Universität Stuttgart

Institut für Formale Methoden der Informatik

Abteilung Algorithmik

Universitätsstraße 38
70569 Stuttgart

Bachelorarbeit

Generierung von Minecraft Welten aus OpenStreetMap-Daten

Friedrich Dürr

Studiengang: Informatik

Prüfer: Prof. Dr. Stefan Funke

Betreuer: Prof. Dr. Stefan Funke

begonnen am: 15.08.2023

beendet am: 15.02.2024

Kurz-Zusammenfassung

Diese Arbeit erforscht die Konvertierung von OpenStreetMap-Daten in spielbare Minecraft-Welten. Durch die Nutzung von Höhendaten und Daten aus OpenStreetMap (OSM) wird eine nahtlose Integration von realen Straßen, Gebäuden und Geländemerkmalen in Minecraft ermöglicht. Der Fokus liegt auf der Automatisierung des Prozesses, um vom Nutzer ausgewählte OSM-Datensätze effizient in Minecraft zu übertragen. Verschiedene Herausforderungen wie Datenaufbereitung, Effizienz und Detailgenauigkeit werden untersucht, um realistische Minecraft-Welten zu schaffen. Die resultierenden Welten dienen nicht nur als unterhaltsame virtuelle Umgebungen, sondern auch als nützliche Schablone, um die echte Welt in Minecraft nachzubauen und als Werkzeug für städtische Planung und Geoinformatik-Anwendung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1

1 Überblick

In diesem Kapitel soll ein kleiner Überblick über die Arbeit gegeben werden.

1.1 Einleitung

In einer zunehmend digitalen Welt gewinnt die Visualisierung von Daten an Bedeutung, um unsere Umgebung besser zu verstehen. Ein entscheidender Aspekt dabei ist die effektive Aufbereitung der Daten für einen umfassenden Überblick. OpenStreetMap (OSM)[2] nimmt als führende Open-Source-Geodatenplattform durch kollaborative Anpassungsmöglichkeiten eine Spitzenposition ein. Das bedeutet, dass jeder die Daten ändern oder hinzufügen kann. Minecraft [3] eignet sich durch seine einfache Darstellung in Blöcken, um ein Abbild der Realität zu schaffen. Mit seinen weitläufigen Welten und unbegrenzten Gestaltungsmöglichkeiten fasziniert Minecraft Millionen von Spielern.

Dafür entwickeln wir ein Programm, das mithilfe von OSM-Daten eine Minecraft Welt generiert.

Mit Hilfe dieser Anwendungen lässt sich ein vertieftes Verständnis für Landschaft und Architektur gewinnen. Während reale Landschaftsproportionen oft aufgrund der begrenzten Perspektive im Alltag übersehen werden, ermöglicht der kreative Modus von Minecraft eine vielfältige Betrachtung aus unkonventionellen Blickwinkeln. Im Kreativmodus hat man unendlich viele „Items“ (Materialien) und man kann fliegen.

Die Umsetzung einer realen Welt in die Minecraft Welt kann bei der Planung von Infrastrukturen wie Straßen, Bahnlinien und Städten unterstützen. Oder wie sich bestimmte Architekturen von Häusern harmonisch in ein Wohngebiet integrieren lassen.

Da die Umwandlung zwischen den stetig, gekrümmten Geometrien der realen Welt und den diskreten Blöcken von Minecraft komplex ist, werden in dieser Arbeit zunächst die Grundlagen beschrieben. Der Unterschied zwischen beiden Topologien beeinflusst die virtuelle Repräsentation und das Aussehen in Spielen wie Minecraft.

Die Herausforderungen und Lösungen bei dieser Aufgabenstellung werden diskutiert, und die Arbeit schließt mit einer Zusammenfassung sowie einem Ausblick für zukünftige Entwicklungen ab. Das Ziel besteht darin, eine Website bereitzustellen, auf der Nutzer einen Abschnitt in OpenStreetMap auswählen können, um diesen in eine Minecraft Welt zu transformieren.

In dieser Arbeit werden deutsche und ebenso englische Begriffe verwendet, weil Programme konventionell in Englisch geschrieben werden. Beispielsweise entspricht der Breitengrad der Latitude und der Längengrad der Longitude. Doch die Bedeutung bleibt stets dieselbe. Für Formulierungsvorschläge wurde ChatGPT genutzt. Die vorgeschlagenen Texte wurden vor ihrer Verwendung inhaltlich und sprachlich geprüft und verbessert.

1.2 Verwandte Arbeiten

Wir waren nicht die ersten, die die Realität in Minecraft abbilden wollten. Eine Reihe von Projekten haben sich bereits mit der Konvertierung von OpenStreetMap (OSM) Daten in Minecraft Welten beschäftigt. Zwei solcher Projekte, die besonders erwähnenswert sind, sind OSM2Map und Terra 1-to-1 ([4] und [5]).

In diesem Kapitel werden wir einen Überblick über diese beiden Projekte geben und ihre Funktionsweise sowie ihre Vor- und Nachteile diskutieren.

1.2.1 Osm2Map

Die Idee, Weltdaten wie die von OpenStreetMap in Minecraft zu integrieren, wird seit langem verfolgt und hat zu verschiedenen Projekten geführt. Ein Beispiel ist das Plugin Osm2Map [4].

Im Unterschied zu unserer Software, die im Zusammenhang mit dieser Arbeit entwickelt wurde, handelt es sich bei Osm2Map um ein Plugin, das auf einem Minecraft Server installiert werden kann. Die Anwendung ändert den Servercode und beeinflusst, wie die Welt in Minecraft generiert wird.

Das Plugin bietet eine Vielzahl von Funktionen, darunter einen benutzerdefinierten Weltgenerator basierend auf OpenStreetMap-Daten, der Gebäude, Gärten, Wasser, Bäume und Bodenfarben unter Verwendung von Satellitenbildern integriert. Des Weiteren ermöglicht das Plugin die Teleportation zu Wegen oder Points of Interest (POIs) anhand ihrer Namen. Eine unterstützte Multi-Weltumgebung ermöglicht individuelle Konfigurationen für jede Welt. Das eingebaute Berechtigungssystem erlaubt das Beanspruchen oder Freigeben von Bereichen, in denen die Spieler gemeinsam bauen können. Durch die Anpassbarkeit des Plugins können sämtliche Blöcke, die vom Generator verwendet werden, nach den Vorlieben der Nutzer verändert werden.

Wenn man auf einem Server mit dem Osm2Map-Plugin die Welt erkundet, werden auf Anfrage die verschiedenen Teile der Welt generiert. Diese flexible Funktionalität erlaubt es den Nutzern, selbst zu bestimmen, welcher Teil der Welt sichtbar sein soll. Allerdings kann dies dazu führen, dass, auf Grund der Schwierigkeiten der Projektion der realen kugelförmigen Welt in eine flache, Übergänge zwischen generierten Bereichen insbesondere Gebäude, unvollständig dargestellt werden.

Insgesamt ermöglicht das Plugin nicht nur die visuelle Repräsentation von OpenStreetMap-Daten in Minecraft, sondern bietet auch eine Fülle von Funktionen zur Anpassung und Interaktion in der generierten Welt.

In unserem Programm kann der Nutzer in einer Webapplikation einen Bereich auswählen, der dann umgewandelt wird. Dies vereinfacht das Programm, da es die Anzahl möglicher Fehler verringert und macht es am Ende leichter die Laufzeiten zu vergleichen.

1.2.2 Terra 1-to-1

Infolge der Idee, die reale Welt in Minecraft nachzubilden, hat sich das ehrgeizige Projekt „Build the Earth“ entwickelt. Dabei handelt es sich um eine Initiative, die aus zahlreichen Bauteams besteht, die die Welt in verschiedene Regionen aufgeteilt haben, um sie gemeinsam in Minecraft detailgetreu nachzubauen. Jedes Bauteam setzt sich aus mehreren Minecraft-Spielern zusammen, die an den Regionen arbeiten, die ihnen zugewiesen sind [6]. Dabei haben sie erheblich mehr Freiheiten und es stehen ihnen mehr Ressourcen zur Verfügung, um Gebäude und Landschaften mit einer viel höheren Detailgenauigkeit nachzubilden, als es mit dem Programm möglich wäre, das in dieser Arbeit behandelt wird.

Es lohnt sich einen Blick auf den Mod namens „Terra 1-to-1“ zu werfen, den die Mitglieder von „Build The Earth“ für ihr Projekt nutzen. Ein Mod ist eine Modifikation, die in Minecraft eingeführt wird, um das Spielerlebnis zu verändern oder zusätzliche Features hinzuzufügen.

„Terra 1-to-1“ ist ein eigenständiger Mod, der nicht nur für das „Build The Earth“-Projekt, sondern auch außerhalb davon verwendet werden kann. Dieser Mod führt eine neue Weltengenerierung ein und nutzt Datenquellen aus der ganzen Welt, um Gelände, Biome, Bäume und Straßen so akkurat wie möglich darzustellen. Biome sind große Ökosysteme, die sich durch bestimmte Pflanzen- und Tierarten sowie klimatische und geografische Bedingungen auszeichnen, wie beispielsweise Wüsten, Regenwälder, Savannen und Tundren [5].

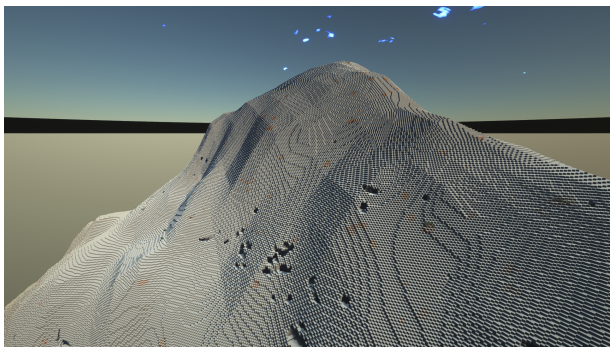

Abb. 1.1: Wie der Mount Everest im Terra 1-to-1 Mod aussieht

Das Hauptziel des Mods besteht darin, die Welt im Maßstab 1:1 darzustellen. Seit September 2022 [7] wird dieser Mod nicht mehr weiterentwickelt und ist daher nur kompatibel mit der Minecraft Version 1.12.2. In dieser Version war die Höhe der Minecraft Welt auf 256 Blöcke begrenzt, weshalb die Entwickler den Mod „CubicChunksMod“ hinzugezogen

haben. Dieser Mod teilt Chunks auch in der Y-Achse auf, um eine Welt zu ermöglichen, die scheinbar endlos in die Höhe und Tiefe geht. Ein Chunk ist eine fest definierte, dreidimensionale Region in Minecraft, die aus Blöcken besteht und eine Größe von 16x16xHöhe hat. Der CubicChunksMod teilt ein Chunk nun also in ein 16x16x16 Bereich auf. Dieser Mod, zusammen mit „Terra 1-to-1“ und anderen Modifikationen, wird als Modpack für die „Build The Earth“-Community angeboten.

Im Gegensatz zu dem Programm, das in dieser Arbeit behandelt wird, nutzt der „Terra 1-to-1“ Mod weitere Quellen, um Höhendaten zu erhalten. Tatsächlich wird auch OpenStreetMap genutzt, um Straßendaten in die generierte Welt zu integrieren. Während des Spielens ermöglicht der Mod den Entwicklern die Verwendung von Befehlen, um sich an beliebige Weltkoordinaten zu teleportieren, ohne sie manuell konvertieren zu müssen.

In der Entwicklungsphase des „Terra 1-to-1“ Mods sind einige Hürden aufgetreten [5]. Besonders hervorzuheben sind die Herausforderungen im Zusammenhang mit der Darstellung von Wasser in der generierten Welt. Diese Schwierigkeiten sind größtenteils auf die Komplexität der realen Erde und die Art und Weise zurückzuführen, wie Höhendaten bereitgestellt werden. Ein Beispiel für die komplexe Aufgabenstellung zeigt sich in Landgebieten, die unterhalb des Meeresspiegels liegen und fälschlicherweise mit Wasser bedeckt werden, als ob sie sich tatsächlich unter Wasser befinden würden. Dies betrifft Regionen wie Teile der Niederlande, die Kaspische See-Region, das Tote Meer und das Imperial Valley. Ein weiteres Beispiel betrifft die unregelmäßige Darstellung von Küstenlinien und das Fehlen von realistischen Stränden in der generierten Welt.

Im Bild 1.1 können Kanten erkannt werden. Diese entstehen durch die lineare Interpolationsmethode, die von diesem Mod verwendet wird. In unserer Arbeit wird eine polynomielle Interpolation verwendet, um diese Kanten zu verhindern.

2 Grundlagen

In diesem Kapitel werden die grundlegenden Konzepte und theoretischen Rahmenbedingungen für die Verbindung von OpenStreetMap (OSM) und Minecraft erläutert. Die Kombination dieser beiden Plattformen eröffnet faszinierende Möglichkeiten im Bereich der Geoinformatik. Wir werden uns eingehend mit den Kernelementen von OSM und den Integrationen in Minecraft beschäftigen, um einen umfassenden Kontext für die anschließende Analyse zu schaffen. Die Synergie zwischen OSM und Minecraft eröffnet neue Perspektiven und Herausforderungen, die in den darauf folgenden Kapiteln eingehend beleuchtet werden.

2.1 OpenStreetMap

OpenStreetMap ist ein zentrales Element unserer Untersuchung. OSM, eine kollaborative Plattform für die Erstellung und Bearbeitung von geografischen Daten, hat in den letzten Jahren erheblich an Bedeutung gewonnen. Die offene Natur dieses Programms ermöglicht es Nutzern weltweit, geografische Informationen zu teilen und zu verbessern. Wir werden die Struktur und Funktionalitäten von OpenStreetMap erkunden, um einen fundierten Überblick über diese Plattform zu bieten. Dies bietet die Grundlage für die Integration von OSM in den Kontext der Weltengenerierung. Zuerst beschäftigen wir uns mit dem Aufbau der Daten und der Zusammensetzung von Koordinaten.

2.1.1 Koordinaten

In der Welt der Geodaten wird die Position auf der Erde üblicherweise in Längen- und Breitengraden gemessen, da die Erde annähernd kugelförmig ist.

Der Längengrad, die Longitude, oft als Lon abgekürzt, gibt an, ob sich ein Punkt östlich oder westlich vom Nullmeridian befindet. Der Nullmeridian wurde auf der Internationalen Meridiankonferenz 1884 festgelegt und verläuft durch die Sternwarte Greenwich in London [8]. In den OSM-Daten wird die Longitude auf den Wertebereich von -180° bis +180° begrenzt. Hierbei stellt -180° den westlichsten Punkt und +180° den östlichsten Punkt dar. Dieser Wertebereich entspricht einer vollen Umdrehung in einem Kreis von 360°. Die Begrenzung erleichtert die Verarbeitung der Daten und entspricht der konventionellen Darstellung auf der Weltkarte.

Der Breitengrad, die Latitude, als Lat abgekürzt, gibt an, wie weit ein Punkt vom Äquator entfernt ist und bestimmt somit, ob sich dieser auf der nördlichen oder südlichen Hemisphäre befindet. Latitudinal werden die Werte in Grad gemessen. Ein Breitengrad entspricht ungefähr 111 Kilometern. Diese Entfernung variiert aufgrund der leicht abgeflachten Form der Erde. Es dient als grobe Schätzung für die Berechnung von Distanzen auf der Erdoberfläche basierend auf Breitengraden [9].

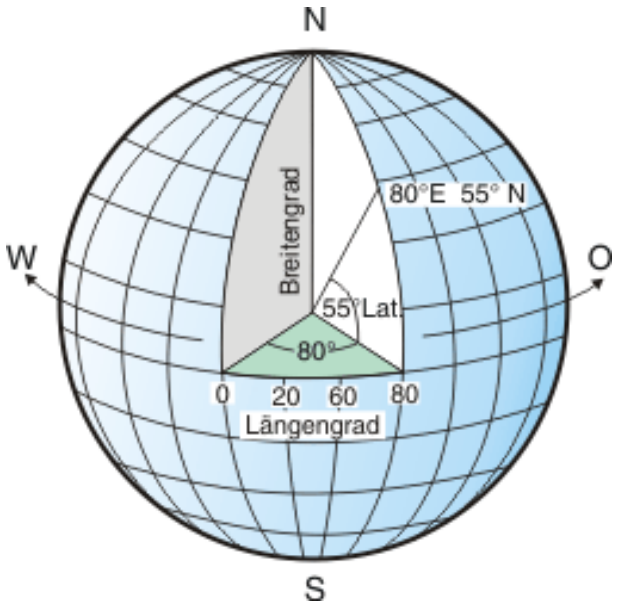

Abb. 2.1: Die Visualisierung einer Koordinate

Die Kombination aus Longitude und Latitude ermöglicht die genaue Darstellung eines Punktes auf der Erde in den OSM-Daten. Auf dem Bild 2.1 sieht man eine Visualisierung von einer Koordinate.

2.1.2 Aufbau einer OSM-Datei

OpenStreetMap hat sich als eine herausragende Plattform etabliert, die es ermöglicht, geografische Daten auf offene und kollaborative Weise zu sammeln, zu verwalten und zu teilen. Als ein zentrales Element der digitalen Weltkarten bietet OSM eine Vielfalt an geografischen Informationen, die von einer globalen Gemeinschaft von freiwilligen Mappern beigesteuert werden.

Die Grundbausteine, mit denen OSM diese reichhaltigen Datensätze aufbaut, sind sogenannte „Nodes“. Nodes sind einzelne Punkte in der geografischen Landschaft und dienen als fundamentale Bausteine, um komplexe Strukturen wie Wege, Umrisse und Gebiete abzubilden. Sie fungieren als grundlegende Bausteine, auf denen die Vielfalt der geografischen Informationen aufbaut. Durch die Verknüpfung von Nodes werden Linien (Wege) und geschlossene Formen (Gebiete) geschaffen, wodurch eine detaillierte und facettenreiche Kartografie entsteht.

Die Hauptaufgabe von OSM liegt nicht nur in den Möglichkeiten, statische Informationen abzubilden, sondern auch die Dynamik, die durch die lebendige Mappergemeinschaft entsteht. Die Daten in OSM sind in ständiger Bewegung, werden aktualisiert und verfeinert, um den sich ändernden Gegebenheiten der realen Welt gerecht zu werden. Diese Flexibilität und Offenheit machen OSM zu einem faszinierenden und lebendigen Instrument für die Darstellung und Verwaltung von geografischen Daten. Jede Woche veröffentlicht OSM ein Update, in dem alle neuen Änderungen der vergangenen Woche zusammengeführt werden [10].

Bevor wir jedoch in die Details der Datenstruktur eintauchen, ist es wichtig, den kollaborativen Geist von OSM zu betonen, der die Grundlage für die Entstehung und Pflege dieser beeindruckenden geografischen Datenbank bildet. OSM ist nicht nur eine Kartierungsplattform, sondern auch ein soziales Netzwerk von Mappern, die ihre Leidenschaft für die Kartografie teilen und gemeinsam dazu beitragen, eine präzise und umfassende Darstellung unserer Welt zu schaffen. In diesem Kontext werden die einzelnen Datenobjekte in OSM nicht nur als isolierte Punkte betrachtet, sondern als Teil eines größeren, globalen Mosaiks, das von der kollektiven Anstrengung einer internationalen Gemeinschaft geformt wird.

2.1.3 Nodes

Im Kontext von OpenStreetMap dienen Nodes (Knotenpunkte) als grundlegende Bausteine, um präzise geografische Daten zu repräsentieren. Es hilft zu betonen, dass OSM-Nodes nicht als abstrakte Punkte in einem Graph zu verstehen sind, wie es in der Graphentheorie der Fall ist. Vielmehr handelt es sich um konkrete geografische Punkte auf der Erde.

Die Analogie besteht darin, Nodes als digitale Markierungen auf einer Weltkarte zu betrachten. Jede Node repräsentiert einen eindeutigen Punkt mit spezifischen Koordinaten, vergleichbar mit einer virtuellen Stecknadel, die exakt auf die Position eines realen Objekts auf der Erde zeigt.

Die Funktion dieser digitalen Knotenpunkte erstreckt sich über die bloße Kartierung hinaus. Sie dienen nicht nur als Basis für die Darstellung von Straßen, Gebäuden oder Bäumen, sondern auch als Bausteine für komplexe geografische Strukturen. In diesem Sinne sind sie der Schlüssel, um die Vielfalt der realen Welt in einer digitalen Umgebung abzubilden.

Jede Node in OSM verfügt über eine eindeutige Id, die als Link fungiert. Diese Id ermöglicht es, von größere Strukturen, wie beispielsweise Straßen oder Häuserblöcken, zu verlinken. Nodes sind besonders wichtig, um die Eckpunkte, Kurven und Enden von Straßen zu definieren und somit komplexe geografische Strukturen zu modellieren, siehe Abbildung 2.2.

Zusätzlich zu den geografischen Koordinaten enthält jede Node einen Zeitstempel und den Namen des Nutzers, der die Node hinzugefügt hat. Die Integration eines Features zur

Visualisierung von Zeit und Nutzer in der Karte wurde diskutiert, aber aufgrund des Ziels dieser Arbeit nicht weiter verfolgt.

In der Regel besitzen Nodes keine weiteren Eigenschaften, jedoch können sie kommentiert werden. Diese zusätzlichen Informationen können in Form von sogenannten „Tags“ definiert werden. Tags bestehen aus unterschiedlichen Schlüsseln, die auf Werte zeigen und es ermöglichen, spezifische Eigenschaften wie Beleuchtung, Zebrastreifen oder Ampeln auf Straßen zu definieren. Diese Technik erweitert die Vielseitigkeit von Nodes in der Darstellung komplexer geografischer Informationen in der OSM-Datenbank.

Beispielsweise liefert die vorliegende XML-Node (2.2 ID: 4417387480) in der OpenStreetMap-Datenbank eine detaillierte Beschreibung eines markanten Objekts in Vaihingen: Es handelt sich um einen Baumstamm, der als öffentliche Bücherei oder „Baumbibliothek“ fungiert. Diese Initiative ermöglicht es der Gemeinschaft, Bücher zu tauschen und zu lesen. Der Baumstamm ist mit Fächern ausgestattet, die den Austausch von Büchern erleichtern.

Die Attribute dieser Nodes bieten zusätzliche Einblicke. Der Timestamp (Zeitstempel) zeigt an, dass die Informationen zuletzt am 20. Mai 2023 um 15:11:24 UTC aktualisiert wurden. Der Benutzer „FileX_Stuff“ mit der Benutzer-ID 16851129 ist für die Aktualisierung verantwortlich und hat dies in der fünften Version dieser Node vorgenommen.

Verschiedene „Tags“ (Schlüssel-Wert-Paare) verfeinern die Beschreibung weiter. Der Tag „amenity“ mit dem Wert „public_bookcase“ weist darauf hin, dass es sich um eine öffentliche Bücherei handelt. Der Tag „description“ gibt eine präzise Beschreibung des Objekts, nämlich einen Baumstamm mit Fächern zum Tauschen und Lesen von Büchern an. Weitere Tags, wie „name“, „opening_hours“ und „operator“, liefern zusätzliche Informationen wie den Namen der Einrichtung („Baumbibliothek“), die Öffnungszeiten (24/7) und den Betreiber (Landeshauptstadt Stuttgart, Garten-Friedhofs-und-Forst-Amt).

```
1  <node id="4417387480" visible="true" version="5" changeset="136341854"
         timestamp="2023-05-20T15:11:24Z" user="FileX_Stuff" uid="16851129"
         lat="48.7349674" lon="9.0882558">
2    <tag k="amenity" v="public_bookcase"/>
3    <tag k="check_date" v="2023-05-20"/>
4    <tag k="description" v="Baumstamm mit Fächern zum Tauschen und Lesen von
         Büchern"/>
5    <tag k="name" v="Baumbibliothek"/>
6    <tag k="opening_hours" v="24/7"/>
7    <tag k="operator" v="Landeshauptstadt Stuttgart,
         Garten-Friedhofs-und-Forst-Amt"/>
8  </node>
9
```

Abb. 2.2: Ein Beispiel wie eine Node in OSM aussieht.

Diese beispielhafte Node illustriert die Vielseitigkeit von OpenStreetMap als Plattform, die nicht nur geografische Positionen, sondern auch umfassende Informationen zu realen Objekten in der physischen Welt bietet.

2.1.4 Way

Ways, als zentrale Datenstruktur in OpenStreetMap (OSM), stellen eine Möglichkeit dar, Linien oder Flächen in der realen Landschaft abzubilden. Ihre Flexibilität eröffnet eine breite Palette von Anwendungen, von der Darstellung von Straßen und Wegen bis hin zur Definition von Grenzen für Häuser, Landschaftselemente wie Wälder und Wiesen sowie sogar die Abbildung von Innenhöfen von Gebäuden.

Die Linien, die durch Ways definiert werden, repräsentieren unterschiedliche Elemente in der physischen Welt.

Neben Straßen können Ways auch als Grenzen dienen, um unterschiedliche Nutzungen von Flächen zu definieren. Sei es die Begrenzung von Wohngebieten, landwirtschaftlichen Flächen, Wäldern oder anderen Bereichen – Ways spielen eine elementare Rolle bei der präzisen Kartierung und Beschreibung der realen Welt in der OSM-Datenbank.

Es ist wichtig zu betonen, dass die Darstellung von Häusern durch Ways eine besondere Eigenschaft ist, da sie nicht nur als Linien, sondern auch als Flächen modelliert werden können. Dies unterscheidet sie von Straßen, die primär als Linien repräsentiert werden. Dieser Unterschied ermöglicht eine nuancierte und detaillierte Erfassung von geografischen Elementen in OpenStreetMap. In dem XML-Abschnitt den man im Bild 2.3 sieht wird ein Fußweg beschrieben. Bei der Beschreibung von realen Wegen erfolgt oft eine Unterteilung in mehrere Abschnitte, die alle von verschiedenen Ways beschrieben werden. Daher genügt es, diesen Weg mit nur zwei Nodes zu charakterisieren, die klar definieren, wo der Way beginnt und endet.

```
1  <way id="299488473" visible="true" version="4" changeset="124552160"
         timestamp="2022-08-06T09:01:01Z" user="xMaxnetx" uid="10868922">
2   <nd ref="3034741362"/>
3   <nd ref="3034741365"/>
4   <tag k="highway" v="footway"/>
5   <tag k="lit" v="yes"/>
6   <tag k="smoothness" v="good"/>
7   <tag k="surface" v="paving_stones"/>
8  </way>
```

Abb. 2.3: Ein Way aus einer OSM-Datei

In Situationen, in denen ein Fußweg oder eine Straße kurvig verläuft, werden für die Kurvenform zusätzliche Nodes herangezogen. Diese präzise Node-Definition ermöglicht eine akkurate Repräsentation der tatsächlichen Gegebenheiten vor Ort.

Die Tags in diesem Abschnitt bieten nicht nur Informationen über den Wegtyp – in diesem Fall ein Fußweg (Highway vom Typ „Footway“) – sondern ermöglichen auch die Speicherung weiterer relevanter Details. Zum Beispiel gibt der Tag „lit“ an, dass der Weg beleuchtet ist, „smoothness“ beschreibt die Beschaffenheit des Weges als gut, und „surface“ gibt Auskunft über den Bodentyp, in diesem Fall Pflastersteine. Diese zusätzlichen Informationen bereichern die Beschreibung des Fußwegs um Details, die für die Kartierung und Nutzung der Daten von Bedeutung sein können.

2.1.5 Relationen

Relationen (auf Deutsch übersetzt: Beziehungen) in OSM bieten die Möglichkeit, abstrakte Zusammenhänge zwischen verschiedenen Objekten zu modellieren. Dies ermöglicht eine sinnvolle Gruppierung von Elementen, die in einer logischen Beziehung zueinander stehen. Ein Beispiel hierfür sind Bus- und Bahnlinien, bei denen Relationen genutzt werden, um die Verbindung der verschiedenen Streckenabschnitte zu beschreiben.

Ein besonderes Anwendungsgebiet von Relationen liegt in der Darstellung von Gebäudestrukturen, insbesondere solchen mit komplexen inneren Strukturen wie Innenhöfen. Im Gegensatz zu Knotenpunkte, Wege und simplen Flächen ermöglichen Relationen eine präzise Modellierung von komplexen Zusammenhängen. Häuser mit Innenhöfen, deren Ways und Nodes keine individuellen Tags enthalten, werden mithilfe von Relationen dargestellt. Ohne die Berücksichtigung von Relationen würden solche Gebäudestrukturen für das Programm quasi nicht existieren.

Ein praktisches Beispiel für die Verwendung von Relationen ist das Informatik-Gebäude der Universität Stuttgart. Hierbei wird ein äußerer Way genutzt, um den Umriss des Gebäudes zu zeichnen, während innere Ways die verschiedenen Innenhöfe repräsentieren. Diese komplexe Struktur wäre ohne die Verwendung von Relationen äußerst schwierig präzise und detailliert abzubilden.

Somit bieten Relationen in OpenStreetMap eine einfache Möglichkeit, um Beziehungen zwischen Elementen zu definieren, insbesondere bei komplexen Gebäude-, Bahn- und Verkehrsstrukturen.

```
<way id="221018878" visible="true" version="1" changeset="16092611"
       timestamp="2013-05-12T09:58:46Z" user="fx99" uid="130472">
  <nd ref="2300715256"/>
  <nd ref="2300715254"/>
  <nd ref="2300715242"/>
  <nd ref="2300715243"/>
  <nd ref="2300715256"/>
 </way>

<relation id="2922269" visible="true" version="7" changeset="110777068"
       timestamp="2021-09-06T05:59:27Z" user="Ze0zohk1" uid="4565074">
  <member type="way" ref="221018878" role="inner"/>
  <member type="way" ref="221018877" role="inner"/>
  <member type="way" ref="221018880" role="inner"/>
  <member type="way" ref="221018879" role="inner"/>
  <member type="way" ref="22741007" role="outer"/>
  <tag k="addr:city" v="Stuttgart"/>
  <tag k="addr:country" v="DE"/>
  <tag k="addr:housenumber" v="38"/>
  <tag k="addr:street" v="Universitätsstraße"/>
  <tag k="building" v="university"/>
  <tag k="building:colour" v="white"/>
  <tag k="building:levels" v="3"/>
  <tag k="name" v="Informatik-Institute"/>
  <tag k="roof:shape" v="flat"/>
  <tag k="type" v="multipolygon"/>
  <tag k="wheelchair" v="yes"/>
 </relation>
```

Abb. 2.4: Eine Relation und ein Weg in OSM-Format

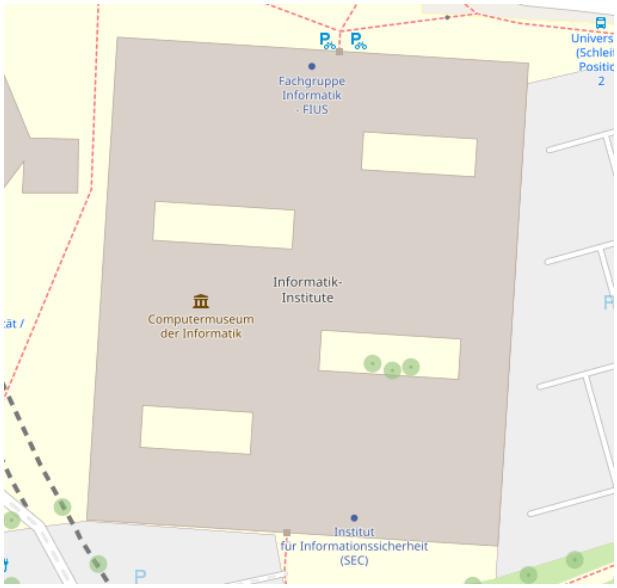

Abb. 2.5: Ein Gebäude mit Innenhof, das mit einer Relation dargestellt wird

In der vorliegenden Beispiel-XML (Bild 2.4) sind zuerst zwei Elemente beschrieben: ein Way (ID: 221018878) und eine Relation (ID: 2922269), die gemeinsam das Informatik-

gebäude der Universität Stuttgart repräsentieren. In Abbildung 2.5 sieht man wie die Relation auf OpenStreetMap aussieht [2].

Der Way (ID: 221018878) definiert einen Pfad mit den zugehörigen Nodes, die durch ihre Referenznummern (ref) identifiziert werden. Diese Nodes markieren den Umriss eines Innenhofes des Gebäudes.

Die Relation (ID: 2922269) aggregiert mehrere Ways, die auf irgendeine Weise miteinander verbunden sind. Jeder Way spielt dabei eine spezifische Rolle („role“) innerhalb der Relation. Der Way mit der Rolle „outer“ (ID: 22741007) definiert den äußeren Umriss des Gebäudes, während die Ways mit der Rolle „inner“ die Innenhöfe repräsentieren.

Die Tags innerhalb der Relation bieten zusätzliche Informationen über das Gebäude. Dazu gehören Adressdetails (addr:city, addr:country, addr:housenumber, addr:street), die Gebäudeart (building), die Gebäudefarbe (building:colour), die Anzahl der Stockwerke (building:levels), der Name des Gebäudes (name), die Form des Dachs (roof:shape), der Typ der Struktur (type) und die Zugänglichkeit für Menschen im Rollstuhl (wheelchair).

2.1.6 Osmium

Osmium ist eine C++-Bibliothek und Toolset zur Verarbeitung von OSM-Daten. Es bietet eine breite Palette von Funktionen zum Lesen, Schreiben, Manipulieren und Analysieren von OSM-Daten. Osmium unterstützt verschiedene Formate von OSM-Daten, einschließlich der bekannten .osm- und .pbf-Formate [11].

2.2 Minecraft

Minecraft, entwickelt von Mojang, schafft seit 2009 eine einzigartige Spielerfahrung, die Kreativität, Abenteuer und Baukunst vereint. Die offene Sandbox-Umgebung erlaubt es den Spielern, eigene Welten zu erschaffen und zu erkunden. Die Grundlagen von Minecraft erstrecken sich über verschiedene Aspekte, von der Erkundung und dem Überleben bis hin zum kreativen Bauen und der Interaktion im Mehrspielermodus.

Das Spiel basiert auf einfachen, aber vielseitigen Prinzipien, bei denen Spieler Blöcke abbauen, Ressourcen sammeln und Strukturen errichten können. Die Welt ist in würfelförmige Blöcke unterteilt, die eine Vielzahl von Materialien repräsentieren. Im Überlebensmodus müssen Spieler Nahrung finden, Monster abwehren und klug mit Ressourcen umgehen. Der kreative Modus ermöglicht unbegrenzte Ressourcen und uneingeschränkte Gestaltungsmöglichkeiten.

Die nächtliche Dunkelheit bringt feindliche Kreaturen hervor, während tagsüber die Welt erkundet werden kann. Der Multiplayer-Modus fördert die soziale Zusammenarbeit, indem

Spieler gemeinsam Projekte umsetzen, miteinander handeln und interagieren. Die lebendige Community erstellt ständig Modifikationen, die das Spiel um neue Funktionen und Texturen erweitern.

Diese Grundlagen bieten einen Einblick in die faszinierende Welt von Minecraft, die sowohl für kreative Köpfe als auch für Abenteuerlustige unendliche Möglichkeiten bietet. Die Bachelorarbeit wird vertiefend die Schlüsselelemente von Minecraft erforschen und ihre Bedeutung in Bezug auf die zugrunde liegenden Prinzipien und Spielererfahrung beleuchten.

2.2.1 Koordinaten

In Minecraft erfolgt die Darstellung von Koordinaten im euklidischen Raum, was bedeutet, dass die Welt auf parallelen Linien basiert, die auch in der Realität parallel bleiben. Im Gegensatz dazu befinden wir uns in der wirklichen Welt auf einer kugelförmigen Erde, auf der lokal parallele Linien irgendwann zusammenlaufen [12].

Da ein Minecraftcharakter (Spieler) 1.8 Blöcke groß ist, kann man davon ausgehen, dass ein Block ungefähr einem Meter entspricht [13]. Das kann man auch im Bild 2.6 erkennen. Der Eisenblock im Hintergrund hilft die Pixel zu zählen.

Der Ursprung der Minecraft Welt, der Weltmittelpunkt, liegt bei den Koordinaten 0, 0, 0. Obwohl es einen Befehl gibt, um den Bereich einer Minecraft Welt zu begrenzen, ist eine normale Welt 60 Millionen mal 60 Millionen Blöcke groß. Diese Größe ist mehr als sechsmal größer als die Erde selbst und sollte ausreichen, um die reale Welt in Minecraft zu repräsentieren.

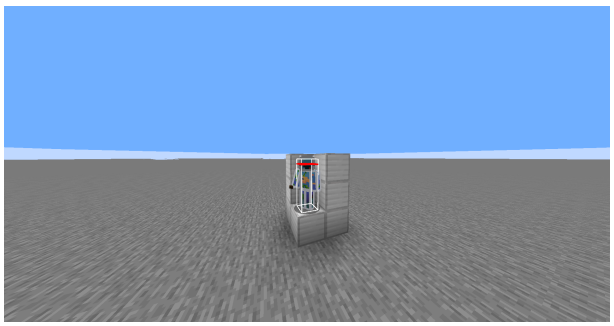

Abb. 2.6: Die Größe eines Minecraftspielers im Vergleich zu Eisenblöcken

2.2.2 Blöcke

In Minecraft entspricht jeder Block einem bestimmten Blockzustand, was bedeutet, dass an einer Integerposition in der Welt ein spezifischer Blockzustand existiert. Ein Blockzustand repräsentiert die Art und Eigenschaften eines Blocks an einer bestimmten Stelle. Zum Beispiel kann Wolle in verschiedenen Farben auftreten, und jede dieser Farben wird durch einen eigenen Blockzustand repräsentiert.

Man kann sich die Minecraft Welt wie ein riesiges drei-dimensionales Array vorstellen in dem jeder Eintrag auf ein Blockzustand zeigt, der dann an dieser Position zu sehen ist. So hat jeder Block einen klar abgegrenzten Bereich und sie können sich nicht überschneiden. Im Laufe der Zeit wurden bis zu 153 verschiedene Blockzustände in Minecraft eingeführt, von denen viele von realen Materialien inspiriert sind [14].

Obwohl einige Blöcke in Minecraft möglicherweise nicht realistisch sind, erfüllen sie den Zweck, unbekannte oder fehlerhafte Strukturen darzustellen. Sie dienen als visuelle Hinweise für den Spieler, dass an dieser Stelle etwas vorhanden ist, und oft lässt sich aus der Umgebung ableiten, welcher Zweck oder welches Element repräsentiert wird.

2.2.3 Minecraft Welt

Dieser Abschnitt beschäftigt sich damit, wo die Minecraft Welten gespeichert werden.

Jede betretbare Minecraft Welt liegt im Verzeichnis „saves“. Hier gibt es Unterordner, die die jeweilige Welt enthalten. Beim Durchsuchen dieser Unterordner überprüft Minecraft, ob eine „level.dat“-Datei vorhanden ist. Bei Vorhandensein werden der Name und weitere Daten ausgelesen und in einer Liste angezeigt. Wenn man tatsächlich eine Welt betritt, spielt der Ordner „region“ eine entscheidende Rolle. Hier werden Dateien mit der Endung .mca gefunden, in denen die konkreten Blockdefinitionen festgelegt sind. Eine Region ist 512x512 Blöcke groß und wird durch eine dieser Dateien repräsentiert. Es existieren weitere Dateien, die von Minecraft für die Speicherung von Weltinformationen genutzt werden. Für diese Arbeit ist jedoch nur der „datapacks“-Ordner von Bedeutung.

Mit dem Caves & Cliffs-Update von Minecraft wurde die Standardgenerierung der Welten verbessert, und die neue Höhlengenerierung führte zu einer Erweiterung der Höhe in Minecraft [15] . Dennoch reicht diese Erweiterung nicht für die Darstellung der realen Welt aus. Glücklicherweise können wir die Höhe mit einem Datapack anpassen. Minecraft verwendet viele Konstanten, um Eigenschaften in der Welt festzulegen. Gleichzeitig ermöglicht es den Spielern, über den „datapacks“-Ordner Datapacks zu laden, die das Spielerlebnis ändern, indem sie diese Konstanten anders setzen. In unserem Fall möchten wir beispielhaft die minimale und maximale Bauhöhe anpassen.

In Minecraft sind verschiedene Konstanten festgelegt, die Aspekte wie die Welthöhe definieren, um einen unterhaltsamen Spielverlauf zu gewährleisten. In den neueren Versionen besteht nun die Möglichkeit, einige dieser Konstanten auf eigene Gefahr zu verändern. Diese Option ist für unsere Arbeit äußerst nützlich, da sie es uns ermöglicht, Blöcke an ihrer realen Höhenposition zu platzieren und somit eine präzisere Umgebung zu schaffen.

2.3 SRTM

Die Space Shuttle Endeavour führte im Jahr 2000 die Shuttle Radar Topography Mission (SRTM) durch, eine 11-tägige Mission, deren Hauptziel darin bestand, Höhendaten der Erde mithilfe eines Radarsystems zu sammeln. Diese Daten wurden als Grundlage für die Erstellung von präzisen Höhenkarten verwendet. Die Abkürzung „SRTM“ steht dabei für Shuttle Radar Topography Mission [16].

Die ursprünglich generierten Daten wiesen nach der Messung einige Lücken auf, insbesondere an Stellen, wo Wasser die Höhenmessung beeinträchtigt hatte, wie beispielsweise im Himalaya-Gebirge. Auf der Website https://srtm.csi.cgiar.org/ wurden diese Fehler korrigiert und zum Download angeboten. Es gibt verschiedene Formate, in denen die Daten verfügbar sind, darunter .tiff (Bilddatei) und .asc (Textdatei). In diesem Projekt verwenden wir das .asc-Format, da es einfacher ist, das Format zu verstehen und Ausrichtungsprobleme zu debuggen [17].

2.3.1 Interpolation

Interpolation ist ein grundlegendes Konzept in der Mathematik und Datenanalyse, das verwendet wird, um fehlende Werte zwischen bekannten Datenpunkten zu schätzen oder zu berechnen. Es ist besonders nützlich, wenn wir nur diskrete Datenpunkte haben, aber den Wert einer Funktion an einem bestimmten Punkt dazwischen kennen möchten.

Im Wesentlichen beinhaltet Interpolation die Schätzung von Werten innerhalb eines bekannten Datensatzes basierend auf den Werten an benachbarten Punkten. Es gibt verschiedene Interpolationsmethoden, darunter lineare Interpolation, polynomiale Interpolation, spline-basierte Interpolation und kubische Interpolation.

Die einfachste Methode ist die lineare Interpolation. Dabei wird nur eine Gerade zwischen zwei benachbarten Datenpunkten gezogen, und der Wert des zu interpolierenden Punktes wird auf dieser Geraden festgelegt. Dies ist die einfachste Form der Interpolation und wurde deshalb in der Erstimplementation der Arbeit verwendet.

In dieser Arbeit wurde die polynomiale Interpolationsmethode implementiert. Bei der Polynomiale Interpolation wird eine Kurve durch die bekannten Datenpunkte gelegt, und der Wert des zu interpolierenden Punktes wird durch Auswerten dieses Polynoms an der gewünschten Stelle berechnet.

2.4 Mercator

Eine Projektion ist erforderlich, um die gekrümmte Oberfläche der Erde auf eine flache Karte abzubilden, da die Erde annähernd kugelförmig ist, während Karten normalerweise flach sind. Die Projektion ermöglicht es, die gekrümmte Erdoberfläche in einem zweidimensionalen Format darzustellen, was für verschiedene Anwendungen unerlässlich ist.

Die Mercator-Projektion, obwohl weit verbreitet und nützlich für die Navigation, weist ein grundlegendes Problem auf: Die Flächen werden abhängig von der Breitengrad unterschiedlich stark verzerrt. Dieses Phänomen tritt aufgrund der spezifischen Art und Weise auf, wie die Mercator-Projektion die dreidimensionale Erde auf eine zweidimensionale Karte abbildet.

Insbesondere bei höheren Breitengraden wird die Verzerrung besonders deutlich und kann zu irreführenden Darstellungen der tatsächlichen Größe von Ländern und Kontinenten führen. Die Verzerrung wird insofern deutlich, dass je weiter man vom Äquator entfernt ist, Landflächen exponentiell größer erscheinen.

Trotz dieser Verzerrungen bietet die Mercator-Projektion den großen Vorteil, Abbildungen winkeltreu darzustellen, was für die Navigation und andere Anwendungen von entscheidender Bedeutung ist [18].

Das Bild einer Kerze in einer Kugel, die Licht durch Öffnungen strahlt, die einen umgebenden Zylinder beleuchten, ist eine anschauliche Darstellung der Mercatorprojektion. Die Öffnungen auf der Höhe der Kerze behalten ihre Größe bei, während sich die Öffnungen weiter vom Äquator entfernt proportional vergrößern. Dieses Experiment verdeutlicht, wie die Mercatorprojektion die Krümmung der Erde auf eine flache Karte abbildet. Je weiter man sich von den Breitengraden des Äquators entfernt, desto stärker werden die Verzerrungen, ähnlich wie die Vergrößerung der Öffnungen auf dem umgebenden Zylinder.

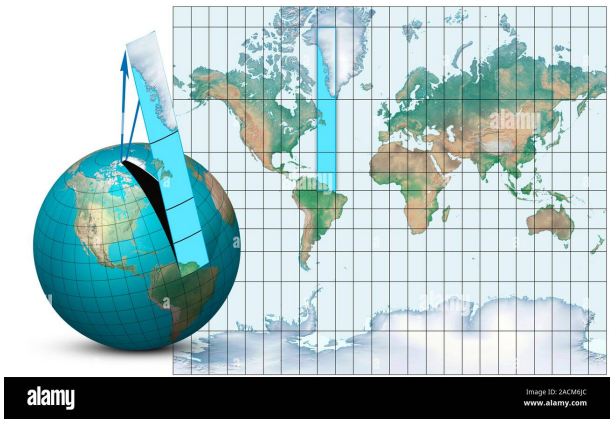

Abb. 2.7: Mercator Visualisierung von einer Kugel zu einer zweidimensionalen Karte

Wenn man einen Ausschnitt von OpenStreetMap bekommen will, kann man die Selektion mit Latitude und Longitude Koordinaten festlegen. Streng genommen müsste dieser Ausschnitt ein Trapez sein (siehe Bild 2.7). Je nachdem wie weit nördlich oder südlich man ist, ist dieses Trapez nach oben oder unten geöffnet. Dies liegt daran, dass die Latitude Koordinaten vom Südpol zum Nordpol gehen und somit jede Latitude Koordinate zu einem Punkt zusammenführen.

Allerdings ist der Ausschnitt von OpenStreetMap ein Rechteck. Das deutet darauf hin, dass OpenStreetmap die Mercator Projektion verwendet, um herauszufinden, ob Nodes noch in einem Abschnitt liegen oder nicht. Um das Programm dieser Arbeit nicht zu komplex zu gestalten, haben wir diese Projektionsweise übernommen. Sonst müsste ein größerer Ausschnitt abgefragt werden oder mehrere Abfragen um die nebenliegenden Sektionen auch auszulesen.

3 Realisierung

Im Realisierungskapitel wird der Prozess der Umsetzung unseres Programms zur Konvertierung von OSM-Daten in Minecraft Welten detailliert beschrieben. Wir beginnen mit den Anforderungen, was das Programm können muss sowie der Aufbau des Systems. Anschließend werden die einzelnen Schritte des Konvertierungsprozesses erläutert, beginnend mit der Datenvorbereitung bis hin zur Generierung der Minecraft Welt. Dabei werden auch spezifische Implementierungsdetails und Herausforderungen diskutiert. Schließlich werden wir im Nachfolgenden Kapitel 4 die Leistung und Effektivität unseres Programms bewerten.

3.1 Anforderungen

Die vorliegende wissenschaftliche Arbeit beschäftigt sich mit der Konzeption und Implementierung eines Webdienstes zur Konvertierung von OpenStreetMap (OSM) und Höhendaten in das Minecraft-Format. Das Ziel des Projekts besteht darin, einen benutzerfreundlichen Webserver zu entwickeln, der es Nutzern ermöglicht, einen ausgewählten Bereich auf der Karte zu bestimmen und diesen in Minecraft-Daten umzuwandeln. Nachfolgend sind die detaillierten Anforderungen für das Projekt aufgeführt:

1. **Webserver-Funktionalität:**
    - Implementierung eines Webdienstes, der das Starten eines Servers ermöglicht.
    - Bereitstellung einer benutzerfreundlichen webbasierten Benutzeroberfläche für den Zugriff auf den Server.
2. **Bereichsauswahl:**
    - Integration einer Funktion, die es Benutzern ermöglicht, einen bestimmten Bereich auf der Karte auszuwählen.
    - Möglichkeit zur Auswahl des Bereichs durch Klicken und Ziehen auf einer interaktiven Kartenansicht.

3. **Integration von Höhendaten:**
   - Berücksichtigung von Höhendaten für die realistische Darstellung des Geländes in Minecraft.
   - Umsetzung von Höhenunterschieden und -konturen für eine präzise Konvertierung.
4. **Integration von OSM-Daten:**
   - Die einbeziehung von OSM-Daten in den Konvertierungsprozess für die Erfassung von Straßen, Gebäuden und anderen Strukturen.
   - Umfassende Umwandlung von OSM-Tags in entsprechende Minecraft-Elemente.
5. **Benutzerfreundliche Schnittstelle:**
   - Entwicklung einer intuitiven Benutzeroberfläche für eine einfache Navigation und Auswahl der Funktionen.
   - Bereitstellung klarer Anweisungen und Rückmeldungen während des Auswahl- und Konvertierungsprozesses.
6. **Effiziente Datenverarbeitung:**
   - Implementierung einer effizienten Umwandlung für bestmögliche Leistung auf Server- und Benutzerseite.
   - Optimierung von Datenverarbeitung und -speicherung.
7. **Validierung und Fehlerbehandlung:**
   - Validierung von Benutzereingaben zur Vermeidung ungültiger Auswahlbereiche.
   - Implementierung von Fehlerbehandlungsmechanismen für unerwartete Situationen.

3.2 Konzeption

Die konzeptionelle Herangehensweise des Projekts ist darauf ausgerichtet, die Daten kohärent zu bearbeiten, um die Lesbarkeit des Programmcodes zu verbessern und gleichzeitig den maximalen Speicherbedarf klar zu definieren. Ein strukturierter Arbeitsfluss wird angestrebt, beginnend mit dem Einlesen der Daten, gefolgt von der Identifizierung markanter Landschaftsobjekte.

Nach dem erfolgreichen Filtern von Gebäuden, Gewässern, Straßen und anderen Elementen können die Daten in eine 2D-Karte übertragen werden. Diese Karte dient als vereinfachte Darstellung der Welt, wobei jeder Punkt eine bestimmte Koordinate repräsentiert. Diese 2D-Darstellung erleichtert das Bearbeiten der Welt, insbesondere wenn Objekte auf denselben Koordinaten auftreten, wie beispielsweise Straßen und Gebäude. Die erstellte 2D-Karte wird dann umgewandelt, um in die dreidimensionale Minecraft Welt integriert zu werden. Dieser Prozess ermöglicht eine präzise Übertragung der gezeichneten 2D-Map in das 3D-Minecraft-Format. Besondere Aufmerksamkeit wird dabei auf die Koexistenz von verschiedenen Elementen an denselben Koordinaten gelegt, wie etwa Wege und Gebäude.

Diese strukturierte Vorgehensweise unterstützt nicht nur eine effiziente Bearbeitung der Daten, sondern ermöglicht auch eine klare Umsetzung in die Minecraft Welt. Die Integration von markanten Landschaftsobjekten und die korrekte Übertragung der 2D-Karte in die 3D-Welt stellen wichtige Schritte dar, um eine präzise und realistische Repräsentation der geografischen Daten in Minecraft zu gewährleisten.

3.3 Aufbau

Die Architektur eines Softwareprojekts bildet das strukturelle Rückgrat, das die Grundlage für eine effiziente Entwicklung, Skalierbarkeit und Wartbarkeit schafft. Im Kontext dieses Projekts, das die Konvertierung von OpenStreetMap (OSM) und Höhendaten in das Minecraft-Format zum Ziel hat, spielt der Aufbau eine entscheidende Rolle bei der Umsetzung dieses komplexen Vorhabens.

Angesichts der verfügbaren Daten war es wichtig, das Programm so zu strukturieren, dass das Hinzufügen weiterer OSM-Daten-Konvertierungen leicht möglich ist.

Die Konvertierung des Projekts gliedert sich in drei wesentliche Teile. Der erste Schritt besteht darin, Informationen aus einer OpenStreetMap (OSM)-Datei zu extrahieren. Die präzise Repräsentation dieser Daten ist von entscheidender Bedeutung, um alle relevanten Informationen effizient abzurufen. In einem Prototypen der Software stieß diese Phase auf Schwierigkeiten, da Tags nur in Verbindung mit Wegen gespeichert wurden. Es wurde erkannt, dass Relations und Nodes ebenfalls wichtige Datenpunkte enthalten können und müssen.

Das Herzstück des Programms liegt in der eigentlichen Konvertierung von OSM-Daten zu Minecraft-Daten. Minecraft selbst kann als Voxelwelt betrachtet werden, was bedeutet, dass bei der Übertragung viele Details aus der realen Welt verloren gehen können. Ein besonderes Augenmerk liegt auf der Herausforderung, wie Ecken, die nicht entlang der Achsen verlaufen, in Minecraft unterschiedlich aussehen können als in der Realität. Abhängig von der Skalierung besteht die Gefahr, dass mehrere Objekte kollidieren. Diese Problematiken müssen vor dem abschließenden Schritt gelöst werden, in dem die Daten in Minecraft-Dateien geschrieben werden.

Der letzte Schritt ist es von zentraler Bedeutung, sicherzustellen, dass alle erforderlichen Informationen bereitgestellt werden, um Minecraft in die Lage zu versetzen, die Welt anzuzeigen und spielbar zu machen. Eine gründliche Bearbeitung und Umsetzung der Daten ist daher unabdingbar, um eine präzise und vollständige Integration in die Minecraft Welt zu gewährleisten.

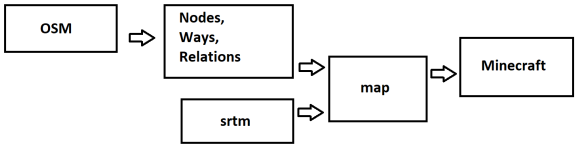

Abb. 3.1: Aufbau des Programms

Die Ausführung des Programms beginnt mit dem Laden der OpenStreetMap- und SRTM-Daten (siehe 3.1). Sobald die Höheninformationen, Wege, Nodes und Relationen erfasst sind, erfolgt die Übertragung in die Karte. Während des Konvertierungsprozesses dient die Karte als zweidimensionales Array, das dazu beiträgt, die Komplexität einer dreidimensionalen Welt zu umgehen.

Die Verwendung dieser zweidimensionalen Repräsentation erleichtert das systematische Schreiben der Daten aus den Nodes, Ways und Relationen in das Array. Dieser Schritt ist wichtig, um eine präzise Abbildung der geografischen Informationen in der Karte zu gewährleisten.

Im abschließenden Schritt erfolgt die Transformation dieser Daten in ein Format, das von Minecraft verstanden und gelesen werden kann. Diese Phase ist maßgeblich für die erfolgreiche Integration der vorverarbeiteten geografischen Informationen in die Minecraft Welt. Jeder der beschriebenen Schritte spielt eine essenzielle Rolle im Gesamtprozess der Konvertierung von OSM- und SRTM-Daten in das Minecraft-Format.

3.3.1 Datei einlesen

Im folgenden Abschnitt wird der Vorgang der Datenintegration eingehend betrachtet. Hierbei steht im Mittelpunkt, wie die umfangreichen Informationen aus den OpenStreetMap (OSM)-Dateien effizient und präzise eingelesen werden. Dieser Schritt bildet die Grundlage für die weiteren Bearbeitungsschritte, die es ermöglichen, die Vielzahl an geografischen Daten in das gewünschte Format zu überführen. Durch eine gezielte Auswahl und Sicherung relevanter Informationen, insbesondere hinsichtlich des Kartenausschnitts sowie aller Nodes, Wege und Relationen, wird eine gute Grundlage für die Modellierung der Landschaft in Minecraft geschaffen.

Angefangen wird mit dem Programm Osmium, das alle Nodes, die im gewünschten Teilabschnitt liegen, aus einer größeren Datei herausfiltert.

Der Fokus liegt darauf, sämtliche relevanten Informationen aus der OSM-Datei zu extrahieren und effektiv zu speichern. Hierzu zählt insbesondere die Sicherung des Kartenausschnitts, den OSM in seinen Daten bereitstellt. Die OSM-Daten beinhalten alle Wege, Relationen und Nodes innerhalb des ausgewählten Ausschnitts. Ohne gezielte Maßnahmen würden viele Wege, die aus der Welt herausführen und letztendlich nicht relevant sind, in die Berechnungen einbezogen werden – potenziell sogar störend, wenn mehrere Weltabschnitte miteinander verbunden werden sollen. Der Ausschnitt ermöglicht die präzise Abgrenzung von Objektabschnitten, die aufgrund ihrer Entfernung irrelevant erscheinen.

Zudem zeichnet sich die OSM-Datei durch eine Vielzahl von Nodes, Wegen und Relationen aus. Es ist von essenzieller Bedeutung, alle Nodes, Wege und Relationen in diesem Schritt sorgfältig zu sichern. Selbst Objekte, die keine Tags enthalten, könnten von Relationen genutzt werden, und ihre korrekte Darstellung erfordert eine umfassende Speicherung.

Beim Betrachten der Dateien fällt auf, dass Relationen häufig verwendet werden, um Bahnstrecken oder Buslinien abzubilden. Gleichzeitig existieren Gebäude mit Innenhöfen, die durch eine Relation dargestellt werden, wobei mindestens zwei Wege als Mitglieder fungieren. Solange keiner dieser Wege zusätzliche Informationen enthält, werden sie nicht als Gebäude markiert und könnten am Ende fehlen, wenn sie nicht durch die Relation gesichert werden.

Ein beträchtlicher Anteil der Nodes weist keine Tags auf. Dies ist nicht ungewöhnlich, da sie oft lediglich die Ecken und Kanten der Wege repräsentieren. Jedoch werden auch Schilder, Bäume und weitere Elemente mithilfe von Nodes dargestellt. Daher ist die umfassende Speicherung aller Nodes, Wege und Relationen von entscheidender Bedeutung, um eine vollständige Konvertierung aller relevanten Landschaftsmerkmale zu gewährleisten.

3.3.2 Höhendaten

Nach dem erfolgreichen Einlesen der Daten widmet sich der nächste Schritt dem Auslesen der Höhendaten. Zu diesem Zweck werden die erforderlichen Dateien berechnet und, sofern sie sich noch nicht im Programmordner befinden, heruntergeladen. Dieser Prozess stellt sicher, dass sämtliche notwendigen Ressourcen vorhanden sind, um präzise Höhendaten für die weitere Verarbeitung zu generieren. Die Höhendaten bestehen aus 5° mal 5° großen Flächen in denen 6000 mal 6000 Höhendateneinträge gespeichert sind. Jeder dieser Flächen muss separat heruntergeladen werden.

Um die Längen- und Breitengradinformationen korrekt zu verarbeiten, wird für jeden Datenpunkt die zugehörige Koordinate in Minecraft berechnet. Hierfür wird zuerst die südwestlichste Ecke dieser Fläche berechnet. Dies wird ein wichtiger Referenzpunkt sein, da hiermit berechnet wird, in welcher Datei ein Höhenpunkt ausgelesen werden muss. Diese Koordinaten werden dann einer Funktion übergeben, die alle benachbarten Punkte abfragt. Die zugrunde liegende Idee besteht darin, eine Funktion zu konzipieren, die einen Graphen erstellt und dabei zwischen allen vier benachbarten Punkten verläuft. Auf diese Weise werden die scharfen Kanten, die durch lineare Interpolation entstehen können, effektiv ausgeglichen. Die äußersten Punkte der Funktion bewirken eine Verschiebung, sodass diese Kanten weniger auffällig sind.

Zuvor wurde das Gaußsche Eliminationsverfahren angewendet, um die Funktion zu berechnen, wodurch lediglich vier Parameter benötigt werden. Diese Parameter sind die Nachbarpunkte sowie deren Nachbarpunkte. Die resultierende Funktion verläuft durch alle vier Punkte. Die Einbeziehung der erweiterten Nachbarpunkte trägt dazu bei, dass die resultierenden Höhendaten besser der Realität entsprechen. Aufgrund der Genauigkeit stimmen die Höhendaten zumindest an den gegebenen Positionen überein, während die Werte dazwischen von der Interpolationsfunktion abhängen.

Da die Höhendaten mehrere Blöcke voneinander entfernt sind, erfolgt im Anschluss eine Iteration durch jedes der enthaltenen Blöcke in einem Quadrat. Dabei werden die Höhenwerte mithilfe der zuvor berechneten Funktion festgelegt. Dieser iterative Prozess ermöglicht eine präzise Anpassung der Höhendaten auf jedes individuelle Blockniveau.

Die beschriebenen Schritte stellen sicher, dass die Höhendaten akkurat in das Minecraft-Format übertragen werden, wodurch eine detaillierte und realistische Landschaftsmodellierung ermöglicht wird. Der Einsatz der Funktionsgraphen gewährleistet dabei eine naturgetreue Anpassung der Höhendaten, sodass potenzielle Unstimmigkeiten in der topografischen Darstellung vermieden werden.

3.3.3 Konvertierung der Objekte

In diesem Kapitel wird der Fokus auf die entscheidende Phase der Objekterkennung in den OpenStreetMap (OSM)-Daten gelegt. Die korrekte Zuordnung und Interpretation von Tags spielt eine zentrale Rolle für die präzise Konstruktion der Minecraft-Landschaft. Hierbei steht die Dynamik der OSM-Daten im Vordergrund, wobei Veränderungen und Anpassungen in der Erfassung von Objekten anhand ihrer Tags betrachtet werden. Die Konzeption sieht vor, dass die Objekte zu Beginn einer gezielten Filterung unterzogen werden. Hierbei werden sogenannte Builder erstellt, die in der darauf folgenden Phase dazu dienen, die Blöcke in der Minecraft Welt zu platzieren. Die Entscheidung, die Objekte in zwei getrennten Phasen zu verarbeiten, bietet den Vorteil, dass abhängige Objekte korrekt gebaut werden können. Dies gewinnt besondere Relevanz in Situationen, in denen sich Gebiete überschneiden oder wenn Straßen aufgrund der blockigen Natur von Minecraft, bedingt durch die fehlenden oder fehlerhaften Tags, ungewöhnlich breit konstruiert werden. In solchen Fällen besteht die Gefahr, dass Straßen über ihre üblichen Grenzen hinauswachsen und in bestehende Gebäude oder Parks eindringen. Die klare Trennung und die vorausschauende Verarbeitung der OSM-Objekte in zwei Phasen tragen dazu bei, solche potenziellen Probleme zu identifizieren und sie in der Bauphase korrekt zu behandeln. Dies führt zu einer präzisen und stimmigen Integration der OSM-Daten in die Minecraft Welt trotz der Herausforderungen, die sich aus der blockigen Natur des Spiels ergeben.

Die Objekterkennung in den OSM-Daten erfolgt in der Regel durch Tests, ob ein bestimmtes OSM-Objekt einen Schlüssel mit einem bestimmten Namen besitzt. Zum Beispiel kann bei Häusern über den Schlüssel „addr:housenumber“ die Hausnummer angegeben werden. Dieser Prozess wurde bewusst abstrakt gehalten, um die Möglichkeit zu bieten, weitere Tests für spezifischere Objekte zu integrieren. Ein Beispiel hierfür könnte die Erstellung spezieller Klassen für verschiedene Oberflächentypen sein, die zusätzlich den Wert-Tag überprüfen, um nur bestimmte Typen zu generieren.

Innerhalb des Rahmens dieser Arbeit wäre die Erstellung spezifischer Codes für alle möglichen Kombinationen von Schlüssel-Wert-Paaren zeitaufwändig und repetitiv. Daher sind für jeden sinnvollen Schlüssel spezielle Klassen vorhanden, die den Wert genauer überprüfen und somit die Minecraft-Oberfläche entsprechend dem festgelegten Typ konstruieren.

Im nachfolgenden Abschnitt erfolgt eine detaillierte Betrachtung der in dieser Arbeit implementierten Objekte.

Bereiche testen

In vielen Fällen war es notwendig, die Anwesenheit eines Blocks in einem bestimmten Bereich zu überprüfen, insbesondere bei der Berechnung von Flächen, beispielsweise für Fußgängerwege, wenn der „area“ -Tag gesetzt ist. Daher ist die Entwicklung eines effizienten Algorithmus von großer Bedeutung.

Mit einem Strahl wird getestet, ob ein Block innerhalb des Bereichs liegt. Dabei müssen alle Linien, die die Fläche umranden, überprüft und gezählt werden. Der entscheidende Punkt dabei ist, die Treffer der Linien zu zählen. Ist die Anzahl ungerade, liegt der Block in der Fläche; sonst befindet er sich entweder außerhalb oder ist mindestens einmal eingetreten und genauso oft wieder ausgetreten aus dem Polygon [19].

Es ist wichtig zu beachten, dass dieser Test an Ecken möglicherweise ungenaue Ergebnisse liefert, da praktisch zwei Linien berührt werden, unabhängig davon, ob sich der Block in der Fläche befindet oder außerhalb. Infolgedessen könnte der Algorithmus fälschlicherweise annehmen, dass eine Berührung das Eintreten und die andere das Austreten markiert, obwohl beide Berührungen eigentlich als dieselbe Stelle gezählt werden sollten. Um dieses Problem zu umgehen, wurden alle Punkte um 0,5 verschoben. Diese Verschiebung stellt sicher, dass ein Strahl und die Kreuzung zweier Linien nicht genau aufeinandertreffen können.

Bereiche

In OpenStreetMap gibt es verschiedene Tags, die anzeigen, in welchem Bereich man sich gerade befindet. Einer der auffälligsten ist „Landuse“, der insbesondere in städtischen Gebieten bei Baustellen oder Grünflächen verwendet wird. Für Freizeitaktivitäten wird der Tag „leisure“ genutzt [20]. Außerdem gibt es noch „natural“ für z.B Wälder. Für Parkplätze wird der „amenity“ Tag verwendet.

Im weiteren Verlauf wird auf die implementierten Bereiche eingegangen und wie sie in Minecraft umgesetzt wurden.

Hier sind die überarbeiteten Abschnitte gemäß Ihrer Anforderungen:

1. **Baustellen:**
    - Am Rand wird versucht, sich mit einem Eisengitter an die Absperrgitter anzunähern.
    - Der Boden auf Baustellen ist oft matschig und dreckig. Dafür passt in Minecraft die Mischung aus Erdblöcken, grober Erde und Podzol (das in Minecraft den Boden von Taigabiomen darstellt).
2. **Diverse Grünflächen und Gärten:**
    - Am passendsten sind dafür Grasblöcke.
    - Bei Gärten wird zusätzlich ein Zaun um die Fläche platziert. (damit ist der „allotments“-Tag gemeint)
3. **Wiesen:**
    - Wiesen (Englisch: meadow) ähneln Grünflächen, weisen jedoch in der Minecraftwelt zusätzlich noch feinere Gräser auf.

4. **Äcker:**
    - Äcker (Englisch: farmland) haben in Minecraft einen äquivalenten Block, der verwendet wird.
5. **Sportplätze:**
    - Sportplätze (Englisch: pitch) bestehen aus hellgrünem Beton. Das entspricht nicht dem realen Material, allerdings gab es keine bessere Darstellungsweise, und die Farbe des Minecraftbetons sah für diesen Zweck am besten aus.
6. **Wasser:**
    - Gewässer werden zwei Meter tief modelliert und an den Rändern nur einen Meter.
7. **Spielplätze:**
    - Es ist schwer, Spielplätze in Minecraft darzustellen. Deswegen wurde als Boden einfach nur ein Pfadblock verwendet. Dieser soll an Sand erinnern, der meistens den Boden von Spielplätzen ausmacht.
8. **Swimming Pools:**
    - Swimmingpools bestehen hauptsächlich aus Wasser und haben am Boden und an den Wänden weißen Beton.
9. **Gletscher:**
    - Gletscher haben eine Eisschicht im Boden.
10. **Sand:**
    - Sand- und Strandgekennzeichnete Gegenden werden mit Sandblöcken dargestellt.
11. **Büsche:**
    - Büsche haben eine Blätterschicht über dem Boden.
12. **Wald:**
    - Wälder zeichnen sich durch ihr intensives Grün aus. Daher besteht der Boden in diesen Gebieten wieder aus Gras, und es werden an zufälligen Positionen Bäume generiert. Diese Bäume werden auch erzeugt, wenn sie separat angegeben werden.

13. **Parkplätze:**
    - Es gestaltet sich als äußerst schwierig, die Anordnung von Parkplätzen zu erfassen. Zudem wäre die Darstellung in Minecraft aufgrund der Notwendigkeit ganzer Blöcke und der Tatsache, dass Parklinien oft weniger als einen Meter breit sind, ungewöhnlich. Aus diesem Grund werden Parkbereiche lediglich mit grauem Beton dargestellt.

Häuser

Häuser können mit besonders vielen Tags ausgeschmückt sein. Auffällig dabei sind Tags die höhenrelevante Informationen beschreiben. Zum Beispiel kann durch Tags wie min_level oder min_height festgelegt werden, dass ein bestimmter Teil eines Gebäudes erst ab einer bestimmten Höhe beginnt. Neben dem Höhentag, der auch bei anderen Objekten verwendet werden kann, gibt es hier zusätzlich den Level-Tag, mit dem die Anzahl der Stockwerke definiert werden kann.

Die Höhe wird gemäß den Angaben auf [21] festgelegt, und innerhalb des Gebäudes werden Holzbretter als Boden für jedes Stockwerk verlegt. Die Wände erhalten darüber hinaus eine zufällig passende Farbe (siehe Bild 3.2).

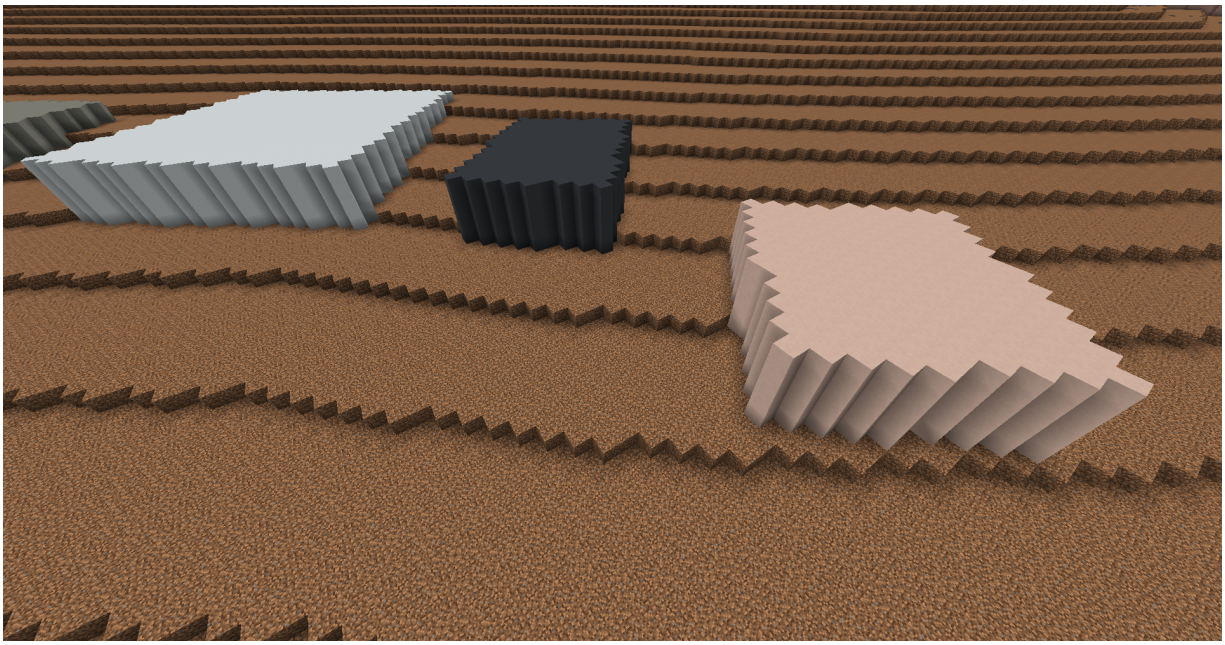

Abb. 3.2: Drei Häuser am Hang von Wernau

Straßen

Im Rahmen der Straßenimplementierung wurde besonderes Augenmerk darauf gelegt, die Vielfalt der von OpenStreetMap bereitgestellten Tags angemessen zu berücksichtigen. Zu Beginn erfolgt eine Typüberprüfung, um einen geeigneten Oberflächentyp zu definieren, falls dieser nicht bereits von OSM festgelegt wurde. Falls keine Breite angegeben ist, wird ein Standardwert verwendet.

In einigen Fällen wird der Weg als Fläche (Area) definiert, was eine spezielle Behandlung erfordert. Alle Blöcke innerhalb dieses Bereichs werden dem entsprechenden Oberflächentyp zugeordnet.

Besondere Herausforderungen ergaben sich bei der Verarbeitung von Gehwegen innerhalb der Straßen. Hier musste berücksichtigt werden, dass möglicherweise sowohl links als auch rechts ein Gehweg vorhanden ist, und der Straßenabschnitt könnte aus mehreren Spuren bestehen. Um diese Parameter unter Berücksichtigung einer beliebigen Skalierung korrekt zu behandeln, wird zuerst die Breite in Metern berechnet. Diese Information wird dann dem modifizierten Bresenham-Algorithmus übergeben, der Linien mit variabler Breite zeichnen kann. Der Bresenham-Algorithmus ist effizient darin, Linien zu zeichnen [22]. In unserem Algorithmus wird diese Linie beliebig breiter. Zusätzlich gibt unser Algorithmus die Position entlang der Breite an. Aufgrund dieser Information kann eine Prozentzahl berechnet werden. Gehwege erhalten ebenfalls ihren prozentualen Anteil. Somit kann der aktuelle Block im Vergleich zu den berechneten Werten bestimmt werden. Wenn der aktuelle Block auf einem Gehweg ist kann eine „glatte Steinstufe“ platziert werden. Ansonsten wird der von den Tags angegebene „surface“-Typ verwendet.

Schienen

Geplant war ursprünglich Bahnschienen mit Ambossen und Holzbrettern zu visualisieren. Allerdings passt diese Darstellung nicht in der Realität 3.3, da die Schienen meist nur eine Breite von zwei Metern haben. Mit nur zwei Blöcken lassen sich Schienen nur schwer darstellen. Da es in Minecraft seit der Alpha Version Schienen gibt, bieten diese eine gute Alternative.

Um die Stellen zu berechnen, an denen Schienen platziert werden sollen, reicht der Bresenham-Algorithmus allein nicht aus. Dies liegt daran, dass nicht alle Blöcke direkt nebeneinander liegen, sondern einige nur an den Ecken verbunden sind. Um dennoch an diesen Ecken Schienen zu platzieren und eine Kurve darzustellen, bietet sich ein sogenannter Raytracing-Algorithmus an. Raytracing-Algorithmen werden in Spielen verwendet, um zu testen, ob ein Objekt mit einem anderen kollidiert. In einem Spiele-Forum hat ein User namens Alice einen Algorithmus vorgestellt, der in Minecraft auf ähnliche Weise verwendet wird [23].

Der Algorithmus vergleicht den Fortschritt für jede Achse, auf der sich die Linie bewegt. Dabei wird auf der Achse, die am wenigsten fortgeschritten ist, als nächstes fortbewegt.

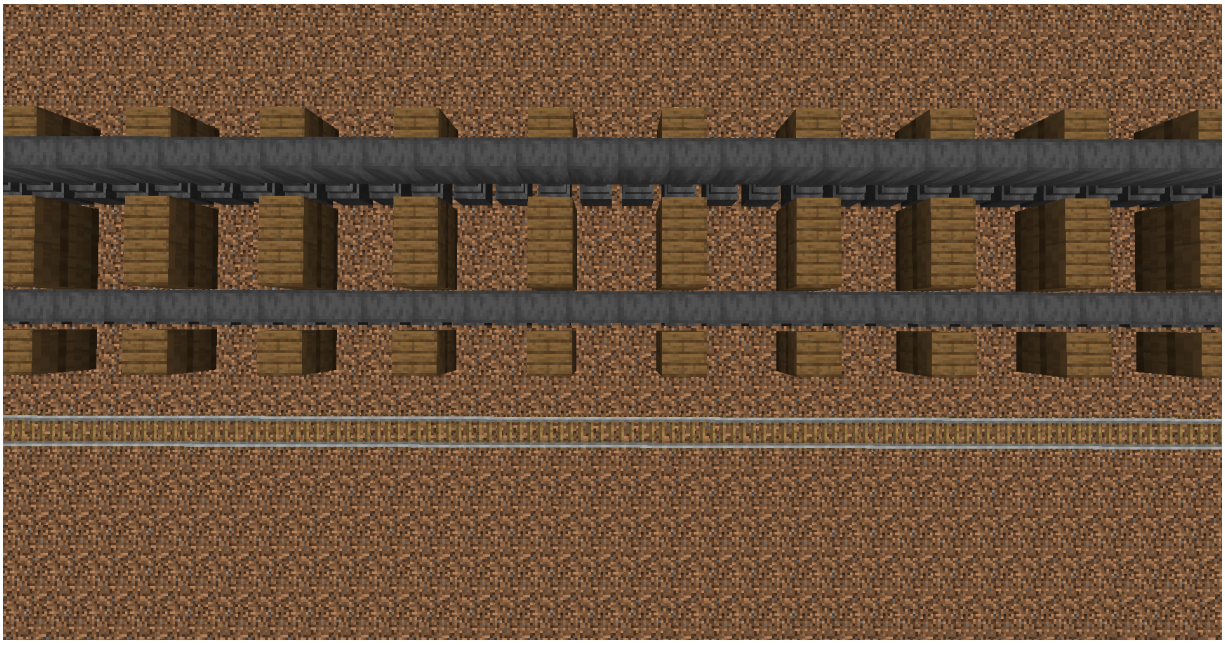

Abb. 3.3: Über dem Bild ist die geplante Darstellung zu sehen. Darunter befinden sich Schienen mit einer Breite von nur einem Meter

Dies führt dazu, dass eine Linie, die beispielsweise sehr nahe der x-Achse verläuft, den x-Wert häufig erhöht. Erst wenn die zurückgelegte Strecke zu hoch ist, wird auch der z-Wert erhöht, um zu verhindern, dass der Fortschritt zu niedrig bleibt.

Beim Algorithmus von Alice wird bei gleichen Fortschritten von x und z ein diagonaler Sprung ausgeführt und somit ein Block übersprungen. In unserem Fall ist es wichtig, dass selbst in diesem Fall ein Block dazwischen liegt, um die korrekte Platzierung der Schienen sicherzustellen. Es dürfen keine diagonalen Schritte erfolgen. Deshalb wird bei uns in jeder Iteration nur ein Achsentest gemacht.

Barrieren

In OpenStreetMap werden verschiedene Barrieren angegeben, die mithilfe dieses Programms bearbeitet werden können. Dazu gehören auch Hecken, die in Minecraft mit Blöcken dargestellt werden, die Blättern ähneln. Es existieren mehrere Typen von Barrieren, die in Minecraft mit Zäunen visualisiert werden. Falls der spezifische Typ nicht eindeutig ist, wird standardmäßig eine Kopfsteinmauer verwendet. Die genaue Positionierung der Barrieren erfolgt mithilfe des Bresenham-Algorithmus [22]. Dieser Algorithmus gewährleistet eine präzise Festlegung der Positionen der Barrieren in der Minecraft Welt.

3.3.4 2D Map

In diesem Kapitel geht es um die Speicherung der Minecraftblöcke, bevor sie in Minecraft-Dateien geschrieben werden. Angesichts der Speicherkomplexität, die mit der dritten Dimension einhergeht, hat es sich angebotenen eine effiziente Methode zur Verwaltung von Informationen in der Spielwelt zu entwickeln. Dieser Spagat wurde mit einem zweidimensionalen Array bewältigt, der auf zwei Listen referenziert, die festlegen, welche Blöcke über und unter der vorgegebenen Höhe liegen. Im Code ist dies mit einem eindimensionalen Array gelöst worden, da damit nur der Wert aus dem Speicher geholt werden muss. Bei einem zweiten Array muss erst der Speicherort dieses Arrays abgefragt werden und das dauert länger. In einer Welt kommt es nur selten vor, dass sich mehrere Dinge auf einem Punkt befinden. Somit hat es sich angeboten für jeden Meter, oder Block in der Minecraft Welt, ein Objekt zu speichern, das die aktuellen Informationen speichert. Da dieses Objekt eine Spalte beschreibt, haben wir es Column genannt [24]. An den meisten Punkten musste nur der Bereich gespeichert werden. Wenn auf einem Punkt ein Haus steht, können die relevanten Blöcke auch in diesem Objekt gespeichert werden.

So kann jedes OSM-Objekt durchiteriert werden und dann die Informationen der Welt, in den Column-Objekten speichern.

Diese zweidimensionale Respräsentation der Welt hat gleichzeitig auch den Vorteil, dass man den aktuellen Fortschritt in einem Bild darstellen kann.

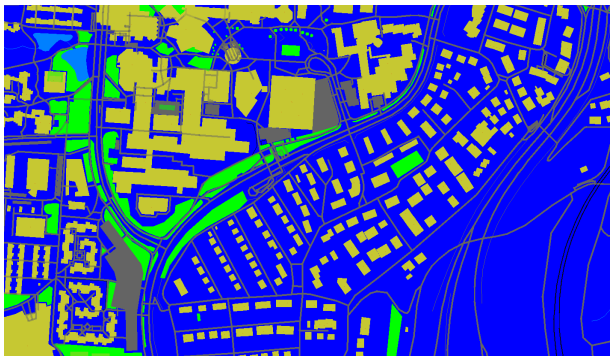

Abb. 3.4: Eine Visualisierung der 2D-Map durch das Programm

Vor allem am Anfang, als die Minecraftspeicherung noch nicht fertiggestellt war, konnte man die Effekte einer Codeänderung auf dem Bild 3.4 sehen.

3.3.5 Minecraft Speicher

Die Speicherung in Minecraft geht über das einfache Ablegen von Daten in Dateien hinaus; sie bildet das Grundgerüst der Wiedergabe, Gestaltung und Interaktion mit der virtuellen Umgebung. Von der strukturierten Organisation der Weltordner bis hin zur präzisen Definition von Spielerinformationen in der level.dat-Datei - die Umsetzung dieses Teils des Projekts spielt eine entscheidende Rolle für die erfolgreiche Integration in die Minecraft-Plattform.

Dafür geht es in dem nachfolgenden Unterkapitel um die Speicherung der Blockdaten.

Blockdaten

Für die Implementierung des Minecraft-Formats wurde die letzte Schnittstelle, bei der die Daten in ein DataOutputStream [25] geschrieben werden, aufgenommen. Diese Aufnahme konnte für unseren Zweck angepasst werden, sodass unsere Daten gespeichert werden und Minecraft sie erkennt. Minecraft speichert die Welt in sogenannten Regiondateien, wobei im ersten Megabyte die Position und Größe, der in der Datei befindlichen Chunks, gespeichert ist. Ein Chunk ist die nächste Unterstufe und besteht aus 16x16 Blöcken und umfasst die komplette Höhe. Die Position, die am Anfang der Regiondatei steht gibt dabei den Ort innerhalb der Datei an. Der darauf folgende Megabyte enthält den Zeitstempel, wann der Chunk zuletzt beschrieben wurde. Es steht zwar fest, wie viele Blöcke pro Chunk maximal gespeichert werden könnten, allerdings gibt es Blöcke, die mehr Speicher benötigen. Da wir in dieser Arbeit allerdings keine dieser speziellen Blöcke verwendet haben, können diese ignoriert werden. Wichtiger ist, dass Minecraft die BlockSequenzen aus einer Sektion komprimiert. Dabei werden 16x16x16 Blöcke zusammengefasst. Wenn innerhalb dieses Bereichs nur ein Blocktyp vorhanden ist, wird nur dieser gespeichert. Andernfalls werden die Blocktypen in einer Palette abgelegt und in einem long[] die Indizes dieser Palette referenziert. Ein long[] kann man sich wie eine Liste mit großen Zahlen vorstellen.

Diese Optimierung ermöglicht nicht nur eine effizientere Speicherung, sondern trägt auch dazu bei, den Ressourcenverbrauch zu reduzieren und die Ladezeiten beim Lesen und Schreiben von Weltinformationen zu verbessern. Dieser Aspekt der Minecraft-Implementation unterstreicht die Bedeutung von Datenkomprimierung und effizienter Speicherverwaltung in der Softwareentwicklung.

level.dat

Jede Minecraft Welt ist durch einen Ordner repräsentiert, der sämtliche weltrelevante Daten enthält. Zur Identifikation und Anerkennung dieser Welt durch Minecraft wird eine level.dat-Datei benötigt. Diese Datei enthält wesentliche Informationen wie den Namen der Welt und Spielerdetails, einschließlich aktueller Koordinaten.

Im Ausgabeergebnis des Programms ist daher die Generierung dieser level.dat-Datei ein essenzieller Bestandteil. Da bestimmte Einträge in dieser Datei häufig konstant sind, wurden die entsprechenden Werte festgeschrieben. Dies gewährleistet eine reibungslose Integration der konvertierten Welt in Minecraft, indem die notwendigen Informationen korrekt und vorab definiert bereitgestellt werden.

Datapack

Um die Welthöhe in Minecraft zu ändern und die minimale Höhenkoordinate anzupassen, müssen wir eine Zip-Datei erstellen, die die benötigten Weltinformationen enthält. Diese Zip-Datei wird dann in die Minecraft Welt integriert. Die meisten Daten bleiben davon gleich und wir müssen nur die Höhe editieren.

3.3.6 Webserver

Um die Konvertierung von OpenStreetMap-Daten in Minecraft für eine breite Nutzerbasis zugänglich zu machen, wird ein Webserver eingerichtet. Dies ermöglicht es den Benutzern, einen Bereich ohne den mühsamen Prozess der manuellen Installation der benötigten Bibliotheken umzuwandeln. Der Webserver stellt eine Schnittstelle bereit, über den die Nutzer einfach einen Bereich auswählen können, um ihn in Minecraft zu konvertieren.

Die Verwendung eines Webservers erleichtert auch die Aktualisierung und Wartung der Konvertierungssoftware. Neue Versionen können zentral auf dem Server bereitgestellt werden, sodass Benutzer immer auf die neuesten Funktionen zugreifen können, ohne ihre eigenen Installationen aktualisieren zu müssen.

Durch die Bereitstellung eines Webdienstes wird die Konvertierung von OpenStreetMap-Daten in Minecraft für eine Vielzahl von Benutzern zugänglich, unabhängig von ihrem technischen Hintergrund oder ihren Fähigkeiten in der Systemadministration.

3.4 Installation

Um mit der Konvertierung von OpenStreetMap-Daten in Minecraft zu beginnen, müssen zunächst einige Softwarekomponenten installiert werden.

Über den Linux Befehl „apt get install osmium“ beginnt die Installation von Osmium.

Da das Programm mit Hilfe von Java [26] geschrieben wurde, muss Java auf dem System installiert sein (Java Version 17 oder neuer sollte funktionieren).

Nach erfolgreicher Installation der beiden Komponenten Osmium und Java, kann der Konvertierungsprozess starten. Hierzu muss eine .osm.pbf-Datei von Geofabrik heruntergeladen werden [27].

Auf der OpenStreetMap-Exportseite gibt es eine Beschränkung für die Anzahl der Objekte in einem Bereich. Die Eingabe wird direkt umgewandelt, und eine Zip-Datei wird ausgegeben. Bei einer .osm.pbf-Datei wird ein Webserver auf Port 80 gestartet und der Anwender kann einen Ausschnitt auswählen, der aus der Datei extrahiert und heruntergeladen wird.

Mit dem folgenden Befehl, wird das Programm gestartet:

„java -jar <JARDatei>.jar -f <OSMPFBDatei>“

Das Programm bietet die Möglichkeit, den Port mit dem Parameter -p anzupassen. Bei Bedarf kann auch durch die Angabe von -h eine Höhenverschiebung in Form eines Integers erfolgen.

Standardmäßig erwartet das Programm eine Datei namens karte.html am Ausführungsort, zusammen mit einem lib-Ordner, der für die Website benötigt wird. Falls eine andere Position gewünscht ist, kann der Pfad mithilfe von -a geändert werden.

Während der Ausführung wird zunächst eine data.osm-Datei generiert. Dies ist das Ergebnis der Osmium-Ausgabe und wird am Ende des Prozesses wieder gelöscht. Die Position dieses Ordners kann ebenfalls durch die Verwendung von -d modifiziert werden. Die Höhendaten werden in diesem Ordner zwischengespeichert und nicht gelöscht.

Wenn Nutzer die Website besuchen, können sie mit zwei Klicks die beiden Eckpunkte eines Abschnitts festlegen. Dieser Abschnitt wird dann aus der angegebenen .osm.pbf-Datei ausgelesen und umgewandelt. Nach Fertigstellung wird die generierte Zip-Datei heruntergeladen.

In Minecraft gelangt man über Optionen, „Resource Packs“ und dann „Open Pack Folder“ in das Verzeichnis, in dem Minecraft seine Daten speichert. In „.minecraft/saves“ kann nun der world.zip Inhalt entpackt werden. Wenn alles übertragen wurde, gelangt man in Minecraft über „Singleplayer“ und dann mit Doppelklick auf „world“ in die Welt. Eventuell „spawned“ man an einer Felswand direkt neben der Welt, das heißt, man erscheint an der

Stelle. Die Minecraftdaten wurden für die Version 1.19.4 geschrieben, können aber auch mit den nachfolgenden Versionen geöffnet werden.

3.5 Limitationen

OSM an sich ist nicht vollständig und mit einem Datenbestand, der meist älter als fünf Jahre ist, nicht immer aktuell. Fehlende Daten können von unserem Programm nicht ergänzt werden.

Selbst bei existierenden Daten fehlen Informationen. Zum Beispiel werden Häuser nur als Grundriss gespeichert.

Wenn dann auch noch der Höhentag nicht ausgefüllt ist, ist es für ein Programm schwierig zu erkennen, welche Art von Gebäude sich an dieser Stelle befinden soll.

Fehlende Daten kommen in großen Städten sehr selten vor. In abgelegenen Orten sind Häuser meist nur als „building=yes“ gekennzeichnet.

Noch auffälliger sind die Limitationen von Minecraft. Da Minecrafts Grundeinheit aus 1x1x1 Meter großen Blöcken besteht, sind auch alle Landschaftsmerkmale darauf angewiesen diese Größe zu haben. Es gibt in Minecraft zwar auch Blöcke, die kleiner als ein Meter sind, dennoch können nicht mehrere dieser Blöcke auf einer Position stehen. Somit kommt man nicht umhin Ecken zum Beispiel in schrägen Hauswänden zu akzeptieren.

Die maximale Bauhöhe ist in Minecraft auf 384 Blockhöhen beschränkt. Für dieses Projekt konnte sie zwar erweitert werden, ist aber immer noch auf 2048 Blöcke beschränkt.

Selten wird in den Tags das Material angegeben, weshalb das Programm diese fehlende Information ausgleichen muss. Ebenso können Angaben zu Höhe oder Aussehen ungenügend sein. Ein sehr großer Teil der Häuser beinhaltet keine Informationen zu den Fassaden. Diese werden einheitlich dargestellt. Vor allem in Regionen von unbekannteren Städten gibt es keine Angaben zu Gärten.

3.5.1 Exceptions

In Java wird Fehlerhandling mit Exceptions gelöst. Um Fehler besser identifizieren zu können, kann man so genannte try-catch-Blöcke setzen. Fehler, die innerhalb dieses Blockes auftreten, können in dem catch-Block behandelt werden.

Beim Einlesen jeglicher Dateien kann es passieren, dass Berechtigungen fehlen, oder sogar dass die Datei nicht existiert. Damit diese Ausnahmen vom Programm behandelt werden können, gibt es in Java Exceptions. Diese stoppen die Codeausführung und springen zu der letzten Codestelle, um den ein „try-catch“ block implementiert wurde.

In unserem Programm wird die OSM Datei eingelesen und mindestens eine Höhendatendatei. Falls diese Höhendaten noch nicht existieren, werden sie von einer externen Website heruntergeladen. Dabei kann der Download fehlschlagen. Somit kommen hier Exceptions zum Einsatz, die behandelt werden müssen.

Da die Höhendaten nicht essenziell für die Minecraft Welt sind, sondern eher ein nice-to-have, setzt das Programm in diesem Fall alle Höhendaten auf Null und gibt eine Warnung aus. In schlimmeren Fällen muss das Programm komplett gestoppt werden. Zum Beispiel wenn eine OSM-Datei nicht eingelesen werden konnte.

Um sicher zustellen, dass alles richtig läuft, überprüft das Programm auch Dateiinhalte, die nicht verwendet werden. So können Abweichungen frühzeitig erkannt werden, um den Nutzer zu warnen. Da die Änderungen an sich keine Probleme verursachen sollten, versucht das Programm normal weiter zu arbeiten.

3.5.2 Validierung und Testing

Um sicherzustellen, dass keine Daten in der OSM Datei verloren gehen, werden alle Nodes, AttributeMaps und NodeList der Datei durchiteriert. Wenn eine OSM Datei syntaktisch falsch ist, werden die resultierenden Unstimmigkeiten behandelt. Im debug modus wird eine Exception ausgegeben, die dazu führt, dass das Programm beendet wird und der Entwickler den Fehler beheben kann. Im normalen Modus für Nutzer wird die Unstimmigkeit in die Kommandozeile ausgegeben und das Programm versucht mit den restlichen Daten fortzusetzen. Je nach Fehlermeldung, kann es passieren, dass die Datei nicht komplett eingelesen wurde. Mit den bereits empfangenen Daten kann weitergearbeitet werden.

3.6 Evaluation

Während des Entwicklungsprozesses stellte sich heraus, dass das Programm gelegentlich längere Laufzeiten aufwies als zunächst angenommen. In solchen Situationen erwies es sich als äußerst hilfreich, einen dedizierten Thread zu implementieren, dessen alleinige Aufgabe darin bestand, den Stack-Trace jedes Threads auszugeben, wenn eine Aufgabe mehr Zeit beanspruchte als erwartet.

Durch diese Maßnahme konnte genau ermittelt werden, in welcher Routine sich der betroffene Thread zu einem bestimmten Zeitpunkt befand. Diese Transparenz ermöglichte eine präzisere Identifizierung potenzieller Probleme oder ineffizienter Prozesse innerhalb des Programms. Anhand dieser Erkenntnisse war es möglich, gezielte Verbesserungsmaßnahmen zu ergreifen und die Leistungsfähigkeit sowie die Stabilität des Programms signifikant zu steigern.

3.6.1 Profiler

Um ausschlaggebende Statistiken geben zu können, wird ein Programmteil benötigt, der misst, wie lange die einzelnen Programmsektionen gebraucht haben. Außerdem zählt er wie oft bestimme Codeabschnitte ausgeführt wurden. Auch der Speicher wurde analysiert. Während des Tests in einem umfangreichen Bereich wurden oft StackTraces gesendet, um festzustellen, ob Blöcke Teil des Bereichs sind oder nicht. Wie in der späteren Evaluation zu sehen ist, traten häufig natürliche Objekte wie Wasser oder Äcker auf. In solchen Fällen war es äußerst hilfreich, den Fortschritt des Programms zu überwachen und zu erkennen, wenn beispielsweise ein Fluss bearbeitet wurde und nun mit der Bearbeitung eines anderen Objekts begonnen wurde. Die Verwendung von StackTraces half dabei, Fehler zu identifizieren und den Status der Konvertierung zu verfolgen, insbesondere in großen und komplexen Umgebungen.

4 Ergebnisse

Die Konvertierung von OpenStreetMap-Daten in Minecraft erfordert eine gründliche Analyse und Verarbeitung verschiedener Datensätze, um die Genauigkeit und Effizienz des Konvertierungsprozesses zu gewährleisten. In diesem Abschnitt werden die verwendeten Datensätze vorgestellt, die als Grundlage für die Konvertierung dienen.

Zunächst wird eine Tabelle mit den Größen der einzelnen Datensätze präsentiert, um einen Überblick über deren Umfang und Komplexität zu geben. Anschließend werden die einzelnen Datensätze detailliert beschrieben, wobei ihre Struktur, Inhalte und Besonderheiten beleuchtet werden.

Die Analyse und Bewertung der Datensätze ermöglicht es, potenzielle Herausforderungen und Probleme bei der Konvertierung zu identifizieren und entsprechende Lösungsstrategien zu entwickeln. Darüber hinaus wird die Leistung der Konvertierungsalgorithmen anhand der verschiedenen Datensätze bewertet und miteinander verglichen, um Erkenntnisse über ihre Effizienz und Skalierbarkeit zu gewinnen.

Dieser Abschnitt bietet somit einen umfassenden Einblick in die Vielfalt und Komplexität der OpenStreetMap-Daten und deren Bedeutung für die Konvertierung in das Minecraft-Format.

4.1 Großer Trango Turm

Der Trango Turm in Pakistan ist berühmt für seine markanten, steilen Wände und den angrenzenden Gletscher auf seiner Nordseite. Dieser Gletscher trägt zur einzigartigen und herausfordernden Natur des Berges bei, der unter Bergsteigern weltweit bekannt ist. Auf dem Bild 4.1 sieht man das Ergebniss des Programmes in Minecraft. Es ist die Spitze des größten Trango Turms zu sehen. Im Vergleich dazu sind im Bild 4.2 die Trango Türme in echt zu sehen.

Durch die weit auseinanderliegenden Höhendaten fällt die Steigung weniger auf. Dennoch wird deutlich, wie die Interpolation der Höhendaten dazu beiträgt, eine glattere Darstellung zu erzeugen. Diese Glättung, auch bekannt als „Smoothing“, ist ein Prozess, bei dem die Unterschiede zwischen den einzelnen Höhenwerten reduziert werden, um eine kontinuierliche und ästhetisch ansprechendere Darstellung zu erreichen.

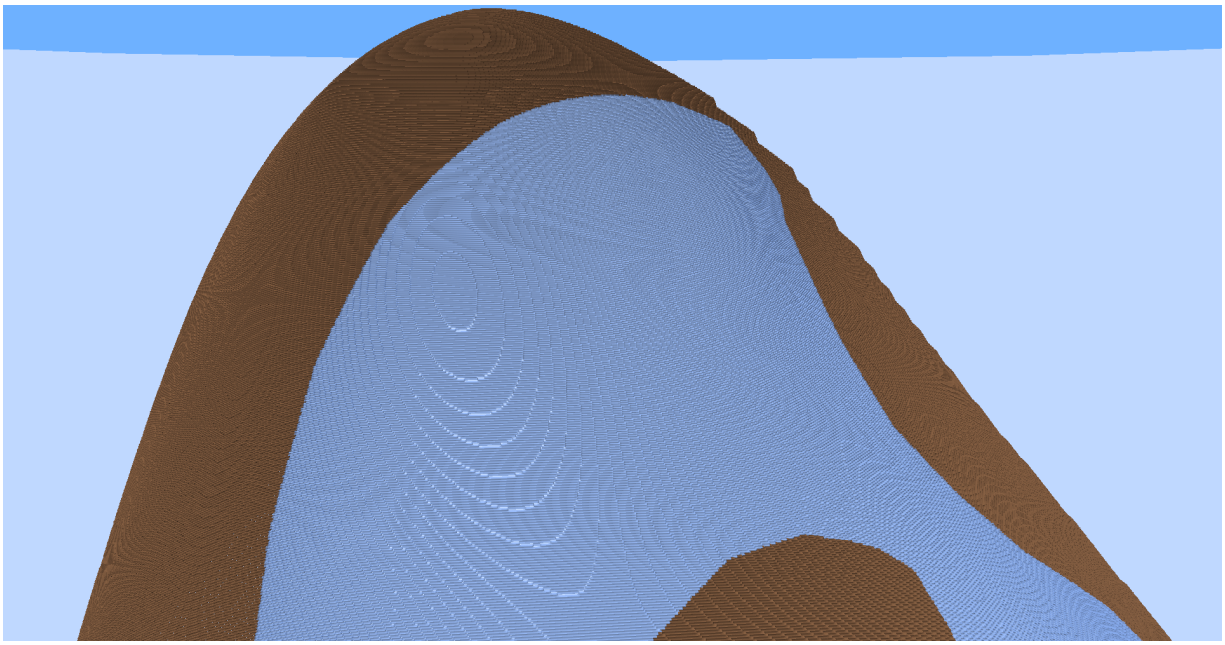

Abb. 4.1: Der Gletscher nördlich des Großen Trango Turms

Abb. 4.2: Wie die großen Trango Türme in echt aussehen

Abb. 4.3: Der große Trango Turm in der OSM Karte

Wie man in dem Bild 4.3 sehen kann, gibt es nur wenige Einträge in den OSM-Daten, die definieren wie die Landschaft aussieht. So ist es nicht überraschend, dass das Einlesen der Daten nur 152 ms benötigt hat. Insgesamt kann man in den Daten nur 52 Objekte finden, die zusammen 6ms verbraucht haben, um herauszufinden wie sie dargestellt werden sollen. In Minecraft lässt sich nur der Gletscher darstellen. Die Berechnung dauerte 704 ms.

Es ist wichtig zu erwähnen, dass eine Einstellung hinzugefügt werden musste, die die Höhe heruntersetzt, weil man die Maximalhöhe in Minecraft nicht größer als 2048 stellen kann. Diese Einstellung kann man über den Parameter -h beim ausführen der jar-Datei vornehmen. Diese Änderung wird nur in Gebirgen, die sehr hoch liegen, benötigt. Fast alle Städte liegen nah der Meereshöhe, sodass es sinnvoll ist den Standarttiefstpunkt bei 0 beizubehalten.

Eine Integration zusätzlicher Höhendatenquellen mit höherer Auflösung und Genauigkeit könnte zu einer realistischeren Darstellung der Geländeformen beitragen.

4.2 Stuttgart

Abbildung 4.4 zeigt die südliche Seite von Stuttgart mit Blick auf die Schienen einer Zahnradbahn, im Volksmund als Zacke bekannt. Die Schienen dieser Bahn sieht man links im Bild nach oben verlaufend. Auf der rechten Seite sind Wohngebäude. Überall, wo in den OSM Daten Wiesen angegeben werden, ist Gras in Minecraft vorhanden. Wenn nichts eingestellt ist, kann auch Erde dargestellt sein.

Im darauffolgenden Bild 4.5 ist ein Abschnitt zu sehen, der näher an der Stadtmitte von Stuttgart liegt. Er markiert den Ausgangspunkt der Zacke. Die Visualisierung umfasst Häuser, Straßen sowie eine U-Bahn, die sich im unteren Bereich des Bildes 4.5 befinden. Zusätzlich werden auch Bäume und Gehwege modelliert, was eine detailgetreue Darstellung der städtischen Umgebung ermöglicht.

Im Bild 4.6 ist der Stadtpark zu sehen. Die Bäume und das Wasser sind dargestellt. Beim Straßenverlauf kann man an der dreieckigen Barriere erkennen, dass die Wegbreite aufgrund der Datenlage geschätzt werden musste. An der linken Seite reicht die Straße nicht nahe genug an die Barriere heran, an der oberen Seite ist die Straße zu breit.

Die nächsten beiden Bilder zeigen das Europaviertel von Stuttgart aus ähnlicher Perspektive. Das eine Bild 4.7 in Minecraft und das andere in der Realität: Abb. 4.8. Das Minecraft-Bild 4.7 zeigt einen größeren Ausschnitt. Das Gebäude im Bild unten Mitte ist die Stadtbibliothek, von deren Dach, stark gezoomt, das Bild aus 4.8 aufgenommen wurde. An diesem Beispiel lässt sich sehr gut zeigen, welche Herausforderungen die Anwendung zu nehmen hat und wie das Ergebnis in Minecraft vom Datenmaterial aus OSM abhängt.

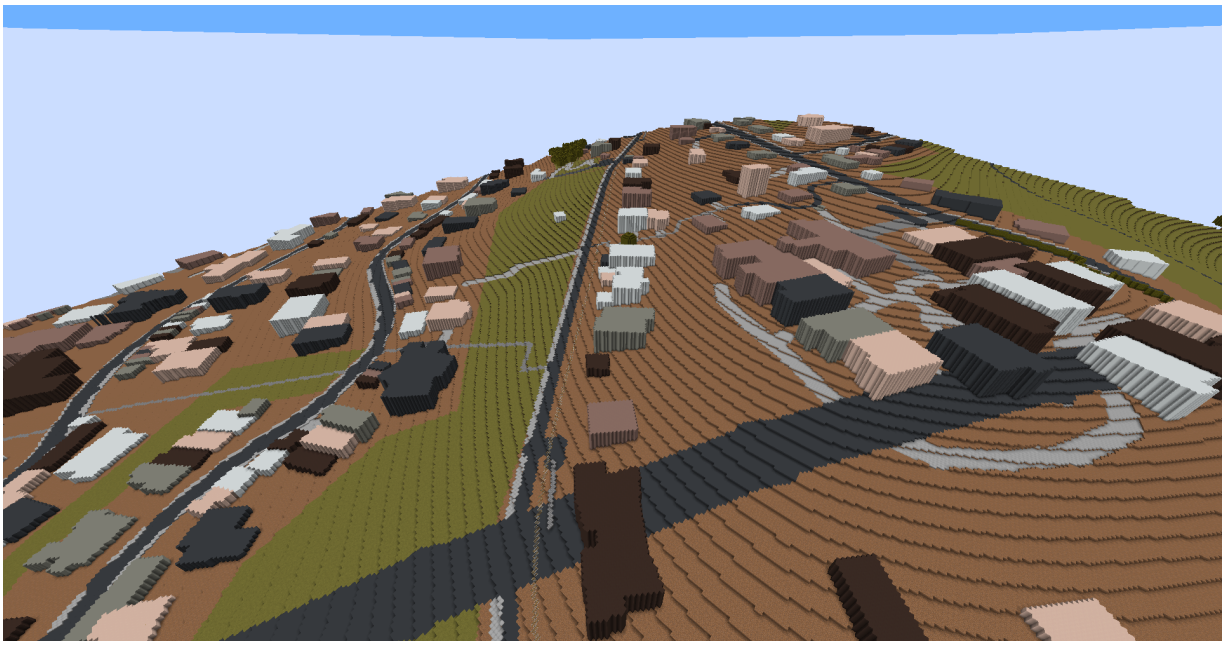

Abb. 4.4: Blick auf die Zahnradbahn/Zacke

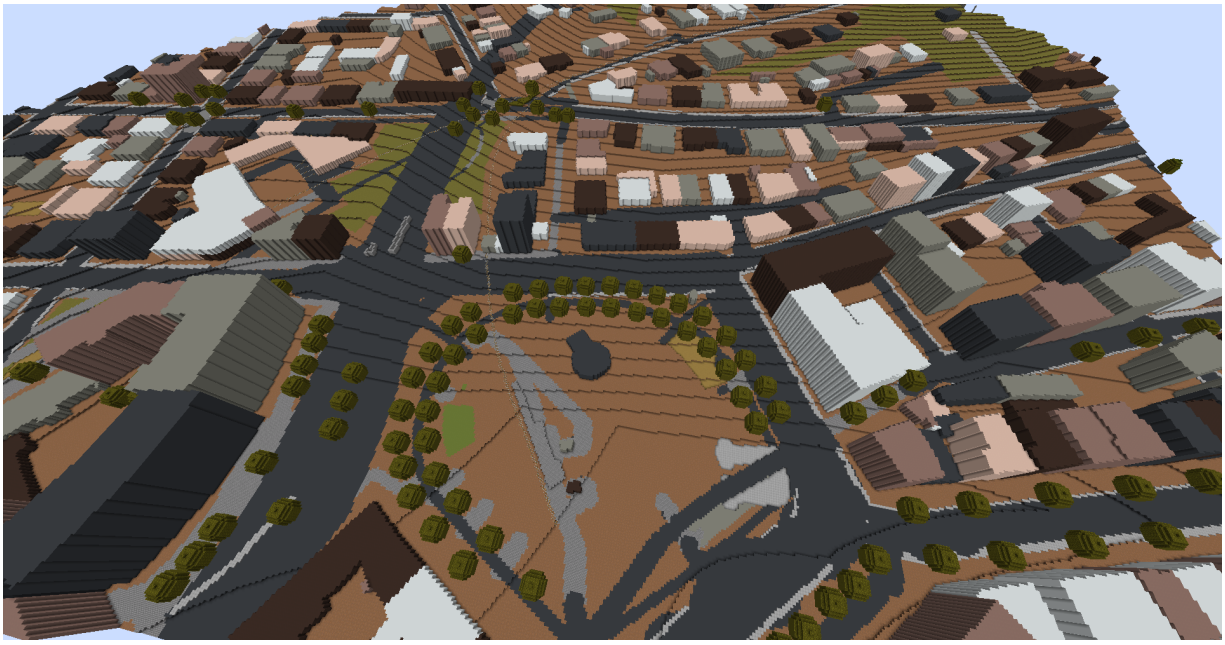

Abb. 4.5: Marienplatz, Stuttgart mit der Starthaltestelle der Zahnradbahn

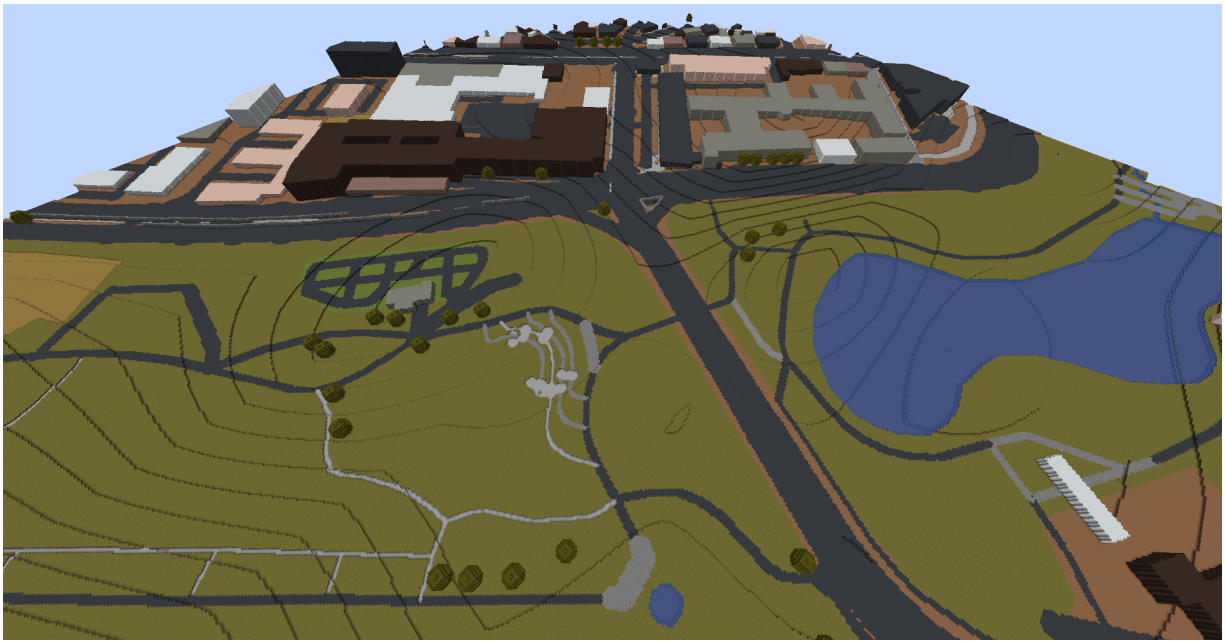

Abb. 4.6: Stadtpark in Stuttgart

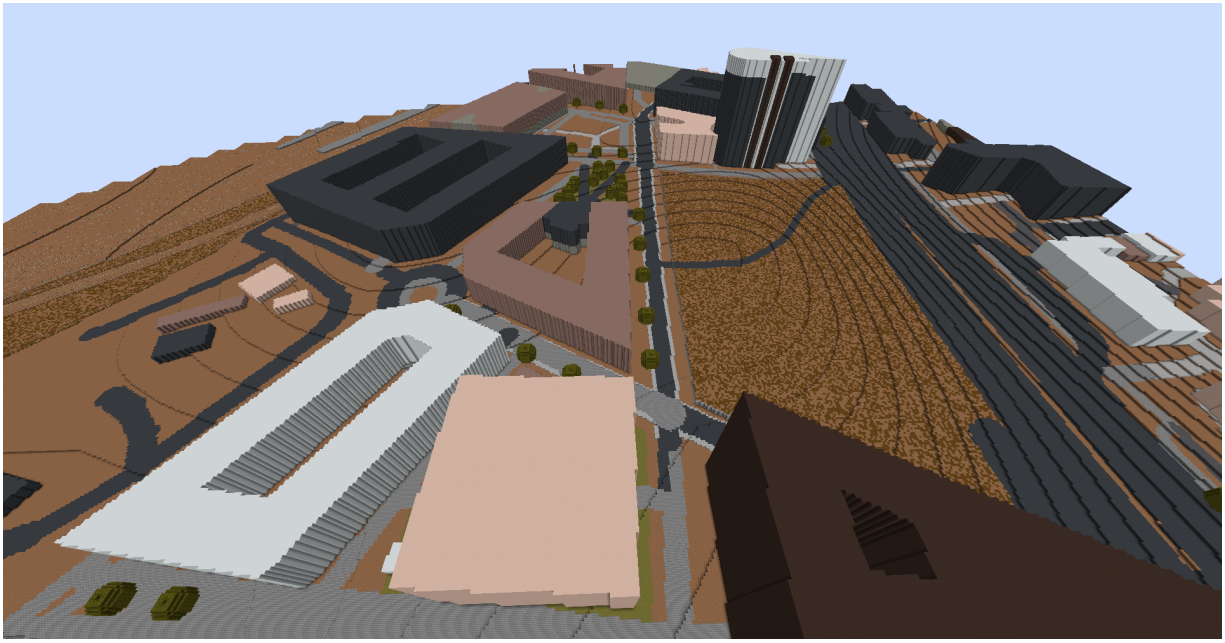

Abb. 4.7: Europaviertel

Abb. 4.8: Die Aussicht vom Dach der Stadtbibliothek auf das Europaviertel[1]

Gut dargestellt sind, wie im Beispiel Stadtpark, die Bäume. Ebenso ist der Turm der LBBW-Bank gut erkennbar. Die gebogene Form sowie die spitzzulaufende Gebäudewand sind erkennbar.

Nicht so sauber ist das Verbindungselement der beiden zulaufenden Gebäudespangen im vorderen Bereich der Fotografie. In der Realität gibt es einen Höhenunterschied von zwei Stockwerken, zudem ist die Verbindung mit Bäumen bepflanzt. Es gibt sogar einen schmalen, unüberdachten Weg, der die Verbindung unterbricht.

In der Mitte, weit im Hintergrund der Fotografie sieht man ein mehrstöckiges Gebäude. Eine breite Dachplatte verbindet es mit dem aus Betrachtersicht rechts befindlichen Gebäude. In OSM ist die Dachverbindung als Stockwerk hinterlegt. Somit sieht sie in Minecraft wie ein Teil des Gebäudes aus.

Außerhalb des Fotoausschnitts sind im Minecraftbild linkerhand die Gleise zu sehen. Auf der rechten Seite des Minecraftbildes verläuft die B27. Links davon befindet sich eine Baustelle.

Das Bild 4.9 zeigt die Baumbibliothek in Minecraft, die im Grundlagenkapitel erwähnt wurde. In Minecraft ist dies die beste Möglichkeit, um diese Bibliothek darzustellen.

Betrachten wir das Informatikgebäude, das bereits in den Grundlagen (bei Bild 2.5) erwähnt wurde. Auf dem Bild 4.10 sehen Sie das Gebäude sowie weitere Teile der Universität im Hintergrund. Zum Beispiel den sogenannten „Roten Platz“ [28], der hier in hellgrün dargestellt ist, da er als Sportfläche eingetragen ist. Interessanterweise ist er in Echt und auf der OpenStreetMap-Seite [2] ebenfalls grün und nicht rot.

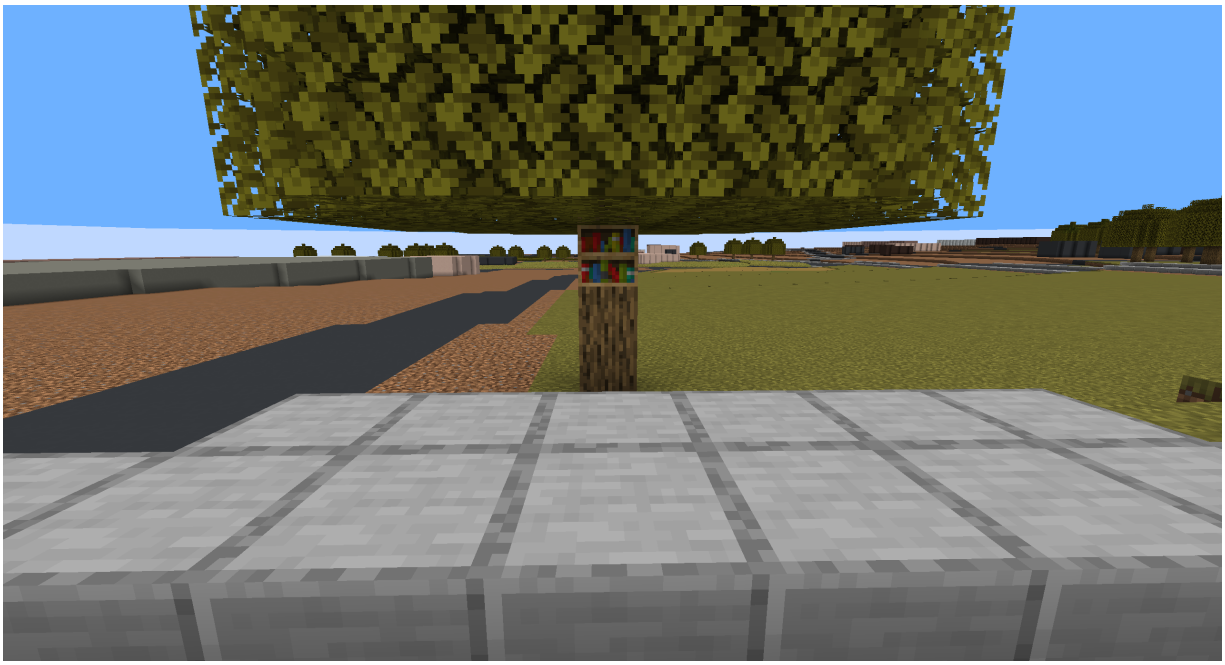

Abb. 4.9: Die Baumbibliothek in Minecraft

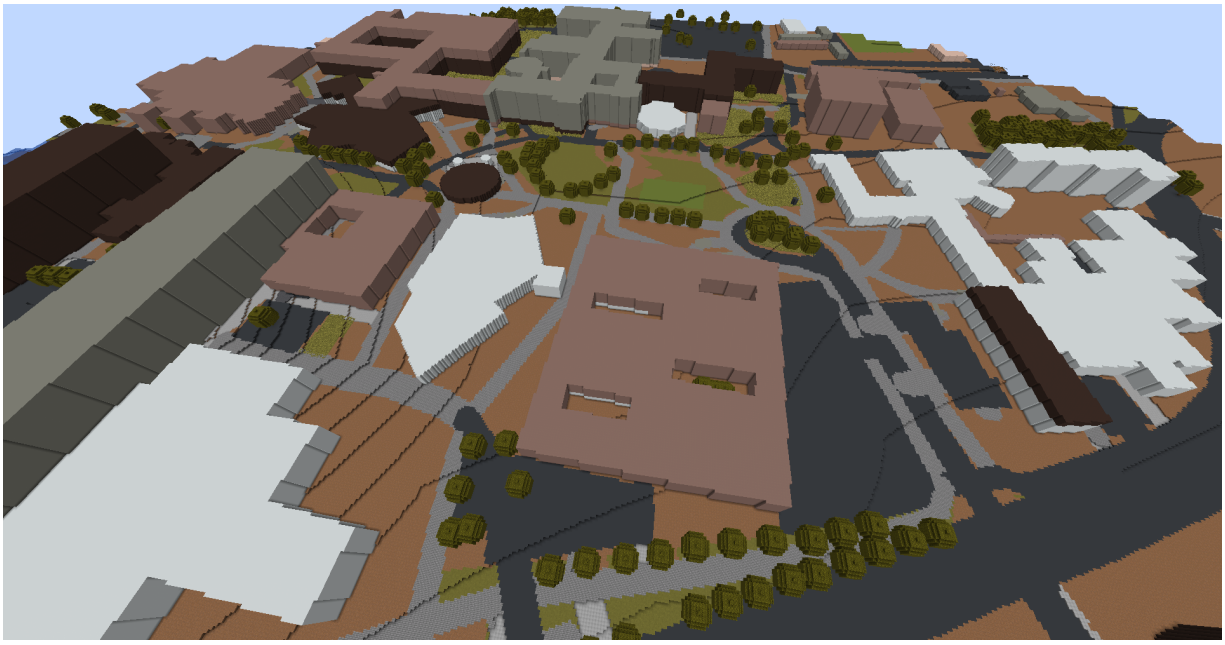

Abb. 4.10: Blick auf das Informatikgebäude der Universität Stuttgart

Vor und neben dem Informatikgebäude sind Parkplätze in Grau markiert, während die Straßen aus einem asphaltähnlichen Block bestehen. Gehwege sind heller grau.

4.3 Berlin

Im Bild 4.11 vom Alexanderplatz in Berlin ist unten rechts ein Teil des Fernsehturms zu sehen. Das pavillonartige Gebäude am Fuße des Fernsehturms ist ebenfalls erkennbar, ebenso wie die beiden Wasserflächen daneben. Das kleine Gebäude rechts des Wassers stellt die Marienkirche dar. Die Parkanlage ist von mehreren Bäumen gesäumt. Auf der linken Seite des Bildes befindet sich ein brauner Bereich, der eine Baustelle darstellt.

4.4 Ländlichere Region

Auf dem Bild 4.12 sieht man in einer ländlicheren Region, in der nähe von Wernau, Felder mit Traktorspuren. Die Äcker wurden in Minecraft mit dem passenden Block dargestellt, auf dem man in Minecraft auch Dinge anbauen kann.

Abb. 4.11: Alexanderplatz in Berlin

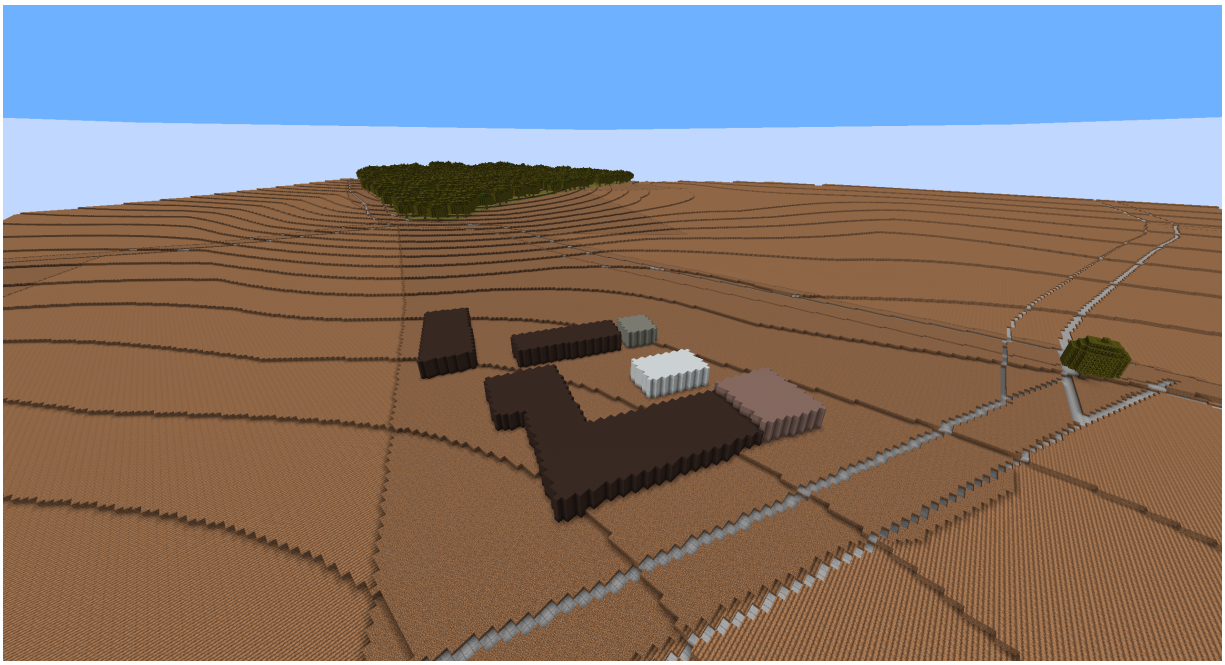

Abb. 4.12: Blick auf ein Bauernhof Richtung Wald in Minecraft

4.5 Performance

Eine entscheidende Metrik für Programme ist ihre endgültige Laufzeit, die stark von der Größe der Auswahl abhängt. Dies wird in den folgenden Tabelle 4.1 dargestellt.

In der Tabelle 4.1 sind verschiedene Datensätze aufgelistet, zusammen mit ihrer Größe in Metern und der Anzahl der Objekte, die sie enthalten. Die Bilder im vorherigen Kapitel stammen ebenfalls aus diesen Datensätzen.

Diese Daten dienen als Grundlage für die nachfolgende Analyse. Der Marienplatz ist der einzige Datensatz, der länger als breit ist, da in diesem Datensatz die Höhe wichtig ist und es Richtung Süden den Berg hinauf geht.

In der Tabelle 4.2 ist die Zeit für die einzelnen Schritte aufgeführt. Die Initialisierung am Anfang kann vernachlässigt werden, da sie lediglich den Profiler und andere Features einstellt, die das Programmieren erleichtern. Das Laden der Daten ist jedoch viel interessanter und hängt stark davon ab, wie viele Daten im Datensatz enthalten sind. Dieser Unterschied ist gut zwischen dem Trango Turm und den Stuttgart Datensätzen erkennbar.

Die Konvertierung von Osmium wurde nicht verfolgt. Bei der Objektfilterung musste durch alle Objekte iteriert werden, um herauszufinden, welche Datenobjekte für die Konvertierung relevant sind. Dieser Schritt hängt auch stark von der Anzahl der Objekte ab.

Die Höhengenerierung kann auf den ersten Blick so aussehen, als käme die Dauer von der Polynominterpolation, aber die lineare Interpolation dauert in etwa genauso lange, wenn nicht sogar länger. Es ist zu bemerken, dass die Höhendaten in diesem Schritt heruntergeladen oder aus dem Speicher geladen werden müssen. Dies könnte der Grund für den Performanceeinfluss sein. In dem Weltspeicher wird die „zweidimensionale Karte“ erstellt

Datensatz	Breite in m	Länge in m	Objekt-Anzahl	verwendete Objekte
Großer Trango Turm	1872	1072	52	4
Stuttgart Marien-platz	1248	2976	22949	4422
Stuttgart Haupt-bahnhof	2640	1504	26852	6377
Wernau	5664	3280	33205	5401
Uni	5280	3024	53878	11069
Berlin	2832	1632	34069	8258

Tab. 4.1: Größe der Datensätze und wie viele OSM-Objekte

Datensatz	Init	Datei-laden in ms	Objekt-filterung in ms	Welt in ms	Höhen-generierung in ms	Welt-speicher in ms	Objekte schreiben in ms	Speicherung in ms
Großer Trango Turm	11	168	72	12	5998	445	725	25120
Stuttgart Marienplatz	16	4829	1223	24	10473	1397	9164	21148
Stuttgart Hauptbahnhof	17	3346	1218	16	9879	1417	33269	16278
Wernau	11	3700	811	19	6105	4862	314179	50980
Uni	10	2417	969	10	9369	6503	62445	85494
Berlin	11	2075	532	11	5059	725	29513	14703

Tab. 4.2: Wie lange die einzelnen Schritte der Konvertierung gedauert haben

und der Speicher zusammen mit den Höhendaten vorbereitet. Die Dauer hängt also stark von der Größe des zu beschreibenden Gebiets ab.

Der wichtigste Schritt ist der, in dem die Objekte in die Welt geschrieben werden. Später wird genauer auf die Dauer der einzelnen Objekte eingegangen. Die Zahlen in der Tabelle zeigen die Zeit, die alle Objekte zusammen benötigt haben.

Es ist zu erwähnen, dass ländliche Gebiete wie Wernau mit den vielen Feldern und dem Fluss länger dauern, weil sie große Objekte und eine große Fläche umfassen.

4.5.1 Performance einzelner Objekte

Die folgenden Grafiken zeigen die Auswirkungen der Objektgröße auf die Umwandlungszeit verschiedener Objekte in Vaihingen. Die x-Achse stellt die Länger in Metern oder die Bounding Box Größe dar, in der sich das Objekt befindet. Auf der y-Achse ist die Umwandlungszeit in Millisekunden ablesbar.

Die Grafik 4.13 illustriert wie schnell Straßen umgewandelt werden konnten. Auf der x-Achse sind die Längen der Objekte dargestellt. Bis auf zwei Ausreißer konnten die Wege in wenigen Millisekunden aufgebaut werden.

Am Beispiel der Leisure-Kategorie lässt sich deutlich erkennen, wie sich die Größe der Objekte auf die Umwandlungszeit auswirkt.

Insbesondere bei Wasser- oder Waldflächen, wie sie in der Grafik 4.14 und 4.16 zu sehen sind, können die Objekte sehr groß sein, was zu längeren Umwandlungszeiten führt.

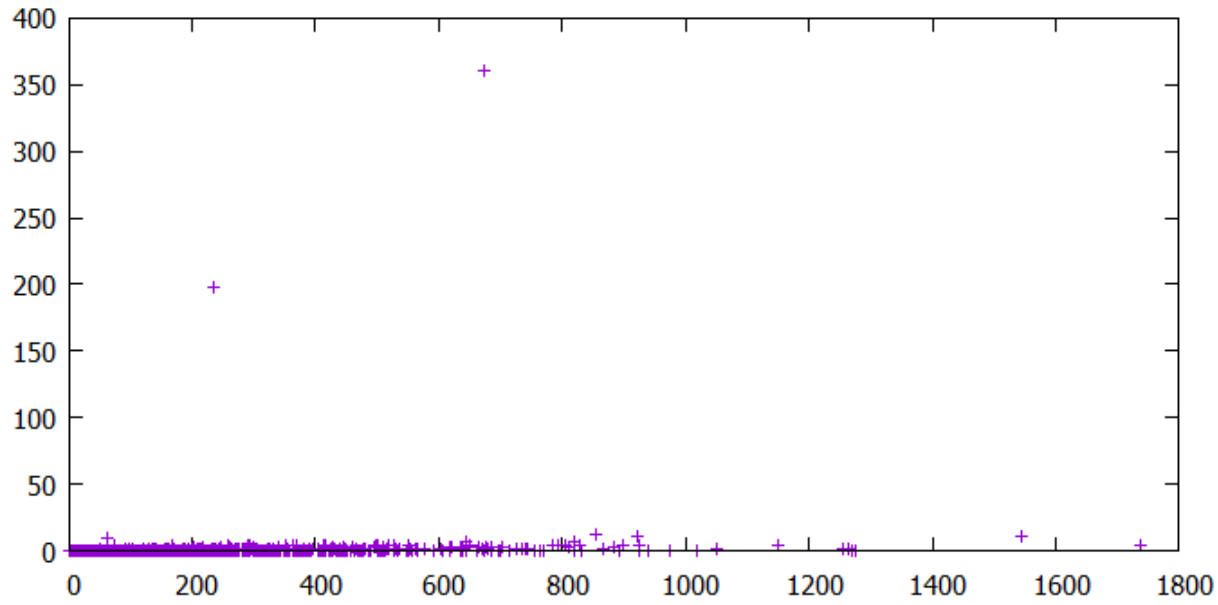

Abb. 4.13: Straße und Wege

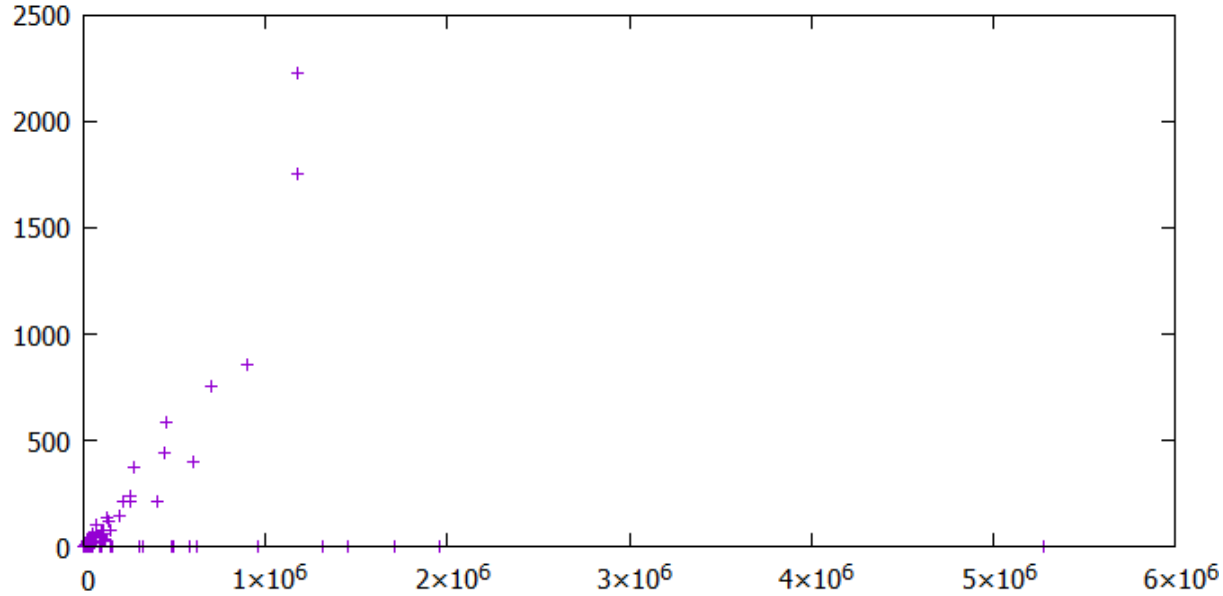

Abb. 4.14: Baustellen, Wiesen, Äcker, Wälder

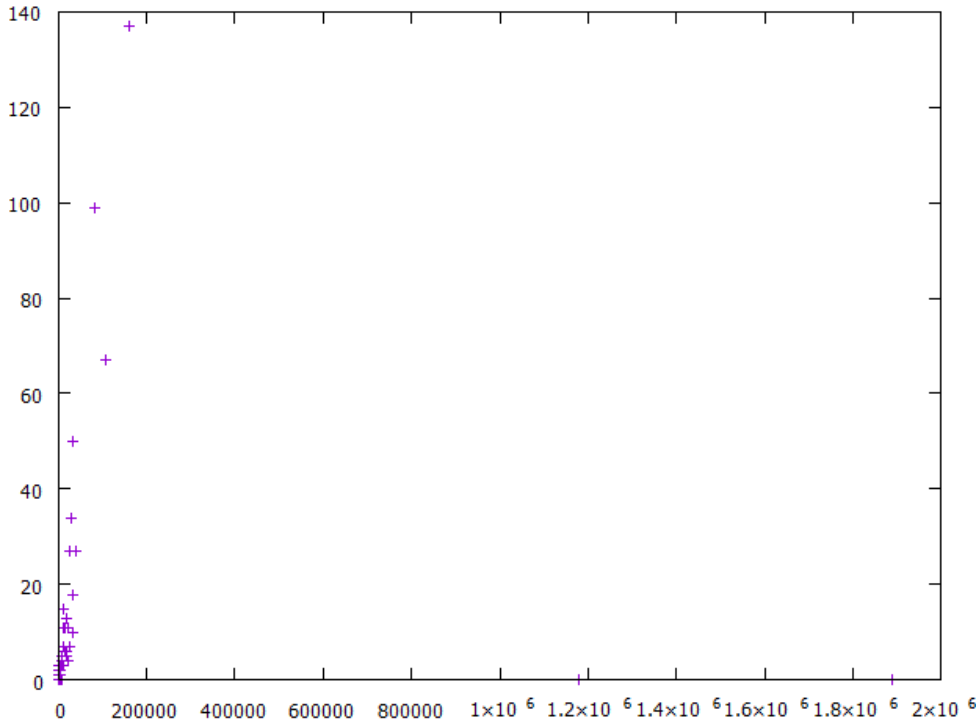

Abb. 4.15: Gärten, Sportplätze, Spielplätze, Parks

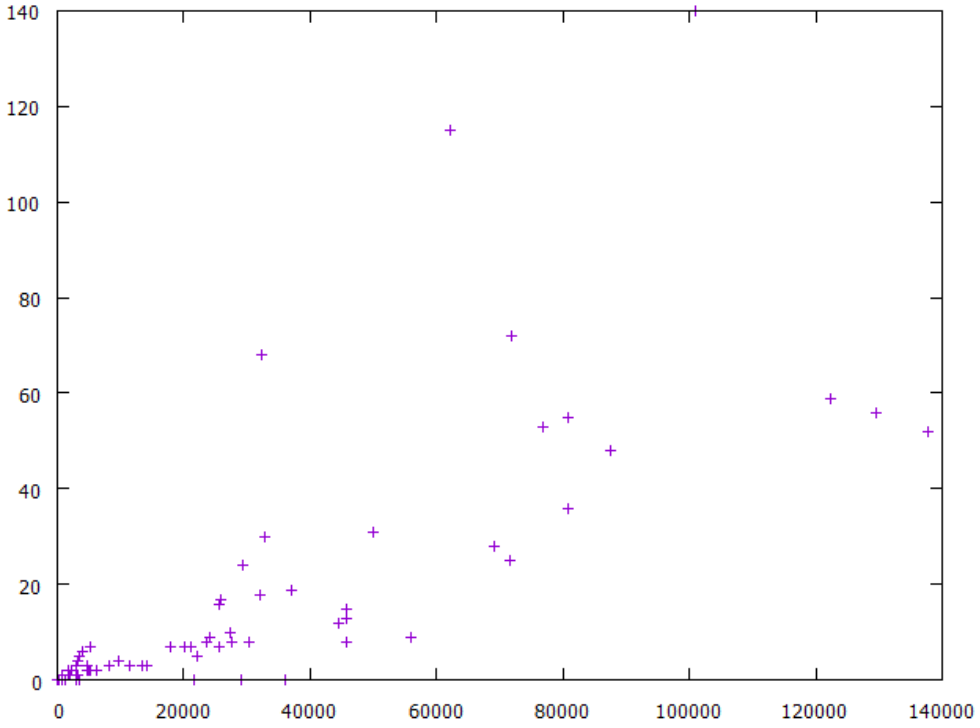

Abb. 4.16: Wasser, Büsche, Wald

Um die Laufzeit beispielsweise für Flüsse zu verbessern, wurde die Bounding Box auf den Abschnitt der Fläche beschränkt, der tatsächlich konvertiert werden soll. Dennoch kann es mehrere Sekunden dauern, bis ein Objekt umgewandelt wird, da Flüsse oft länger sind als der ausgewählte Abschnitt.

Leisure bezeichnet Orte, an denen Menschen ihre Freizeit verbringen können, wie Gärten, Sportplätze, Spielplätze und Parks (vgl. [29]). Die Grafik 4.15 zeigt die Zeit, die benötigt wurde, um die Leisures umzuwandeln. Da diese Flächen sehr groß sein können, kann die Umwandlung entsprechend lange dauern. Der Punkt rechts oben steht beispielsweise für den Büsnauer Park mit einer Größe von etwa 1,8 Quadratkilometern und einer Umwandlungszeit von 7 Sekunden. Die meisten Objekte in der Nähe vom Nullpunkt haben eine geringere Größe und dementsprechend kürzere Umwandlungszeiten.

Der Graph 4.17 zeigt die Dauer zur Generierung von Häusern im Vergleich zu ihrer Größe. Zur Annäherung der Größe wurde die Boundingbox verwendet, da jede Zelle in dieser Box traversiert werden muss. Es ist jedoch auch die Anzahl der Linien, die benötigt werden, um das Haus zu beschreiben, von entscheidender Bedeutung.

In der Theorie sollte der Graph einer Exponentialfunktion ähneln. Abgesehen von einigen Ausreißern lässt sich diese Tendenz gut erkennen.

In der Grafik 4.18 und 4.19 sieht man wie lange die Konvertierung für Schienen und Barrieren gebraucht hat. Die X-Achse beschreibt bei beiden die Länge des Objekts. An der Y-Achse kann man die Dauer in Milisekunden erkennen. Man kann sehen, dass die Eingaben konnten schnell umgewandelt werden konnten.

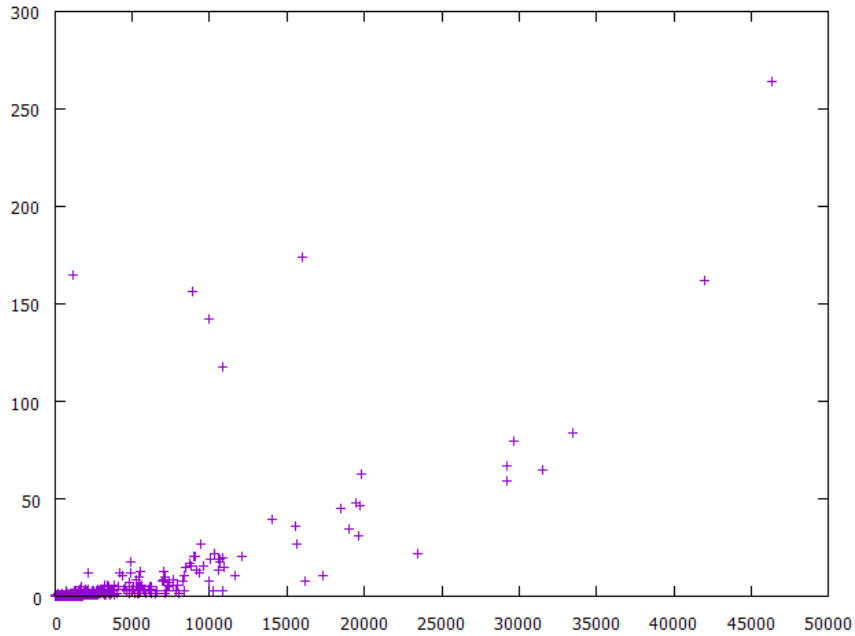

Abb. 4.17: Die Laufzeit von Gebäuden

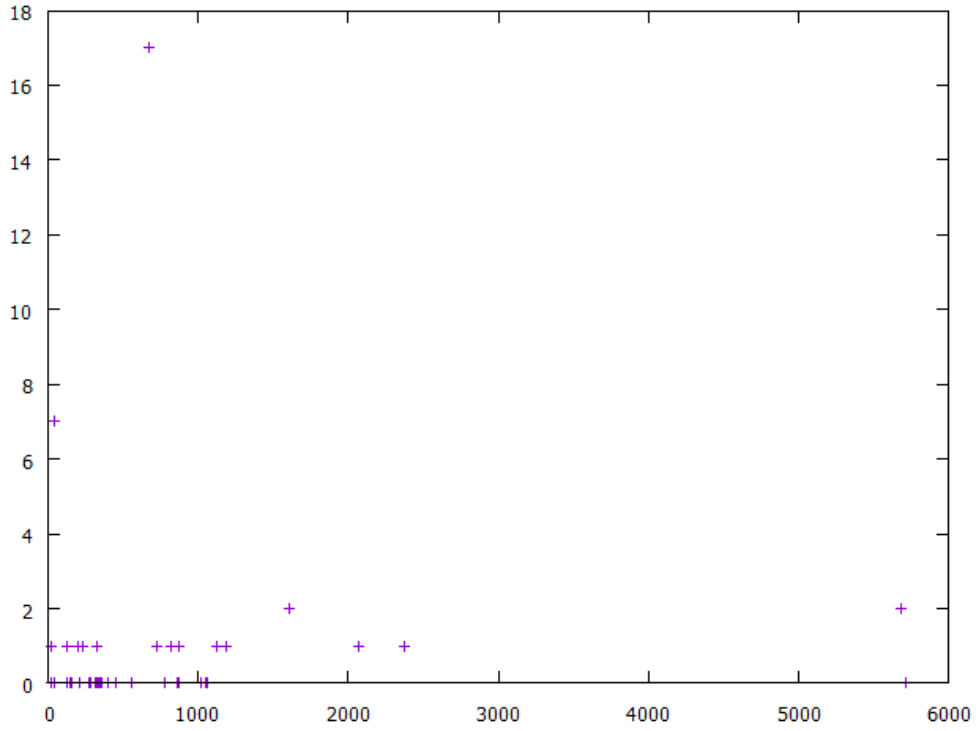

Abb. 4.18: Wie lange die Konvertierung für Schienen gebraucht hat

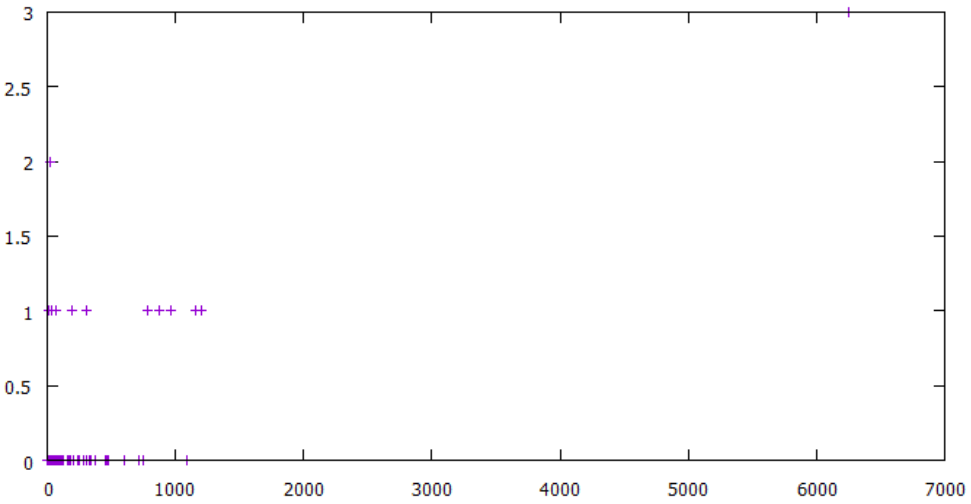

Abb. 4.19: Die Dauer verschiedener Arten von Barrieren wie Wände, Zäune, Hecken

4.5.2 Crash-Test

In diesem Kapitel soll untersucht werden, wie groß die Datensätze maximal sein dürfen, damit sie immer noch vom Programm umgewandelt werden können.

Das Experiment wurde auf einem Laptop mit einem i7-Prozessor und 16 GB Arbeitsspeicher durchgeführt. Allerdings wurde nur 4 GB Arbeitsspeicher verwendet.

Das Experiment fand in der Umgebung von Stuttgart statt, bei den Koordinaten 48,8° 9,2°. Dieser Punkt markierte den südwestlichsten Punkt eines zu berechnenden Rechtecks. Ein Fixpunkt wurde 0,02° nördlich davon festgelegt, etwa 3376 Meter entfernt. Ab 0,07° östlich traten Abstürze während der Berechnung von Wegen und Straßen in der Minecraft Welt auf. Bei noch größeren Breiten stürzte das Programm bereits während der Weltgenerierung ab. Die größte erfolgreiche Welt war mit 0,05° etwa 5568 Meter breit. Bei einer Breite von 0,06° stürzte das Programm beim Zusammenfassen der Minecraftdaten für die resultierende Zip-Datei ab.

5 Zusammenfassung

Im abschließenden Kapitel dieser Arbeit werden die verschiedenen Aspekte zusammengeführt und ein Fazit gezogen.

Allerdings werden auch potenzielle Verbesserungen aufgezeigt, insbesondere in Bezug auf die Feinabstimmung und Detailgenauigkeit der Konvertierung. Es wird diskutiert, wie eine noch realistischere und ansprechendere Spielerfahrung erreicht werden kann.

Insgesamt bietet die Arbeit einen Einblick in das Potenzial der Konvertierungstechnologie und zeigt Möglichkeiten auf, wie sie weiterentwickelt und verbessert werden kann, um eine optimale Spielerfahrung zu gewährleisten.

5.1 Fazit

Abschließend lässt sich feststellen, dass die Konvertierung einer Minecraftwelt aus OSM-Daten möglich ist. Insbesondere große Objekte lassen sich dabei gut umwandeln, und die Konvertierung erfolgt in einer akzeptablen Zeit. Dennoch gibt es Potenzial zur Verbesserung, insbesondere in Bezug auf die Feinabstimmung und Detailgenauigkeit der Konvertierung, um ein noch realistischeres und ansprechenderes Spielerlebnis zu schaffen. Es fallen immer wieder Kleinigkeiten auf, die noch verbessert werden könnten. Doch auch nach der Verbesserung kamen weitere Herausforderungen auf.

Es ist wichtig anzumerken, dass es bestimmte Herausforderungen geben kann, wenn es um die Konvertierung bestimmter Objekte oder Strukturen geht. Einige komplexe oder spezielle Elemente könnten möglicherweise nicht vollständig oder korrekt umgewandelt werden, was weitere Optimierungen erfordern würde.

Insgesamt bietet die Konvertierung von OSM-Daten in Minecraft jedoch ein vielversprechendes Potenzial, um realistische Spielwelten zu erschaffen und kreative Möglichkeiten für Spieler und Entwickler zu eröffnen. Es bleibt jedoch wichtig, kontinuierlich an der Weiterentwicklung und Verbesserung der Konvertierungstechnologien zu arbeiten, um die bestmögliche Spielerfahrung zu gewährleisten.

5.2 Verbesserungspotenzial

Unsere Anwendung baut die verschiedenen Elemente nacheinander auf und folgt dabei einer festgelegten Reihenfolge die von der Objektart abhängt. Die Priorisierung von Objekten ist berücksichtigt, könnte jedoch noch detaillierter definiert werden.

Ein deutliches Beispiel hierfür sind Straßen und Gehwege, wie in Abbildung 5.1 zu sehen ist. Der grau dargestellte Gehweg im Hintergrund überlagert die schwarze bzw. anthrazitfarbene Straße.

Es ist wichtig, die Reihenfolge und Priorisierung der Objekte weiter zu verfeinern, um sicherzustellen, dass die Darstellung in der Minecraft Welt möglichst realistisch und stimmig ist. Dies kann dazu beitragen, potenzielle Überlappungen oder Inkonsistenzen zwischen den verschiedenen Elementen zu minimieren und ein hochwertiges Spielerlebnis zu gewährleisten.

Die weißen Blöcke in der Mitte repräsentieren den Zugang zu einer S-Bahnstation, wobei es hier zu einer Überlagerung mit dem Gehweg kommt. Die auffälligen Löcher sind im Bild 5.1 entstanden, weil zuerst der weißgraue Weg und anschließend der Weg aus etwas dunklerem Stein gebaut wurde, der von unten links im Bild verläuft.

Im Kontext des Überlagerns von Straßen und Wegen ist auch die Berechnung der Breite von Bedeutung. An vielen Stellen handelt es sich um Schätzungen, die möglichst nah an den optimalen Wert heranreichen sollten. Abweichungen führen entweder zu Überlagerungen

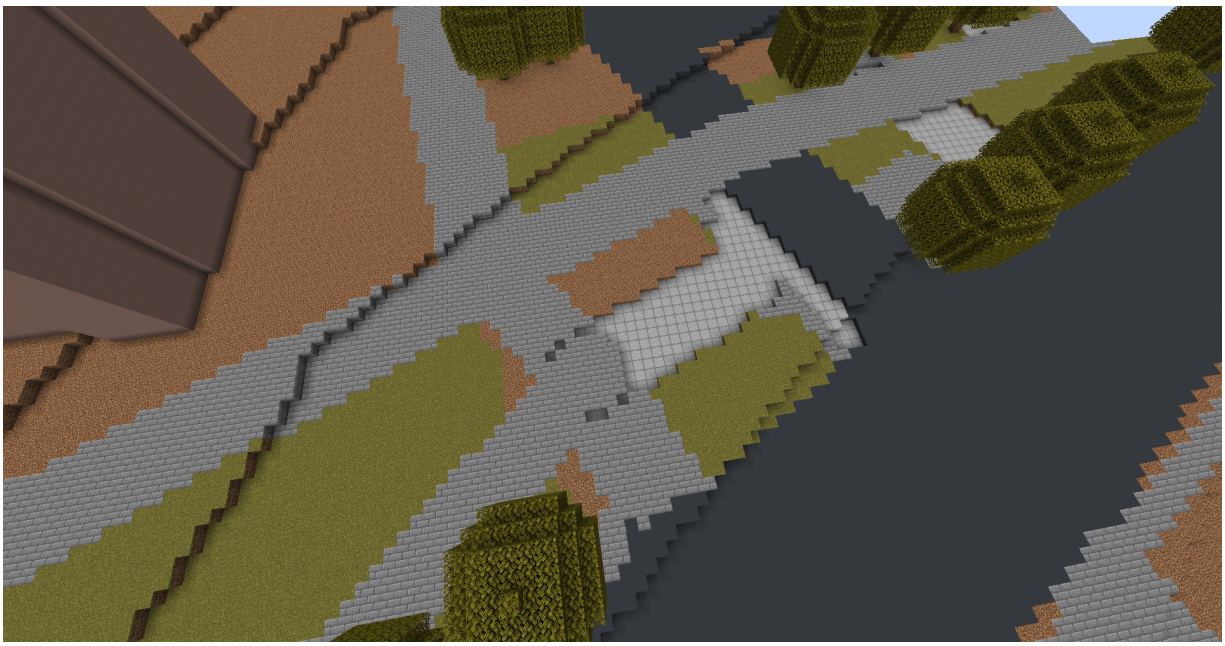

Abb. 5.1: Beispiel für das Überlagern von Straßen und Gehwegen. Hinter dem Informatikgebäude, Vaihingen

oder zu fehlenden Stellen, die von anderen Objekten oder anderweitig aufgefüllt werden müssen.

Der weißgehaltene Stein in der Mitte sollte eigentlich aus Treppenstufen bestehen, da dort in der Realität ein Eingang zu einer S-Bahnstation verläuft. Unterirdische Wege, U-Bahnen und Straßen wurden jedoch noch nicht implementiert. Diese Umsetzung gestaltet sich sehr schwierig, da darauf geachtet werden muss, dass alle Verbindungen zueinander passen, und verschätzte Breiten gravierende Konsequenzen haben können.

Das Gleiche gilt für Brücken, die momentan behandelt werden, als ob sie auf dem Boden verlaufen würden. Hier könnten weitere Verbesserungen implementiert werden, um eine realitätsnähere Darstellung zu erzielen.

Braune Blöcke, die im Bild 5.1 zu erkennen sind, repräsentieren den Blockzustand „Erde“, der verwendet wird, wenn das Programm kein passendes Material für eine Position ermitteln konnte. Weitere Prüfungen könnten eingeführt werden, um den Kartenausschnitt, z.B. ländlich oder städtisch, zu berücksichtigen und einen passenderen Block zuzuweisen, was jedoch zusätzliche Berechnungsroutinen und Performance-Einbußen mit sich bringen würde.

Weitere Optimierungsmöglichkeiten bestehen bei Zebrastreifen, Laternen und Schildern. Da deren Lage immer auf einen Punkt in der Straßenmitte verweist, muss sowohl ein realistischer Ort als auch die Breite des Objektes geschätzt werden.

In einer Minecraftwelt findet man verschiedene Biome. Damit werden in Minecraft verschiedene Klimatische Zonen nachgeahmt. Beim Betreten eines neuen Bioms fällt zunächst die Veränderung der Blöcke auf, aber auch die Atmosphäre ändert sich, wie etwa der Himmel oder die Farbe von Gras und Wasser. Eine Möglichkeit, diese Funktionalität zu nutzen, besteht darin, verschiedene Biome in bestimmten Bereichen zu speichern. Der Code dafür ist bereits vorbereitet, sodass sogenannte „Areas“ an Orten gespeichert werden können, die kurz vor der Speicherung in Biome umgewandelt werden. Es stellt sich jedoch die Frage, an welchen Orten vom Standardbiom abgewichen werden soll. In dieser Arbeit liegt der Fokus auf der Umwandlung von OpenStreetMap-Objekten, die sich hauptsächlich in städtischen Gebieten befinden. Klimatische Zonen sind in OSM in der Regel nicht verzeichnet, daher müsste zusätzlich automatisiert erkannt werden, wenn sich ein Spieler z.B. in einem Gebirge befindet.

Bisher werden nicht alle OSM-Daten in Minecraft umgewandelt, da jede Struktur einzeln implementiert werden muss. Zudem verfügen einige Objekte über eine Vielzahl von Zusatzoptionen, die in den Tags definiert sind. Einige dieser Optionen könnten noch hinzugefügt oder die Verarbeitung optimiert werden.

Der Webserver bietet derzeit nur grundlegende Funktionen, wie die Auswahl von zwei Punkten zur Generierung der Download-Datei. Es gibt jedoch Verbesserungsmöglichkeiten hinsichtlich des Bedienkomforts, wie die Anzeige eines Rechtecks beim ausgewählten Bereich oder eines Fortschrittsbalkens während der Generierung der Downloaddatei. Außerdem wäre eine Ortssuche eine sinnvolle Erweiterung. Für die Konvertierungsoptionen

im Programm könnten dem Benutzer auf der Webseite mehrere Auswahlmöglichkeiten angeboten werden.

5.3 Abschließende Worte

Das in dieser Arbeit behandelte Programm ermöglicht es dem Nutzer, einen beliebigen Ausschnitt der realen Welt über eine Webseite in Minecraft zu importieren. Die Darstellung ist ausreichend, um Gebäude, Straßen, Bahnlinien und Landschaften außerhalb der Stadtgebiete gut zu erkennen und zu identifizieren. OpenStreetMap bietet allerdings noch weitere Daten an, als derzeit vom Programm verarbeitet werden können.

Die Generierung von detailreichen Gebäuden und Strukturen gestaltet sich für die Software als Herausforderung, da viele komplexe Aspekte berücksichtigt werden müssten. Mit einem höheren Grad an Datenverarbeitung kann die Detailgenauigkeit erhöht und ein präziseres Ergebnis erzielt werden.

Für interessierte Hobby-Minecraftspieler bietet das Programm bereits eine solide Grundlage, um einen beliebigen Ausschnitt in Minecraft nachzubilden und diesen je nach Detailverliebtheit manuell zu optimieren. Im Hinblick auf den Einsatz der Software in der Städteplanung ist es wichtig, stets den Blick auf die reale Welt zu richten und die Möglichkeiten sowie Grenzen der digitalen Darstellung zu berücksichtigen.

Literaturverzeichnis

[1] "Aussicht Europaviertel," https://www.stuttgart-tourist.de/stuttgart-entdecken/lieblingsviertel/mittendrin/europaviertel, accessed: 2024-2-3.

[2] "OpenStreetMap Webseite," https://www.openstreetmap.org, accessed: 2024-2-12.

[3] M. Studios, "Minecraft," https://www.minecraft.net, 2009.

[4] "Osm2Map Spigot Plugin," https://www.spigotmc.org/resources/osm2map.78724/, accessed: 2024-1-3.

[5] "Terra1-to-1 auf Curseforge," https://www.curseforge.com/minecraft/mc-mods/terra-1-to-1-minecraft-world-project, accessed: 2023-12-13.

[6] "BuildTheEarth Website," https://buildtheearth.net/, accessed: 2023-12-25.

[7] "letzter Commit von Terra1-to-1," https://github.com/orangeadam3/terra121/commit/ffcbbca77d1adb488026f80a81ba315af8cde6c7, accessed: 2023-12-13.

[8] "Nullmeridian," https://www.lexas.de/erde/koordinatensystem/nullmeridian.aspx, accessed: 2023-12-16.

[9] "Koordinatensystem," https://www.ibm.com/docs/de/db2/10.5?topic=systems-geographic-coordinate-system, accessed: 2023-12-28.

[10] "OpenStreetMap Planet.osm," https://wiki.openstreetmap.org/wiki/Planet.osm, accessed: 2024-1-3.

[11] "Osmium," https://osmcode.org/osmium-tool/, accessed: 2024-2-14.

[12] "Euklidischer Raum," http://www.math-grain.de/download/m1/vektoren/einleitung/einleitung-1.pdf, accessed: 2024-2-1.

[13] "Die Größe eines Minecraftspielers," https://minecraft.wiki/w/Player, accessed: 2024-2-1.

[14] "Anzahl der Minecraftblöcke," https://minecraft.fandom.com/wiki/Block_states, accessed: 2023-12-16.

[15] "Minecrafts Caves & Cliffs-Update," https://www.minecraft.net/de-de/updates/caves-and-cliffs, accessed: 2024-1-14.

[16] "Wikipedia srtm," https://de.wikipedia.org/wiki/SRTM-Daten, accessed: 2023-12-13.

[17] "srtm Daten," https://srtm.csi.cgiar.org/, accessed: 2023-12-13.

[18] "Mercatorprojektion," https://de.wikipedia.org/wiki/Mercator-Projektion, accessed: 2023-12-13.

[19] "Bereich testen," https://www.sanfoundry.com/java-program-check-whether-given-point-lies-give accessed: 2023-7-1.

[20] "OpenStreetMap leisure," https://wiki.openstreetmap.org/wiki/DE:Key:leisure, accessed: 2024-1-14.

[21] "OpenStreetMap building:levels," https://wiki.openstreetmap.org/wiki/Key:building:levels, accessed: 2024-1-14.

[22] P. Koopman, "Bresenham line-drawing algorithm," *Forth Dimensions*, vol. 8, no. 6, pp. 12–16, 1987.

[23] "Collision Algorithmus," https://forum.gamemaker.io/index.php?threads/how-to-find-every-square-a-line-passes-through.101130/, accessed: 2024-1-14.

[24] "java-multi-dimensional-array-vs-one-dimensional," https://stackoverflow.com/questions/2512082/java-multi-dimensional-array-vs-one-dimensional, accessed: 2023-12-16.

[25] "Javadoc von DataOutputStream.java," https://docs.oracle.com/javase/8/docs/api/java/io/DataOutputStream.html, accessed: 2024-2-2.

[26] "java," https://www.java.com/de/, accessed: 2024-2-12.

[27] "Geofabrik," https://download.geofabrik.de/, accessed: 2024-2-13.

[28] "OpenStreetMap roter Platz an der Uni Stuttgart," https://www.openstreetmap.org/way/49852799, accessed: 2024-2-12.

[29] "OpenStreetMap Leisure," https://wiki.openstreetmap.org/wiki/DE:Key:leisure, accessed: 2024-2-4.

Ehrenwörtliche Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

15. Februar 2024, _____

Unterschrift des Studierenden