Visualisierungs- und Filterungsmethoden von CityGML-Daten in einer VR-Umgebung

HELGE OLBERDING [1]

Zusammenfassung: Geodaten in VR-Szenen werden vorwiegend beim Einsatz von Game Engines, in vielen Einsatzgebieten möglichst realitätsnah und in einer First-Person-Perspektive, visualisiert. 3D-Stadtmodelle im Format CityGML wurden hierbei eingesetzt, der Import erfolgte mithilfe eines neu entwickelten Plug-ins für die Unreal Engine 5. Für eine gezieltere Informationsvermittlung ist in so einer Visualisierung eine Informationsreduzierung der 3D-Szene notwendig, insbesondere in einer First-Person-Perspektive bei einem interaktiven 3D-Stadtmodell. Die hieraus entwickelte Anwendung demonstriert verschiedene Hervorhebungsmethoden mit Symboliken und Rendering Effekten. Die Hervorhebungs-techniken sind wiederum auf weitere Anwendungsszenarien übertragbar.

1 Einleitung

VR-Headsets (Virtual Reality) bzw. Head-Mounted Displays (HMD) gehören zu einer Technologie, welche in den letzten Jahren für die Visualisierung von Geodaten zunehmend an Bedeutung gewonnen hat (EDLER & KERSTEN 2021). Herausforderungen, Schwierigkeiten und Chancen für VR-Headsets werden im Bereich der Geoinformatik in verschiedenen Arbeiten zusammengetragen (ÇÖLTEKIN et al. 2020; VIRTANEN et al. 2020).

Geodaten in VR-Szenen werden in vielen Anwendungsbereichen möglichst realitätsnah visualisiert. Realitätsnahe Visualisierungen werden hierbei vermehrt mithilfe von Game Engines realisiert. Insbesondere für die Orientierung des Nutzenden ist eine realitätsnahe Geovisualisierung hilfreich. Die meisten VR-Anwendungen betrachten die 3D-Szene nicht von einer Top-Down-Perspektive, sondern aus der First-Person-Perspektive. Das bedeutet, dass aus der Betrachtungsperspektive mit einer VR-Brille die Gebäude 1:1 in der Größe darstellt, wie sie in der Realität wahrgenommen werden. Dadurch kann ein 3D-Stadtmodell für die Nutzenden größer und realitätsnaher wirken. Für eine gezieltere Informationsvermittlung ist in so einer Visualisierung eine Informationsreduzierung bzw. eine Filterung der 3D-Szene notwendig.

Die Informationsreduzierung kann mithilfe von alternativen bzw. abstrakten Visualisierungsansätzen erreicht werden. Das Gegenteil zu einer fotorealistischen Visualisierung ist die nicht fotorealistische Visualisierung (Non-photorealistic Rendering, kurz NPR). Diese Visualisierungen können auch als abstrakte Visualisierungen oder als stilisiertes Rendering bezeichnet werden. Umgesetzt wird eine NPR durch die Anpassung des Renderings, der Texturen und durch den Einsatz von Filtern, welche ihren Ursprung aus der Bildverarbeitung haben. Ein nicht fotorealistisches Rendering nutzt eine Vielzahl von Darstellungen, um entweder die Komplexität in der 3D-Visualisierung zu vermindern, damit ein Kunststil imitiert werden kann oder zur Hervorhebung von

[1] Technische Hochschule Würzburg-Schweinfurt, Fakultät Kunststofftechnik und Vermessung, Röntgenring 8, D-97070 Würzburg, E-Mail: Helge.Olberding@thws.de

 DOI: 10.24407/KXP:1841163147

ausgewählten Objekten. Umgesetzt wird eine NPR durch die Anpassung des Renderings, der Texturen und durch den Einsatz von Filtern, welche ihren Ursprung in der Bildverarbeitung haben (AKENINE-MÖLLER et al. 2018).
Die NPR-Ansätze sollen kontextbezogene Kommunikationen mithilfe abstrakter Methoden verbessern. Die Grundbasis für die meisten entwickelten Konzepte beruht hierbei auf der Kartosemiotik als Stilmittel. Die dargestellten Informationen und Objekte werden hierzu auf das Notwendigste reduziert. Abhängig von der Wirkung der Visualisierung werden Filterung, Texturen oder das Rendering an unterschiedlichen Punkten angepasst (SEMMO et al. 2015).
Hervorhebungsmethoden werden für 3D-Objekte benötigt, um diese potenziell auszuwählen, zu platzieren, zu verschieben oder einfach anzuzeigen. Dabei können die Hervorhebungen in ihrer Darstellung variieren (TRAPP et al. 2011).
Eine Kombination realistischer und alternativer Visualisierungsmethoden kann die Informationsvermittlung fördern und gleichzeitig können Nutzende in der VR-Szene im Maßstab 1:1 navigieren. Hierbei ist es sinnvoll, bestehende Hervorhebungsmethoden im VR-Kontext zu testen und potenzielle neue Methoden zu entwickeln.
Durch die Möglichkeiten der Unreal-Engine, der VR-Brille und dem CityGML-Plug-in ergeben sich verschiedene Forschungsfragen. Welche Möglichkeiten der Interaktionen sind mit CityGML-Daten in der Unreal Engine 5 möglich? Inwieweit bieten sich Bewegungseingaben für das Verändern der Modelle an? Welche Auswirkung hat die Perspektive einer VR-Brille auf das 3D-Stadtmodell?

2 Methoden

Für die Visualisierungsvarianten von Stadtmodellen in VR-Anwendungen wurde die Game Engine Unreal Engine eingesetzt. Für die 3D-Szenen wurden exemplarisch freizugängliche Daten der Stadt Berlin und der Stadt Würzburg verwendet, hierzu gehören Metadaten, digitale Geländemodelle und 3D-Stadtmodelle im Format CityGML. Anschließend wurden verschiedene Hervorhebungsmethoden und Interaktionsmöglichkeiten auf Basis unterschiedlicher Techniken entwickelt.

2.1 Grundlagen für die VR-Anwendung

Eine Game Engine bzw. Spieleentwicklungsumgebung ist ein Softwareframework, welches die Entwicklung von Anwendungen, die eine Echtzeit-Visualisierung benötigen, ermöglicht. Die Engines vereinen mehrere Komponenten, welche in die virtuelle Welt implementiert werden können, sodass diese wiederum in Echtzeit visualisiert werden kann (TRENHOLME & SMITH 2008). Für interaktive 3D-Geovisualisierungen in Echtzeit bieten sich verschiedenste Game Engines an. Game Engines besitzen vielfältige Möglichkeiten, detaillierte Landschaften und Modelle darzustellen. Als Game Engine wurde die Unreal Engine 5 (UE5) in der Version 5.0 verwendet. Die UE5 ermöglicht es im Vergleich zu deren Vorgänger der UE4 mehr Polygone und Objekte gleichzeitig darzustellen.
Als VR Hardware wurde als HMD die HTC Vive Pro 2 vom Hersteller HTC eingesetzt. Die HTC Vive Pro 2 beinhaltet neben dem HMD noch zwei Motion Controller. Die VR-Brille und deren

Motion Controller werden automatisch von jedem Unreal Engine Projekt erkannt. Das ist ein Vorteil bei der Entwicklung, da die Brille konstant als Kontrollbildschirm für die Anwendung verwendet werden kann.
Für eine neue Anwendung bzw. Entwicklungsumgebung wird ein neues Projekt in der UE5 angelegt. Echtzeit-Anwendungen werden mit einer digitalen Kamera betrachtet. Nutzende lenken die Kamera, um den Bildausschnitt der 3D-Szene zu verändern. Bei VR-Anwendungen wird die Kamera Position und Ausrichtung mithilfe der Kopfbewegungen angepasst.
Eine Visualisierung wird am häufigsten mit Begriffen wie der Realitätsnähe, Echtheit, Glaubwürdigkeit und dem Fotorealismus verbunden. Von diesen Begriffen erfüllt der Begriff der Realitätsnähe am ehesten die Ziele der meisten Geovisualisierungen. Grundlegend sollte eine Visualisierung bestimmten Prinzipien folgen. Sie sollte einen repräsentativen Charakter wie typische und wichtige Merkmale der jeweiligen Objekte besitzen. Daneben sollten die Bedeutung und die Details der Visualisierung eindeutig erkennbar sein. Darüber hinaus sollte die Genauigkeit der Geoinformationen feststellbar sein, damit die Visualisierung eine Berechtigungsgrundlage besitzt eine Geovisualisierung zu sein. Ebenso sollte die Genauigkeit die ursprüngliche Gestalt des Objekts wiedergeben (MACH & PETSCHEK 2006).
Die Unreal Engine 5 hat automatisiert in jedem Projekt einen realistischen Himmel in Form einer Skybox, mit volumetrischen Wolken und einem anpassbaren Sonnenstand. Für die realitätsnahen Visualisierungselemente wurde auf kostenfreie Asset-Bibliotheken zurückgegriffen. Quixel beinhaltet hierzu Physically Based Rendering Material mit hochaufgelösten Texturen, insbesondere für die Darstellung der Vegetation.

2.2 CityGML-Plug-in

3D-Stadtmodelle liegen in Deutschland vermehrt im Format CityGML vor. CityGML ist ein Anwendungsschema für die Speicherung und den Austausch von Geoinformationen. Der Zweck des CityGML-Standards und des einheitlichen Objektkatalogs ist die Festlegung der Erfassung, Datenhaltung, Bereitstellung und Visualisierung von 3D-Stadtmodellen, damit diese in unterschiedlichen Bereichen und Anwendungsszenarien verwendet werden können (COORS et al. 2016). CityGML ist ein quelloffenes Datenmodell, baut auf einem XML-Format auf und basiert auf den Standards der ISO 19xx Familie (GRÖGER et al. 2012). Ein weiterer wichtiger Kernbestandteil von CityGML ist die Verwendung verschiedener, klar definierter Detaillierungsstufen.
Ein direktes Importieren von CityGML-Daten in die Unreal Engine ist nicht möglich. Es wird eine zusätzliche Software benötigt, um das CityGML-Format umzuwandeln, dabei können je nach Format Informationen wie z.B. die enthaltenen Metadaten verloren gehen. Mithilfe eines neu entwickelten Plug-ins ist es möglich, CityGML-Daten mitsamt ihren Metadaten in die Unreal Engine 5 zu importieren. Das kostenfreie Plug-in ist in Zusammenarbeit der THWS Würzburg, der Abteilung Geovisualisierung, und der Uni Würzburg, der Abteilung Game Engineering, entstanden. CityGML Plug-in ist auf Basis von C++ Code. Der Ordner vom Plug-in wird in das Plug-in Verzeichnis der Unreal Engine 5.0 geschoben.
Nach dem Import wird es in einem Unreal Projekt manuell aktiviert. Anschließend ist es möglich, CityGML-Daten per Drag-and-Drop in das jeweilige Unreal-Projekt zu integrieren. Anschließend kann die importierte Datei mithilfe des Befehls „Build Mesh“ generiert werden. Es entsteht für

jedes Gebäude ein eigenes 3D-Modell bzw. ein Static Mesh mit drei automatisierten einfarbigen Texturen für Boden, Wand und Dachfläche.
Metadaten bleiben bei diesem Verfahren für jedes Gebäude bestehen. Zusätzliche Informationen werden direkt in den jeweiligen 3D-Modellen, unter „Building Data“ hinterlegt. Diese Informationen können manuell angepasst und mithilfe von Blueprints abgefragt werden. Hierbei sind es 21 Felder mit festen Bezeichnungen für die Informationen. Unter die Informationen fallen z.B. das Jahr der Erbauung, die Adresse, die Anzahl der Stockwerke oder das LoD. Weitere Informationen, die dem Modell zugewiesen sind, werden unter dem Abschnitt „Unused Data“ abgelegt, hierzu gehören auch die Maße jeder einzelnen Fläche eines Gebäudes.
Mithilfe des Plug-ins ist es möglich, Gebäude bis zur LoD 2 (Level of Detail) fehlerfrei zu erstellen. Mit dem Plug-in wurde getestet, dass mehrere 10.000 Gebäude in ein Projekt implementiert dargestellt werden können (Abb. 1).

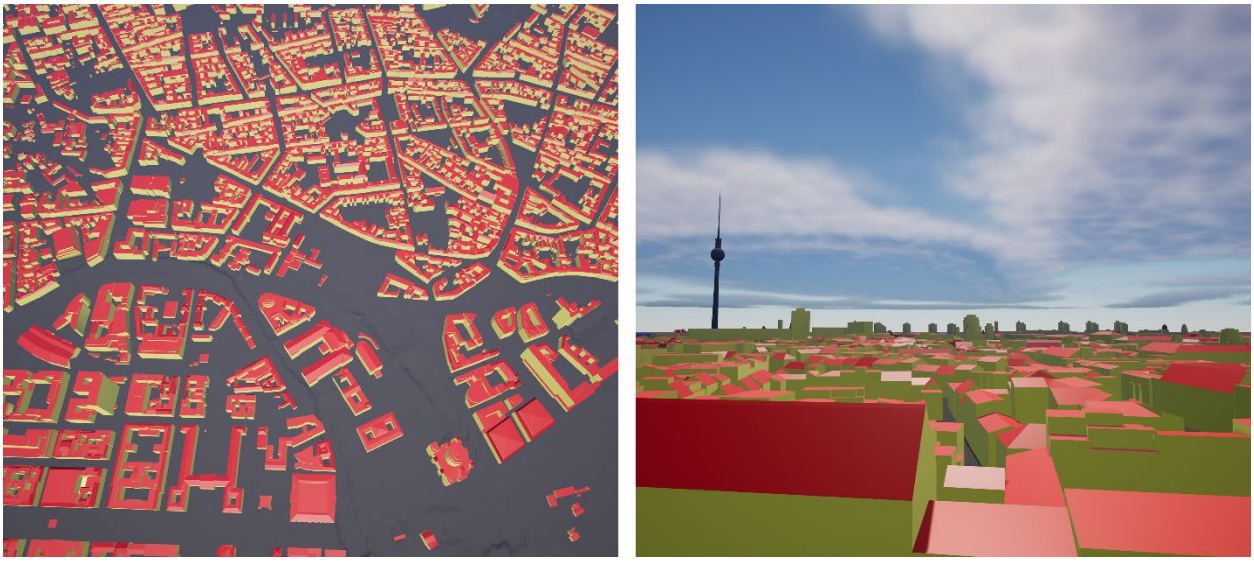

Abb. 1: Generierte Gebäude von der Stadt Berlin, mithilfe des CityGML-Plug-ins

2.3 Hervorhebungsmethoden

Es wurde zwischen drei Techniken zur Hervorhebung unterschieden. Zum einen können Objekte hervorgehoben werden durch die Implementierung von Symboliken in Form von Objekten an dem festgelegten Objekt. Diese Symbole können Marker mit 2D-Grafiken oder 3D-Modelle sein. Es sind alternativ auch Highlight-Effekte auf Basis von Partikelsysteme möglich. Mithilfe von Renderingtechniken wie dem Post Processing ist ebenfalls machbar, ausgewählte Bereiche oder die ganze 3D-Szene hervorzuheben bzw. zu verändern.

2.3.1 Post Processing

Das Post Processing ist die Nachbearbeitung von digitalen Bildern. Es ermöglicht, die gesamte Darstellung einer Szene im letzten Rendering Schritt zu verändern. In der Unreal Engine können verschiedenste Visualisierungen und Effekte zu dieser Thematik implementiert werden. Die Effekte werden hierbei immer dann angewendet, wenn sich die Kamera innerhalb eines PostProces-

sVolumes befindet. Die Begrenzung des PostProcessVolume kann gelöst werden und die vollständige Szene beeinflussen. Mittels des PostProcessVolume können verschiedenste Parameter angepasst werden. Hierzu zählen zum einen simulierte Effekte physikalischer Kameraeigenschaften wie z.B. Tiefenunschärfe und Linsenreflexionen, aber auch Kantenglättung oder eine Farbkorrektur sind möglich (EPIC GAMES INC. 2022).
Des Weiteren können individuelle Post Processing Materialien genutzt werden, um zusätzliche Effekte zu implementieren. Hierbei können Mapping Verfahren genutzt werden, wie bei der Texturierung einzelner Objekte. Dies findet alles in der Unreal Engine 5 in Form von Materialien statt. Mit den Materialien ist es möglich, NPR Darstellungen zusätzlich zur realitätsnahen Darstellung zu integrieren, dies können z.B. Cartoon-Shader bzw. Toon-Rendering sein. Kartenähnliche Darstellungsmethoden, insbesondere auf Basis von 2D-Karten können auf Cartoon-Shadern basieren. Ein Beispiel ist hierbei eine schwarze Kontur, die als eine zusätzliche Outline für 3D-Modelle dient. Mithilfe von Outlines können 3D-Objekte klarer voneinander unterschieden werden.
Für die Erstellung eines solchen Materials, z.B. für die Erkennung von geometrischen Kanten und Ränder wird die Funktion SceneDepth genutzt. Die Funktion ermöglicht es, die Entfernung einzelner Objekte mit Farbinformationen darzustellen, abhängig von der Szenenposition der Kamera. Durch die Implementierung eines Filters kann die benötigte Distanz festgelegt werden, in der eine Kante dargestellt wird. Durch eine Rundung auf 0 oder 1 werden nur zwei Farbwerte ausgegeben. Alle Werte unterhalb der gewünschten Distanz werden weiß dargestellt, ansonsten schwarz. Im verwendeten Material müssen nun alle weiteren Szeneninformationen integriert sein und anschließend werden diese mit den schwarzen Linien vereint. Das Ergebnis zeigt, durch die Randbetonung, eine deutlich abstraktere zeichnerische Darstellung, in der angrenzende Gebäude deutlich leichter unterscheidbar sind.
Der Vorteil durch eine visuelle Veränderung der Szene durch Post Processing Einstellungen und Materialien ist zum einen die Vielfalt unterschiedlicher Darstellungsmethoden der gleichen Szene, unabhängig von jeglichem vorliegenden 3D-Objekt. Außerdem lassen sich Post Processing Effekte leicht in Interaktionen implementieren.
In der UE5 kann für jedes Mesh der Ausdruck SceneDepth aktiviert werden. SceneDepth gibt die vorhandene Szenentiefe für das jeweilge Mesh aus. Dabei kann SceneDepth die Tiefe an jeder beliebigen Stelle abtasten, auch wenn z.B. andere Objekte sich vor dem Mesh befinden. Mithilfe der Funktion SceneDepth und einer Anpassung der Post Processing Materialien, ist es dadurch möglich, nur auf bestimmte 3D-Modelle Filter einzusetzen. Dadurch ist es möglich, Effekte wie eine Variante eines Röntgenblicks zu visualisieren. Hierzu bietet sich die Outline-Visualisierung an.

2.3.2 Kartographische Symbole

Die eingesetzten Symbole basieren auf kartografischen Signaturen. Die Signaturen sind grafische Zeichen, die die Lage, Art sowie qualitativen und quantitativen Merkmale des Objektes durch eine symbolische oder ikonische Darstellung zum Ausdruck bringen. Gleiche Signaturformen (Kreis, Quadrat, Dreieck etc.) werden von Karte zu Karte immer wieder an andere Begriffe gebunden. Hierbei wurde sich zunächst auf die punkthafte Klassifizierung mit zwei technischen Varianten konzentriert.

Die erste Variante besitzt als Grundlage 2D-Objekte. Diese 2D-Objekte beinhalten PNG-Grafiken. Damit eine 2D-Grafik in einem 3D-Raum am besten erkennbar ist, muss diese frontal betrachtet werden. Um zu gewährleisten, dass eine frontale Sicht gegeben ist, wird die Ausrichtung der Symbole dynamisch zu der Kameraperspektive stetig neu angepasst. Dieser Ansatz wurde schon in frühen 3D-Spielen der 90er eingesetzt wie z.B. Doom oder Duke Nukem 3D. Dadurch ist es ein rechenschonender Ansatz, der es ermöglicht aus einer 2D-Karte Signaturen zu nutzen, ohne diese aufwendig für den 3D-Raum neu zu modellieren. Die 2D-Symbole können zusätzlich aus mehreren PNG-Grafiken bestehen, die zusammen eine Sprite-Animation darstellen können.
Die zweite Variante besitzt als Grundlage 3D-Objekte. Hierzu wird ein FBX-Modell mit dem jeweiligen Symbol verknüpft. Neben komplexeren 3D-Modellen können auch einfache Kugeln oder Quadrate genutzt werden. Das Material der 3D-Objekte beschreibt die farbliche Darstellung. Animationen sind mithilfe einer Rigid Hierarchical Animation oder mit einer Morph Target Animation umsetzbar.
Das Vorhandensein und die Positionen der Symbole beruhen entweder auf der Basis einer Tabelle mit Metadaten oder auf festgelegten Informationen aus den CityGML-Gebäuden. Theoretisch ist es möglich, dass Form und Farbe der Symbole dynamisch veränderbar sind.

2.3.3 Partikelsysteme

Ein Partikel-Rendering-System bzw. ein Partikelsystem befasst sich mit der Visualisierung von atmosphärischen Effekten und gehört bei den Rendertechniken zu den Visual Effects. Das Besondere an einem Partikelsystem ist die sehr große Anzahl relativ einfacher Grafikobjekte (Partikel) für die Darstellung von Funken, Rauch, Feuer, Wasser etc. als Partikeleffekte. Alle Darstellungen haben hierbei meist keine eindeutig definierbare, nicht glatte und nicht eindeutig abgrenzte Oberfläche. Durch die einzigartigen Eigenschaften sind eigene Renderingsysteme in Game-Engines implementiert. Die Besonderheiten eines Partikelsystems sind zum einen, dass die Grafikobjekte hierzu meist einfache 2D-Objekte sind, welche aus zwei Dreiecken bestehen, diese werden auch als Quad bezeichnet. Zum anderen können Partikel unterschiedlich animiert werden. Hierbei können Parameter von Bild zu Bild variieren, dazu gehören: Position, Ausrichtung, Größe, Texturkoordinate. Animationen sind hierbei manuell oder prozedural möglich. Außerdem werden einzelne Partikel in vielen Fällen kontinuierlich erschaffen und gelöscht. Ein Emitter ermöglicht im System, die Partikel mit einer benutzerdefinierten Rate zu erzeugen. Wenn Partikel sich außerhalb einer festgelegten Zone bewegen oder ihre Zeitspanne abläuft, werden sie wieder entfernt (GREGORY 2019).
Partikelsysteme können z.B. eingesetzt werden um Wasser oder Smogausbreitungen zu visualisieren. Partikelsysteme sind aber auch hilfreich, um einem3D-Objekt zusätzliche Highlights zu geben, hierbei müssen Partikelsysteme nicht durchgehend aktiv sein. Einzelne Effekte können für einen kurzen Moment abgespielt werden, um den Blick des Nutzenden auf bestimmte Punkte zu lenken. Das besondere bei Partikeln ist die Möglichkeit tausende von kleinen animierten Objekten darzustellen.

2.4 Interaktive Elemente

In virtuellen Welten werden dem visuellen Sinn Bewegungen vorgetäuscht die, den Gleichgewichtssinn betreffend, nicht exakt der Realität entsprechen. Die Bewegungskontrolle wurde daher innerhalb der Anwendung eingeschränkt, um der Motion Sickness vorzubeugen. Teleportation ist eine alternative Fortbewegungsvariante, durch die keine Bewegungsvisualisierung dargestellt wird. Daraus ergibt sich, dass kein Konflikt zwischen sensorischem und vestibulärem System entsteht. Die Person kann sich mithilfe einer Teleportation durch die 3D-Szene bewegen oder sich an festen Punkten teleportieren. Durch die festen Teleportationspunkte können leichter größere Distanzen überwunden werden und Personen können an einer idealen bzw. gelenkten Position befinden, um bestimmte Informationen besser zu erkennen.

Die Motion Controller können die Bewegungen der Handpositionen abbilden. In den meisten Anwendungen mit VR-Brillen ist das 3D-Modell von Händen oder der Controller das Einzige, was Nutzende in der virtuellen Welt von sich selbst sehen. Mit diesen „Händen“ ist es für viele Menschen verständlich, auf Objekte zu zeigen oder auf Objekte zu drücken, um diese zu aktivieren oder auszulösen. Dementsprechend sind alle interaktiven Elemente auf diese zwei Varianten ausgerichtet.

Informationen können neben der 3D-Visualisierung, mithilfe einer grafischen Benutzeroberfläche (Graphical User Interface, Abk. GUI) dargestellt werden. Hierbei werden 2D-Elemente zu den 3D-Visualisierungen eingebunden. Hierzu gibt es in der Unreal Engine 5 die Head-up Displays (HUD). Ein HUD liefert zusätzliche Informationen z.B. in Form von Anzeigen, Balken, Texten und Bildern. Innerhalb der Anwendung sind im HUD Informationen dargestellt zum Szenario, Koordinateninformationen des Nutzenden, die festen Teleportationspunkte und Angaben, welche Hervorhebungstechniken aktiv sind.

Ein HUD kann je nach Einbindung die Immersion und die Übersicht beeinträchtigen, dies geschieht, wenn sich zu viele Elemente auf der Bildschirmoberfläche befinden. Außerdem ist es schwerer HUD-Elemente in den Randbereichen zu sehen, wenn ein VR-Headset verwendet wird. Im Gegensatz zu einer Desktopanwendung kann sich der Kopf mit einem HMD, nicht anders ausrichten, um Randbereiche der Bildausschnitte auf den Linsen des VR-Headsets zu betrachten. Dementsprechend gibt es zwei Lösungen. Die Erstere ist das Verschieben der HUD-Elemente stärker Richtung Mitte. Die Alternative ist die Erstellung eines HUD-Elements als Objekt in der 3D-Szene.

Durch das implementieren von HUD-Elementen, als Objekte ist es möglich, Interaktionen mit den Motion Controller zu ermöglichen. Damit nicht an einer festen Position das HUD steht, wurde dies für alle Anwendungsszenarien an das 3D-Modell des linken Motion Controllers gekoppelt. Um zu verhindern, dass das HUD die ganze Zeit eingeblendet ist, ist es zunächst unsichtbar. Das HUD wird erst angezeigt, wenn der linke Motion Controller zu einem festgelegten Winkel der Kamera ausgerichtet ist. Für einen Nutzenden ergibt sich nun der Eindruck, als würde eine interaktive Informationsfläche jedes Mal auftauchen, wenn die linke Hand vor dem Kopf gehalten wird. Im HUD haben die Nutzenden nun die Möglichkeit, eine Auswahl an Hervorhebungsmöglichkeiten zu aktivieren (Abb. 2).

Abb. 2: Beispielhafte HUD-Oberfläche in VR

3 Ergebnisse

Um die Forschungsfragen zu beantworten, ist eine Anwendung entstanden. Diese Anwendung demonstriert verschiedene Filterungs- bzw. Hervorhebungsmethoden. Dies ist zum Beispiel durch das Ausblenden oder auch das farbliche oder stilisierte Hervorheben ausgewählter Objekte möglich.

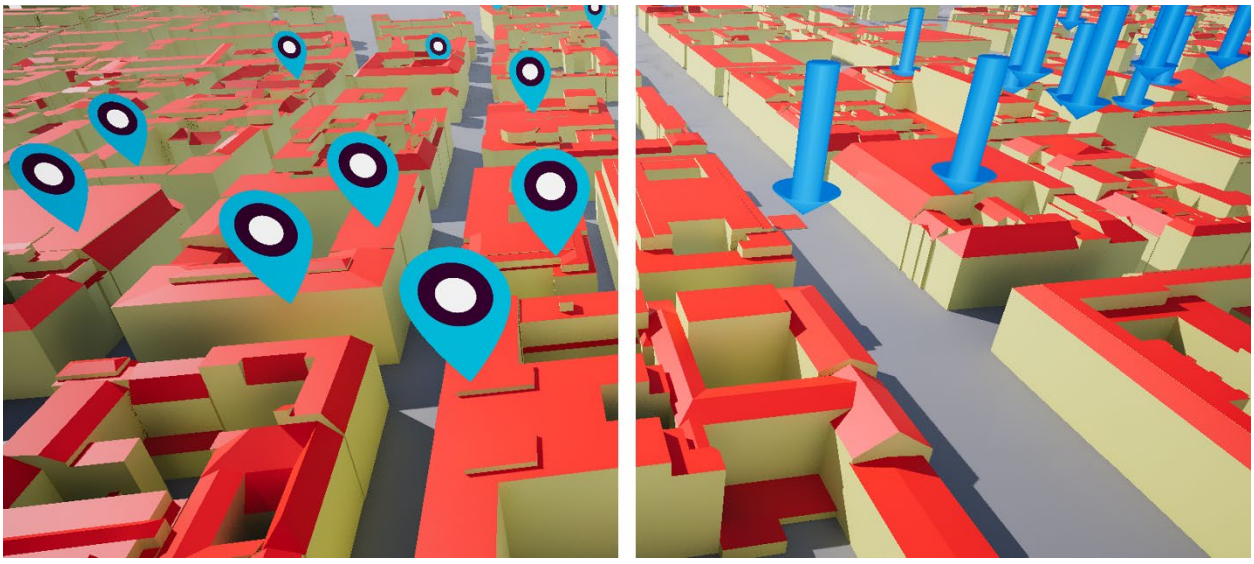

Abb. 3: Verschiedene Symbol zur Hervorhebung selektierter Gebäude

Mithilfe des CityGML-Plug-ins ist es möglich, Gebäude abhängig von enthaltenen Metadaten, in den „Building Data“, hervorzuheben. Hierzu werden Gebäude mithilfe von Blueprints ausgewählt und anschließend werden Symbole (Abb.3) oder NPR-Effekte für diese Gebäude erzeugt. Bei einer

Materialanpassung innerhalb der CityGML-Gebäude können jedoch noch fehlerhafte Visualisierungen auftreten. Mit der Hilfe des HUDs und zusätzlichen Bewegungseingaben können die Hervorhebungen in ihrer Darstellung von Nutzenden, innerhalb der Anwendung, dynamisch ohne Schwierigkeiten verändert werden.
Ein großer Vorteil von allen Hervorhebungsmethoden ist, dass diese Darstellungen einfach miteinander ausgetauscht werden können. Bei den Symbolen und den Partikeln muss nur das Asset und bei dem Post Processing das Material gewechselt werden. Dadurch sind Anpassungen der Hervorhebungen unkompliziert machbar und auf verschiedene Anwendungsszenarien beim Vorliegen der Metadaten und Assets übertragbar. Praxisbezogene Anwendungsszenarien könnten für die Hervorhebung bzw. Filterung zum Beispiel Informationsanzeigen für die Bereiche Tourismus oder dem Energieverbrauch sein. Eine spielerische Hervorhebungsmethode ist ein Röntgenblick, um ein gesuchtes Objekt hinter anderen Gebäuden dennoch erkennbar zu machen. Solche Hervorhebungsmethoden sind insbesondere in der First-Person Perspektive durch die eingeschränkte Übersicht hilfreich (Abb. 4).

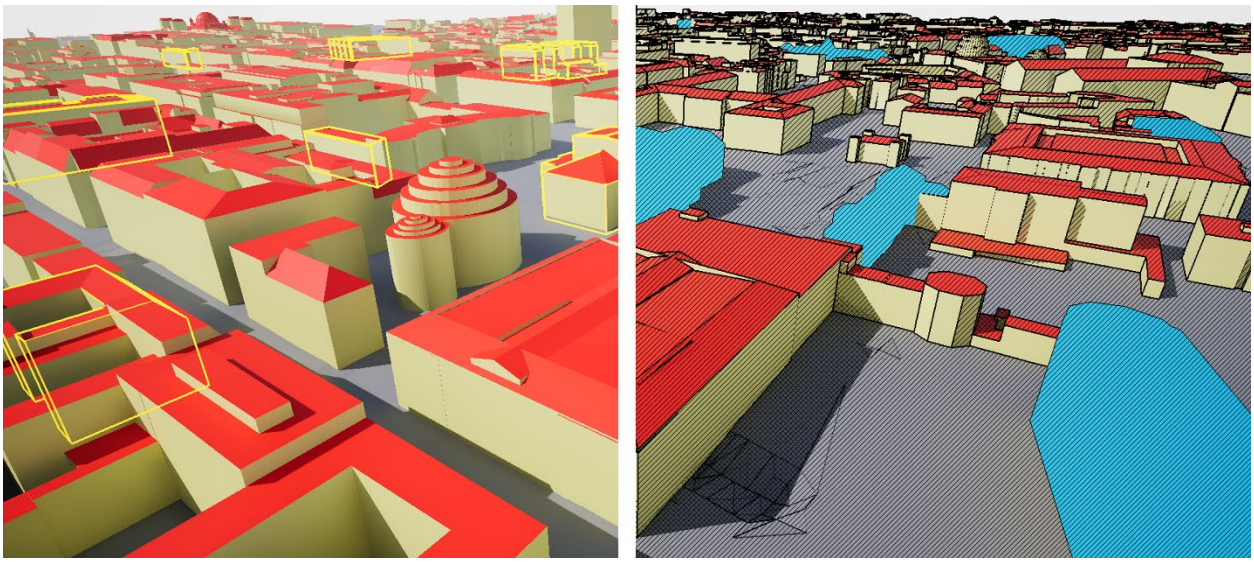

Abb. 4: Vollständige und selektierte Filterungsbeispiele mithilfe von NPR

4 Fazit & Ausblick

3D-Stadtmodelldaten werden je nach Anwendungsbereich und Ziel visualisiert. Innerhalb einer Game Engine, existieren kaum technische Limitationen wie die Echtzeit-Renderings aussehen können. Fotorealistische und nicht fotorealistische Visualisierungen können in einer Anwendung sein und mit einem Knopfdruck vom Nutzenden verändert werden. Komplexe dynamische Materialien und Partikeleffekte in Kombination mit einer VR-Brille könnten bei einer genaueren Untersuchung noch mehr sinnvolle Optionen ermöglichen, den Blick des Nutzenden zu lenken.
Durch das schnelle und vollständige Importieren von CityGML-Daten in die Unreal Engine 5 können größere Stadtmodellszenen als ergänzendes Element in Anwendungen implementiert werden. Die Anwendungsszenarien sind erweiterbar und die Hervorhebungstechniken sind mit anderen Metadaten durch das Anpassen von Parametern übertragbar und veränderbar.

Wichtig ist beim Einsatz von der VR-Brille als Ausgabegerät inwieweit die neue Betrachtungsweise spezielle Anpassungen erfordert, um Informationen effektiv zu vermitteln. Die Betrachtung eines Stadtmodells aus der First-Person Perspektive ist im Vergleich zu den meisten Geovisualisierungen auf einem Desktop Bildschirm eine andere. Geovisualisierungen mit dem Ausgabegerät Desktopbildschirm, nehmen ehe eine Top-Down-Perspektive ein. Natürlich könnte ein Stadtmodell auch von oben in VR betrachtet werden, es ist aber auch möglich, vermehrt Ansätze der Hervorhebung aus dem Bereich der Videospiele zu implementieren.
Die beiden größten Features der Unreal Engine 5 sind Lumen und Nanite, welche noch realitätsnähere Visualisierungen ermöglichen. Lumen dient zur Echtzeit Global Illumination und Reflektion. Nanite automatisiert die Skalierung der Polygonanzahl von Objekten abhängig der Kameraposition. Diese waren in der Version 5.0 für VR-Anwendungen noch nicht zugänglich, erst mit der Version 5.1. Daraus ergibt sich eine Anpassung des CityGML-Plug-ins auf die Version 5.1.
Die visuellen Elemente der Anwendung können in nachfolgenden Implementierungen durch das Involvieren zusätzlicher Sinneseindrücke erweitert werden. Dies sind zum einen auditive Elemente, wie z.B. der Einsatz von Sound Cues, um die Betrachtungsrichtung zu lenken, zum anderen haptische Elemente beim Motion Controller, wie z.B. starke und schwache Vibrationen oder unterschiedliche Intervalle.
Eine weitere Aufgabe ist die Untersuchung der Verbesserungsqualität der Informationsübermittlung mithilfe der verschiedenen Hervorhebungstechniken z.B. durch den Einsatz von Eye Tracking. Eye Tracking Möglichkeiten sind hierzu in der HTC Vive Pro 2 integriert.

5 Literaturverzeichnis

AKENINE-MÖLLER, T., HAINES, E. & HOFFMANN, N., 2018: Real-Time Rendering, Fourth Edition. 4th ed. Milton: Chapman and Hall/CRC, 651-656.

CÖLTEKIN, A., LOCHHEAD, I., MADDEN, M., CHRISTOPHE, S., DEVAUX, A., PETTIT, C, LOCK, O., SHUKLA, S., HERMAN, L., STACHOŇ, Z., KUBÍČEK, P., SNOPKOVÁ, D., BERNARDES, S. & HEDLEY, N., 2020: Extended Reality in Spatial Sciences: A Review of Research Challenges and Future Directions. IJGI, **9**(7), 439, https://doi.org/10.3390/ijgi9070439.

COORS, V., ANDRAE, C. & BÖHM, K., 2016: 3D-Stadtmodelle. Konzepte und Anwendungen mit CityGML. Berlin, Offenbach, Wichmann, 13-14.

EDLER, D. & KERSTEN, T., 2021: Virtual and Augmented Reality in Spatial Visualization. J. Cartogr. Geogr. inf., **71**(4), 221-222, https://doi.org/10.1007/s42489-021-00094-z.

EPIC GAMES INC., 2022: Unreal Engine. Post Process Effects. Online verfügbar unter https://docs.unrealengine.com/5.1/en-US/post-process-effects-in-unreal-engine/, letzter Zugriff 15.01.2023.

GREGORY, J., 2019: Game engine architecture. Third edition. Boca Raton, London, New York: CRC Press Taylor & Francis Group (An A.K. Peters book), 710-712.

GRÖGER, G., KOLBE, T., NAGEL, C. & HÄFELE, K., 2012: OGC City Geography Markup Language (CityGML) Encoding Standard. Version 2.0.0, OGC Doc. 12-019. Hrsg. v. Open Geospatial Consortium Inc. Online verfügbar unter http://www.opengeospatial.org/standards/citygml, letzter Zugriff 01.11.2022.

MACH, R. & PETSCHEK, P., 2006: Visualisierung digitaler Gelände- und Landschaftsdaten. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 1-4.

SEMMO, A., TRAPP, M., JOBST, M. & DÖLLNER, J., 2015: Cartography-Oriented Design of 3D Geospatial Information Visualization – Overview and Techniques. The Cartographic Journal **52**(2), 95-106, https://doi.org/10.1080/00087041.2015.1119462.

TRAPP, M., BEESK, C., PASEWALDT, S. & DÖLLNER, J., 2011: Interactive Rendering Techniques for Highlighting in 3D Geovirtual Environments. Advances in 3D Geo-Information Sciences, Thomas H. Kolbe, Gerhard König und Claus Nagel (Hrsg.), Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg (Lecture Notes in Geoinformation and Cartography), 197-210.

TRENHOLME, D. & SMITH, S., 2008: Computer game engines for developing first-person virtual environments. In: Virtual Reality, **12**(3), 181-187. https://doi.org/10.1007/s10055-008-0092-z.

VIRTANEN, J., JULIN, A., HANDOLIN, H., RANTANEN, T., MAKSIMAINEN, M., HYYPPÄ, J. & HYYPPÄ, H., 2020: Interactive Geo-Information in Virtual Reality – Observations and future challenges. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., **44**(4/W1-2020), 159-165, https://doi.org/10.5194/isprs-archives-XLIV-4-W1-2020-159-2020.