der Simulationsanwendung

Die zentrale Idee und Aufgabe dieses Projekts war es, eine Anwendung zu entwickeln, welche durch die automatisierte Verarbeitung von OSM-Daten eine dynamische Simulation des ÖPNVs für jeden in OSM verfügbaren Ort erzeugt und dadurch als Basisinfrastruktur für die Entwicklung von weiteren, komplexeren Simulationsanwendungen dienen und einen Mehrwert für diverse Nutzergruppen schaffen kann. Da sich die Software noch immer in einer sehr frühen Entwicklungsphase befindet, stehen zahlreiche Verbesserungs- und Weiterentwicklungsmöglichkeiten zur Verfügung. Einige von diesen wurden anhand der Evaluation abgeleitet und werden in den Unterkapiteln 6.1.1 und 6.1.2 behandelt..

6.1.1 Verbesserungsvorschläge der aktuellen Anwendung

Da die Entwicklung der Anwendung in einem kurzen Zeitfenster von nur 6 Monaten stattfand, konnten einige Inhalte nicht zu dem gewünschten Detaillierungsgrad umgesetzt werden. Einige von diesen Inhalten sind den Nutzern aufgefallen und wurden dementsprechend als Verbesserungsvorschläge innerhalb der Evaluation aufgeführt. Beispielsweise sollte es verschiedene Stationsschilder für Busse und Bahnen geben, um zwischen den Verkehrsmitteln zu differenzieren. Des Weiteren sollte sich das Stations-UI in Blickrichtung des Nutzers drehen (derzeit ist es statisch in eine feste Richtung ausgerichtet). Ein weiterer Vorschlag war, die Stationsnamen schon vorher anzuzeigen, beispielsweise auf den Stationsschildern. Diese könnten ähnlich zu *Google Streetview* einige Fotos von der Stationsumgebung, den Gebäuden und Sehenswürdigkeiten enthalten, um dadurch den Nutzern eine noch höhere Orientierungsmöglichkeit zu bieten. Dementsprechend wurde die Unterstützung der Anwendung auf Smartphones durch zahlreiche Nutzer angefragt, um diese auch mobil nutzen zu können. Darüber hinaus sollten Abfahrtzeiten der ÖPNV-Linien in die Anwendung fest integriert werden, um das Programm als ÖPNV-Informations- und Kartenanwendung abzurunden. Grundfunktionen wie das Auslesen des aktuellen Standortes oder Suchfunktionen für den ÖPNV von Punkt A nach Punkt B sollten in diesem Zusammenhang ebenfalls fester Bestandteil der Anwendung werden. Da die Anwendung auch großes Simulationspotential aufweist, bieten sich hier ebenfalls zahlreiche Erweiterungsmöglichkeiten an. Ergänzend zum ÖPNV-Verkehr könnte der allgemeine Autoverkehr in die Simulation integriert werden, um dadurch das gegenseitige Zusammenwirken zu simulieren. Zahlreiche Nutzer haben sich das Besteigen des Zuges und eine Perspektive als Passagier oder Berufskraftfahrer gewünscht, wobei ebenfalls Menschenmodelle innerhalb der Simulation vorhanden wären.

Eine derartige Anwendung könnte interessante Ergebnisse in einer VR-Umgebung erzielen. Dabei könnten weitere Aspekte wie Wetterbedingungen, Texturen und andere Komponenten integriert werden, um eine gute und realitätsgetreue Simulation zu bilden. Ebenfalls interessant wäre die Implementierung der Erstellung von benutzerdefinierten Fahrplänen, da diese insbesondere von den Verkehrsverbünden genutzt werden könnten.

6.1.2 Nutzergruppen mit Mehrwert und allgemeine Erweiterungsmöglichkeiten

Eine zentrale Nutzergruppe dieser Anwendung sind Privatpersonen bzw. Touristen im Alltagsablauf. Diese können schnell an gewünschte ÖPNV-Informationen herankommen, Anbindungen besser erkennen und sich im Allgemeinen besser orientieren. In diesem Zusammenhang wäre eine sehr nützliche Funktion, reelle Abfahrtzeiten der einzelnen Zug- und Buslinien in das Programm zu integrieren und somit eine flächendeckende Anwendung zur Fahrplanauskunft zu schaffen. Dementsprechend sollte diese Anwendung auf mobilen Endgeräten unterstützt werden, um auch mobil einen Mehrwert zu bieten.

Eine weitere Nutzergruppe wären die ÖPNV-Berufskraftfahrer bzw. deren Arbeitgeber und Ausbilder. Diese könnten (vor allem bei Buslinien) sich die entsprechende Strecke aneignen und leichter Ausweichrouten bestimmen. Zugleich bietet diese Simulation Potential zur Weiterentwicklung einer Fahrerkomponente, bei der der Nutzer die Position des Berufskraftfahrers einnehmen könnte und eine realitätsnahe Simulation des Berufsalltags bekäme.

Spiele- und Anwendungsentwickler können durch den Algorithmus dieser Anwendung einen Einblick in die Vorgehensweise der Verarbeitung von OSM-Daten und der Erzeugung von 3D-Objekten bekommen. Dies könnte dazu führen, dass zeitintensive, manuelle Arbeit vermieden wird und zahlreiche neue Anwendungsbeispiele (abseits des ÖPNVs) entstünden.

Den wohl größten wirtschaftlichen Nutzen könnten Verkehrsverbünde ziehen. Diese könnten unter anderem durch die Weiterentwicklung von entsprechenden Funktionen den Ausbau von neuen Strecken simulieren, die Auslastung der Schieneninfrastruktur überprüfen, diverse Eskalationsfälle abfragen und vorsimulieren, den allgemeinen Betrieb optimieren oder auch Betriebsänderungen vorsimulieren, den Verkehr überwachen usw. Dies könnte letztendlich zu besseren Investitionen und Unternehmensentscheidungen führen, durch welche die komplette Gesellschaft profitieren könnte.

Zu guter Letzt sollte ebenfalls die OSM-Community als Nutzergruppe erwähnt werden, da durch die Entwicklung von OSM-basierten Anwendungen das Konzept des OpenStreetProjekts weiterlebt und an andere Personen vermittelt wird. Es fördert das allgemeine Bewusstsein über die Potentiale solcher Crowdsourcing-Projekte und führt dazu, dass sich weitere Personen am Projekt beteiligen.

6.2 Fazit

Zusammenfassend kann gesagt werden, dass die in der Einleitung formulierten Anforderungen zur Entwicklung einer robusten Simulationsanwendung erfüllt wurden. Es hat sich erwiesen, dass das Anwendungsdesign durch eine Mehrheit der Anwender als benutzerfreundlich empfunden wurde und anhand einer abstrahierenden Darstellung der Szene eine optimale Übersichtlichkeit den Nutzern geboten wird. Dies in Verbindung mit der Bewegungsfreiheit innerhalb des Modells ermöglicht den Anwendern ein hohes Maß an Orientierung und bietet verschiedene Interaktionsmöglichkeiten zwischen dem Nutzer und dem Modell. Durch die Auswahl von OSM als zentrale Datenquelle wird durch die Anwendung ein breites Angebot an unterstützten Orten ermöglicht, was dazu führt, dass die Anwendung flächendeckend (sogar global) einsetzbar ist, im Gegensatz zu anderen verwandten Werken. Des Weiteren hat sich die Datenqualität von OSM, vor allem im Bezug zum ÖPNV, als sehr hoch erwiesen und ermöglicht dadurch eine akkurate Wiederspiegelung der Wirklichkeit. Ein weiterer grundlegender Vorteil besteht auch darin, dass durch die Pflege der OSM-Daten (durch die OSM-Community), die zeitlich bedingten Veränderungen ohne Zutun des Entwicklers ins Modell übernommen werden, wodurch kein manueller Aktualisierungs- oder Pflegebedarf bei den Entwicklern besteht. Dementsprechend hat sich OSM als kostenfreie Datenquelle für Geoinformationen als voller Erfolg erwiesen und es wurde durch diese Simulationsanwendung, neben anderen OSM-Anwendungen, nochmals auf die Bedeutung von derartigen Crowdsouring-Projekten aufmerksam gemacht. Durch dieses Projekt konnte dem Leser ein Einblick in die Verarbeitung von OSM-Daten mit dem Ziel der Erzeugung von entsprechenden 3D-Modellen verschafft werden. Dieser Prozess hat sich durch die kurzen Ladezeiten und die Unterstützung von größeren OSM-Dateien als sehr effizient erwiesen und bietet dementsprechend viel Raum für Auslastung. Des Weiteren konnten durch die integrierten Verkehrsanimationen weitere Ausbaumöglichkeiten angedeutet und das Simulationspotential der Anwendung veranschaulicht werden. Die Evaluation der Anwendung ermöglichte den Nutzern unter anderem einen Vergleich von 3D-Stadtmodellen gegenüber klassischen 2D-Karten hinsichtlich der Orientierungs- und Visualisierungseigenschaften. Dabei konnte festgestellt werden, dass ein Großteil der Nutzer ein 3D-Modell intuitiever und orientierungsfreundlicher empfand. Der gleiche Trend konnte bei dem Vergleich zwischen interaktiven und statischen Karten festgestellt werden. Alle diese Aspekte deuten darauf hin, dass die entwickelte Anwendung durch ihre zahlreichen Vorteile sehr viel Potential birgt und durch entsprechende Weiterentwicklung gewisser Anwendungskomponenten durchaus mit anderen (kommerziellen) Anwendungen konkurrieren kann. Diese Erkenntnisse ist zugleich sehr erstaunlich, wenn man bedenkt, dass für die Entwicklung der Anwendung keinerlei finanzeller Ressourcen zur Verfügung standen und komplett auf lizenz- und kostenfreie Inhalte zurückgegriffen musste. Dementsprechend stellt sich auch die Diskussionsfrage, ob nicht große Unternehmen ihre Ressourcen zur Verfügung stellen sollten (trotz kommerzieller Potentiale), um dadurch die Entwicklung von unabhängigen Projekten und Anwendungen

zu fördern, welche im Endeffekt einen Mehrwert für diese Unternehmen und die ganze Gesellschaft schaffen könnten.

Kapitel 7
Literaturverzeichnis

[1] Apple Inc. "*iOS and iPadOS Feature Availability*". https://www.apple.com/ios/feature-availability/. Zugriff: 16. März 2020

[2] Biljecki F., Ledoux H., Stoter J. (2016). "*An improved LOD specification for 3D building models*". Computers, Environment, and Urban Systems, vol. 59, pp. 25-37.

[3] Brenner C., Haala N. & Fritsch D. (2001). "*Towards fully automated 3D city model generation*". Am Institut für Photogrammetrie der Universität Stuttgart.

[4] Dennis N. "*Google Earth's incredible 3D imagery, explained*". https://blog.google/products/earth/google-earths-incredible-3d-imagery-explained/. Zugriff: 14. März 2020

[5] Deutsche Bahn AG. "*Kooperation zwischen Google und Deutscher Bahn: Google Transit startet in Deutschland*". https://www.youtube.com/watch?v=0sriDHfZGok. Zugriff: 14. März 2020

[6] Duality Beratungs-GmbH. "*Besserer ÖPNV dank Digitalisierung? Ein Gewinn für alle Beteiligten!*". https://www.axians.de/de/blog/2018/11/29/besserer-oepnv-dank-digitalisierung/. Zugriff: 14. März 2020

[7] Duality Beratungs-GmbH. "*Duality Hololens 3D Visualisierung ÖPNV | Digitaler Leitstand*". https://www.youtube.com/watch?v=2Xk1ntQHtlo. Zugriff: 14. März 2020

[8] F4 Group. "*F4map connecting people with maps*". https://www.f4map.com/. Zugriff: 12. März 2020

[9] F4 Group. "*F4map Demo*". https://demo.f4map.com/. Zugriff: 12. März 2020

[10] Finstad K. (2010). "*The usability metric for user experience*". In: Interacting with Computers 22.5, S. 323-327

[11] FOSSGIS Webseite. "*ÖPNV bei OpenStreetMap. Warum sich der Verkehrsverbund Rhein-Sieg (VRS) und der Aachener Verkehrsverbund (AVV) bei OSM engagieren*". https://fossgiskonferenz.de/2011/programm/attachments/271_20110406_Fossgis2.pdf. Zugriff: 24. März 2020

[12] Google LLC. "*Google Earth*". https://www.google.com/intl/de/earth/. Zugriff: 14. März 2020

[13] Google LLC. "*Google Maps*". https://www.google.com/maps. Zugriff: 14. März 2020

[14] Götz M. (2012). "*OpenStreetMap – Datenqualität und Nutzungspotenzial für Gebäudebestandsanalysen*". Flächennutzungsmonitoring IV. Genauere Daten – informierte Akteure – praktisches Handeln. IÖR Schriften Band 60 · 2012.

[15] Gunderson P. "*FlyCamera.cs*". https://gist.github.com/gunderson/d7f096bd07874f31671306318019d996. Zugriff: 21. April 2020
[16] Haklay, Mordechai. "*How good is volunteered geographic information. A comparative study of OpenStreetMap and Ordnance Survey datasets*". Environment and Planning B: Planning and Design, 2010, 683-703.
[17] Hermann S. (2019). "*Prozedurale Generierung von 3D-Stadtmodellen*". Bachelorarbeit am Institut für Computervisualistik der Universität Koblenz-Landau.
[18] Ibessard. Turbosquid Webseite. "*Tramway.sit*". https://www.turbosquid.com/3d-models/free-3ds-model-tramway/221138. Zugriff: 17. Mai 2020
[19] Kada M. (2007). "*Zur maßstabsabhängigen Erzeugung von 3D-Stadtmodellen*". Dissertation bei der Fakultät für Luft- und Raumfahrttechnik und Geodäsie der Universität Stuttgart.
[20] Kelly S. "*Scene Base Real World Map Data*". https://github.com/codehoose/real-world-map-data/tree/master/src/Scene%20Based%20Real%20World%20Map%20Data/Assets/Scripts. Zugriff: 15. April 2020
[21] Koppers L. (2002). "*Generierung von Objekten für 3D Stadtmodelle*". Dissertation bei der Fakultät für Bauingenieur- und Vermessungswesen der Universität der Bundeswehr München.
[22] Kulas O. (2012). "*Prozedurale Generierung von Echtzeit-optimierten 3D-Stadtmodellen*". Bachelorarbeit durchgeführt bei weltenbauer GmbH an der Technischen Hochschule Mittelhessen.
[23] Mapbox. "*Built with Mapbox*". https://www.mapbox.com/showcase/. Zugriff: 24. März 2020
[24] Mapbox. "*Data Sources*". https://www.mapbox.com/about/maps. Zugriff: 24. März 2020
[25] Mapbox. "*How Mapbox works*". https://docs.mapbox.com/help/how-mapbox-works/. Zugriff: 24. März 2020
[26] Mapbox. "*Mapbox pricing*". https://www.mapbox.com/pricing/. Zugriff: 24. März 2020
[27] Mapbox. "*Maps for Unity*". https://www.mapbox.com/unity/. Zugriff: 24. März 2020
[28] Mapbox. "*Unity applications*". https://docs.mapbox.com/help/how-mapbox-works/unity/. Zugriff: 24. März 2020
[29] Marsch J. "*OSM Buildings*". https://osmbuildings.org/. Zugriff: 12. März 2020
[30] Mehler A., Abrami G., Bruendel S., Felder L., Ostertag T., Spiekermann C. "Stolperwege: An App for a Digital Public History of the Holocaust," in Proceedings of the 28th ACM Conference on Hypertext and Social Media, New York, NY, USA, 2017, pp. 319-320

[31] Mehler A., Abrami G., Spikermann C., Jostock M. “VannotatoR: A Framework for Generating Multimodal Hypertexts“, Proceedings of the 29th ACM Conference on Hypertext and Social Media, New York, NY, USA, 2018
[32] Moos M. “*ÖPNV Karte*”. https://www.öpnvkarte.de. Zugriff: 25. März 2020
[33] Ohorodnichuk M. Turosquid Webseite. “*3D model Maglev train low-poly 3D model*”. https://www.turbosquid.com/3d-models/3d-model-maglev-train-low-poly-1431858. Zugriff: 18. Mai 2020
[34] OpenStreetMap Webseite. “*FAQs*”. https://www.openstreetmap.de/faq.html#was_ist_osm. Zugriff: 19. März 2020
[35] OSM Wiki. “*3D*”. https://wiki.openstreetmap.org/wiki/3D. Zugriff: 12. März 2020
[36] OSM Wiki. “*Map Features*”. https://wiki.openstreetmap.org/wiki/Map_Features. Zugriff: 19. April 2020
[37] OSM Wiki. “*Mercator*”. https://wiki.openstreetmap.org/wiki/Mercator. Zugriff: 19. April 2020
[38] OSM Wiki. “*OSM XML*”. https://wiki.openstreetmap.org/wiki/OSM_XML. Zugriff: 27. März 2020
[39] OSM Wiki. “*Rhein-Main-Verkehrsverbund*”. https://wiki.openstreetmap.org/wiki/Rhein-Main-Verkehrsverbund#Frankfurt. Zugriff: 21. März 2020
[40] OSM Wiki. “*Stats*”. https://wiki.openstreetmap.org/wiki/Stats. Zugriff: 24. März 2020
[41] ÖPNV Simulationen. https://www.youtube.com/channel/UCO1BuqVzRmpsgZt8Tfxhipg. Zugriff: 29. April 2020
[42] ÖPNV Simulationen. https://www.youtube.com/watch?v=etmjra_CXOc. Zugriff: 16. Mai 2020
[43] Parrish R. “*How Apple Creates 3D Flyover Maps*”. https://www.applegazette.com/apple-inc/how-apple-creates-3d-flyover-maps/. Zugriff: 16. März 2020
[44] Pixabay GmbH. “*Pixabay*”. https://pixabay.com/. Zugriff: 25. April 2020
[45] Prochitecture. “*Blender-osm: OpenStreetMap and terrain for Blender*”. https://github.com/vvoovv/blender-osm. Zugriff: 12. März 2020
[46] Ramthun R. (2012). “*Offene Geodaten durch OpenStreetMap*”. In U. Herb (Hrsg.), Open Initiatives: Offenheit in der digitalen Welt und Wissenschaft (S. 159-184). Saarbrücken: universaar.
[47] Roick O., Neis P., Zipf A. (2011). “*Volunteered Geographic Information – Datenqualität und Nutzungspotentiale am Beispiel von OpenStreetMap*”.
[48] Spikermann C., Abrami G., Mehler A. “VannotatoR: A Gesture-driven Annotation Framework for Linguistic and Multimodal Annotation”, Proceedings of the Annotation, Recognition and Evaluation of Actions (AREA 2018) Workshop, Miyazaki, Japan, 2018

[49] TextTechnologyLab. Github-Seite der Evaluation. https://github.com/texttechnologylab/PublicTransportSimulator. Zugriff: 16. Mai 2020
[50] Typeform. https://www.typeform.com/. Zugriff: 16. Mai 2020
[51] Wikipedia-Artikel über *Apple Maps*. https://en.wikipedia.org/wiki/Apple_Maps. Zugriff: 16. März 2020
[52] Wikipedia-Artikel über *Google Maps*. https://de.wikipedia.org/wiki/Google_Maps. Zugriff: 14. März 2020
[53] Wikipedia-Artikel über *XML*. https://de.wikipedia.org/wiki/Extensible_Markup_Language. Zugriff: 27. März 2020
[54] Zielstra D., Zipf A. "*A Comparative Study of Proprietary Geodata and Volunteered Geographic Information for Germany*". The 13th AGILE International Conference on Geographic Information Science. Guimaraes, Portugal, 2010.