Otto-von-Guericke Universität Magdeburg

Fakultät für Informatik

Bachelorarbeit

VR-unterstütze Kollisionserkennung von Satelliten

Autor:

Christopher Olson

25. August 2021

1. Prüfer: Prof. Dr.-Ing. Bernhard Preim
Institut für Simulation und Graphik
Otto-von-Guericke Universität Magdeburg

2. Prüferin: Dr.-Ing. Georgia Cesar de Albuquerque Richers
Institut für Softwaretechnologie
Deutsches Zentrum für Luft und Raumfahrt

Betreuer: Dr.-Ing. Patrick Saalfeld
Institut für Simulation und Graphik
Otto-von-Guericke Universität Magdeburg

Betreuer: Andreas-Christoph Bernstein
Institut für Softwaretechnologie
Deutsches Zentrum für Luft und Raumfahrt

Olson, Christopher:
VR-unterstütze Kollisionserkennung von Satelliten
Bachelorarbeit, Otto-von-Guericke Universität Magdeburg, 2021.

Zusammenfassung

Das Ziel dieser Arbeit ist es zu untersuchen, ob die VR-Technologie bei der Kollisionsanalyse von Satelliten eine sinnvolle Ergänzung ist. Um die Forschungsfrage zielführend zu beantworten, wurde zunächst ein Konzept ausgearbeitet und mit Mitarbeiterinnen und Mitarbeitern des Deutschen Zentrums für Luft- und Raumfahrt (DLR) erörtert. Anschließend wurde eine Virtual Reality (VR)-Applikation in Unity auf Basis des besprochenen Konzepts entwickelt. Diese beinhaltet u.a. die Möglichkeit, aktuelle Satellitenpositionen zu betrachten und Conjunction Data Messages (CDMs) mithife verschiedener Modi zu analysieren. Abschließend wurde die VR-Applikation bei einem Experteninterview evaluiert. Dabei stellte sich heraus, dass VR für die Kollisionsanalyse von Satelliten nicht geeignet ist, da die Analysedauer zu groß ist und einige Kollisionsberechnungen nur per Hand durchgeführt werden können. Allerdings sei die entwickelte VR-Anwendung für die mediale Aufmerksamkeit sehr gut geeignet, sodass hiermit die Weltraumschrottproblematik Personen, die nicht vom Fach sind, zugänglich vermittelt werden könnte. Dies könnte im Rahmen einer weiteren Forschungsarbeit näher untersucht werden.

Inhaltsverzeichnis

Danksagungen

In diesem Abschnitt möchte ich mir die Zeit nehmen, um mich bei all den Menschen zu bedanken, die mich bei der Bearbeitung meiner Bachelorarbeit unterstützt haben. Damit anfangen möchte ich bei Patrick Saalfeld und Andreas Bernstein, die mir als Betreuer dieser Arbeit immer wieder zur Seite standen und in wöchentlichen Meetings diverse Ideen und Tipps beigesteuert haben. Ich hätte mir wirklich keine besseren Betreuer wünschen können! Weiterhin möchte ich mich im Voraus bei Georgia Richers und Bernhard Preim für ihre Zeit bei der Begutachtung dieser Arbeit bedanken. Mein Dank geht auch an alle Mitarbeiterinnen und Mitarbeiter des DLR, die bei meiner Zwischenpräsentation zugesehen und weitere Ideen beigesteuert haben. Ein großes Dankeschön auch an Hauke Fielder, der sich sehr viel Zeit genommen hat, die Arbeit anzusehen und zu evaluieren.

Außerdem möchte ich mich bei meiner Mutter – Antonia Olson – für ihre großartige Unterstüzung bei dem Korrekturlesen und im Verlauf meines Studiums bedanken. Ebenso möchte ich mich bei meiner Schwester – Isabel Olson – bedanken, die sich ebenfalls viel Zeit für das Korrekturlesen genommen hat. Vielen Dank auch an Sven Klaaßen, der mich bei mathematischen Fragen unterstützt hat. An dieser Stelle auch einmal meine Glückwünsche zu deiner Promotion! Ein großes Dankeschön auch an Tim Krusch, der meine VR-Applikation getestet hat. Zuletzt möchte ich mich bei meiner wundervollen Freundin Charlotta-Marlena Geist bedanken, die mir sowohl beim Korrekturlesen als auch in mentaler Hinsicht eine große Stütze war.

Abkürzungsverzeichnis

CA	Conjunction Assessment
CDM	Conjunction Data Message
CSG	Constructive Solid Geometry
DLR	Deutsches Zentrum für Luft- und Raumfahrt
ESA	European Space Agency
GESTRA	German Experimental Space Surveillance and Tracking Radar
GHOST	GPS High Precision Orbit Determination Software Tools
NORAD	North American Aerospace Defense Command
RTN	Radial Tangential Normal
SDF	Signed Distance Function
SGP	Simplified Perturbations Model
SSA	Space Situational Awareness
TCA	Time of Closest Approach
TLE	Two-Line Elements Set
UI	User Interface
VR	Virtual Reality

1. Einführung

Der zunehmende Weltraumschrott bereitet der Weltraumfahrt ein immer größer werdendes Problem, da von diesem stets Kollisionsgefahr ausgeht. So können Objekte bereits ab einem Durchmesser von 1 cm Satelliten beschädigen oder sogar zerstören (ESA [2005]). Bei Weltraumschrott unterscheidet man zwischen natürlichem Schrott wie z.B. Meteoroiden und künstlichem Schrott wie bspw. Raketenteile und inaktiven Satelliten. Die Anzahl an Objekten im Orbit der Erde wächst seit Beginn der Raumfahrt kontinuierlich, wie in Abb. 1.1 zu sehen ist. Dabei handelt es sich in der Grafik ausschließlich um die Objekte, die derzeit geortet und katalogisiert werden können. Zusätzlich gäbe es laut der European Space Agency (ESA) schätzungsweise 900.000 nicht geortete Objekte mit einem Durchmesser von 1 bis 1 0cm, die, wie zuvor geschildert, jeweils eine große Gefahr für aktive Satelliten ausstrahlen (ESA [2021a]).

In einer wissenschaftlichen Arbeit von Kessler und Cour-Palais [1978] wurde bereits sehr früh vor den Folgen einer steigenden Anzahl von Objekten im Orbit der Erde gewarnt. Eine zu hohe Menge an Weltraumschrott in der niedrigen Erdumlaufbahn könne dazu führen, dass eine Kollision zwischen zwei Objekten eine Kettenreaktion von weiteren Kollisionen hervorruft und den Einsatz von Satelliten in der niedrigen Erdumlaufbahn für Generationen blockiert („Kessler-Syndrom“, Kessler et al. [2010]).

Die erste und bislang einzige Kollision zwischen zwei Satelliten fand am 10. Februar 2009 statt. Dabei stieß der bis dahin noch aktive US-Satellit *Iridium 33* mit dem bereits inaktiven russischen *Cosmos 2251* Satelliten zusammen und erzeugte über 2201 messbare und schätzungsweise 100.000 kleine – und daher nicht verfolgbare – Bruchstücke (Liou [2014]). Die Kollision hätte durch ein Ausweichmanöver verhindert werden können, jedoch befand sich das Kollisionsrisiko am Tag der Kollision lediglich auf Platz 152 der von SOCRATES (Satellite Orbital Conjunction Reports Assessing Threatening Encounters in Space) herausgegebenen Kollisionswarnungen (Kelso et al. [2009]).

Die Kosten, die durch die Weltraumschrottproblematik entstehen, werden für den geostationären Orbit auf 5-10% der Missionskosten (Undseth et al. [2020]) und

1.1: Count evolution by object type, 1958-2019. Quelle: ESA, About Space Debris, https://www.esa.int/Safety_Security/Space_Debris/About_space_debris, letzter Zugriff: 22.08.2021

somit auf mehrere hundert Millionen Dollar geschätzt (ESA [2020]). Diese seien bei Missionen in der niedrigen Erdumlaufbahn sogar noch höher.

Um die Kollisionsgefahr und die daraus resultierenden Kosten zu verringern, aber auch das „Kessler-Syndrom" zu vermeiden, werden bereits zahlreiche Forschungsarbeiten in den unterschiedlichsten Bereichen geführt. So werden Ortungsverfahren verbessert, neue Richtlinien für Satellitenbetreiber beschlossen und es wird z.B. mit der „ClearSpace-1"-Mission versucht, bereits vorhandenen Weltraumschrott zu entfernen. (ESA [2019]).

Zielstellung

Das Ziel dieser Arbeit ist es, an weiteren Methoden für die Analyse von möglichen Kollisionen zu forschen, um der wachsenden Bedrohung durch Weltraumschrott entgegenzuwirken. Dabei wird insbesondere untersucht, ob VR für die Kollisionsanalyse von Satelliten geeignet ist, da VR hierfür bislang noch nicht verwendet wurde, jedoch – im Vergleich zu Desktop-Anwendungen – eine stärkere Immersion und ein verbessertes räumliches Vorstellungsvermögen bietet. Die Forschungsfrage lautet daher: „Ist die VR-Technologie bei der Kollisionsanalyse von Satelliten eine sinnvolle Ergänzung?"

Zur Beantwortung dieser Frage wurden zunächst einige Umsetzungsideen gesammelt und bei einer Zwischenpräsentation mit einigen Mitarbeitern des Deutschen Zentrums für Luft- und Raumfahrt (DLR) erörtert. Auf Basis des aus der Zwischenpräsentation entstandenen Konzepts wurde anschließend eine VR-Applikation in Unity entwickelt. Innerhalb jener Anwendung ist es möglich, alle katalogisierten Weltraumobjekte auf einmal zu betrachten, beliebig im Orbit der Erde zu reisen und Satelliten aus der Nähe zu besuchen. Weiterhin lassen sich sogenannte CDMs analysieren, indem unter anderem die Positionsabweichungswahrscheinlichkeiten der Objekte visualisiert

und deren Überschneidungen farblich hervorgehoben werden. Zuletzt wurde die Applikation mittels eines Experteninterviews evaluiert. Hierfür wurde Dr. Hauke Fiedler, Leiter der Space Situational Awareness (SSA)-Gruppe des DLR, hinsichtlich seiner Meinung zu den entwickelten Funktionen und den Vor- und Nachteilen einer VR-Analysesoftware im Vergleich zu den herkömmlichen Methoden befragt.

Struktur

Beginnend werden in Kapitel 2 verwandte Arbeiten aufgeführt, sodass die Forschungsfrage in den wissenschaftlichen Kontext eingeordnet wird. Nach eigener Recherche existieren bislang noch keine VR-Applikationen im Bereich der Kollisionsanalyse von Satelliten. Daher werden anstelle dessen Desktop-Anwendungen in diesem Bereich, VR-Applikationen in unterschiedlichen Bereichen der Risikoanalyse und Visualisierungssoftware für Weltraumschrott vorgestellt.

Daraufhin werden in Kapitel 3 Grundlagen, die für das Verständnis der Arbeit erforderlich sind, vermittelt. Dabei werden bspw. Datenformate und Algorithmen aus der Raumfahrt, aber auch verschiedene verwendete Koordinatensysteme erläutert.

In Kapitel 4 und 5 wird die Methodik der Arbeit dargelegt. Zunächst wird im 4. Kapitel das Konzept der VR-Anwendung beschrieben, indem geplante und umgesetzte Features im Hinblick auf ihren Nutzen bewertet und ausführlich dargelegt werden. Anschließend wird im 5. Kapitel auf deren Implementierung eingegangen.

Darauf folgend beschreibt Kapitel 6 den Aufbau und Ablauf des Experteninterviews mit Herrn Dr. Fiedler. Hier wird seine Beurteilung zu den entwickelten Funktionen zusammengefasst und im Anschluss die Forschungsfrage beantwortet. Ein Transkript des Interviews befindet sich im Anhang der Arbeit.

Ablauf und Ergebnis der Arbeit werden schließlich in Kapitel 7 zusammengefasst und diskutiert. Zuletzt wird ein Ausblick auf mögliche fortführende Arbeiten gegeben.

2. Verwandte Arbeiten

In diesem Kapitel werden verschiedene Forschungsarbeiten und Projekte zusammengefasst, die ähnliche Bereiche behandeln. Zunächst werden verschiedene Einsatzgebiete vorgestellt, in denen VR bereits als Mittel zur Risikoanalyse eingesetzt wird. Anschließend werden Desktopanwendungen vorgestellt, die derzeit für die Risikoanalyse von Satellitenkollisionen verwendet werden. Eine VR-Applikation, die eine Risikobewertung von potentiellen Satellitenkollisionen ermöglicht, gibt es zum Zeitpunkt des Schreibens dieser Arbeit nicht. Da auch eine Übersicht aller georteten Objekte im Weltall Bestandteil der in dieser Arbeit beschriebenen neu entwickelten VR-Applikation ist, werden abschließend verwandte Arbeiten hierzu geschildert.

2.1 Risikoanalyse in VR

„Combining VR Technology and Human Factors Methods for Supporting Risk Analysis“ von Salem [2008] war eine der ersten Forschungsarbeiten, welche sich mit dem Thema der Risikoanalyse in VR befasste. Zu Beginn der Arbeit wird aufgezeigt, dass VR-Anwendungen für die Sicherheits- und Risikoanalyse nicht zu den bekannten Anwendungsgebieten von VR gehören. Als mögliche Gründe hierfür werden sowohl die Knappheit und Unzulänglichkeit von VR-Systemen, aber auch das Fehlen von Anwendungen zur Erfüllung der Anforderungen und Bedürfnisse im Bereich des Arbeitsschutzes aufgeführt. Um die Entwicklung solcher VR-Applikationen voranzutreiben, werden im weiteren Verlauf funktionale Anforderungen und Funktionsspezifikationen erörtert. Dabei werden „Human Factor“-Methodiken – Methodiken, die den Einfluss fehlerhafter Eingaben des Nutzers in einem System minimieren, um eine optimale Mensch-Maschine-Schnittstelle zu ermöglichen – verwendet und angepasst. „Human Factors Methods“ werden von Cacciabue und Cacciabue [2004] in ihrem Buch „Guide to Applying Human Factors Methods“ weiter erläutert.

Puschmann et al. [2016] entwickelten eine VR-Applikation, bei der Arbeitsunfälle an Maschinen mit Hilfe virtueller Realität vermieden werden sollen. Bei der Evaluierung wurden zwei sich in der Art der Visualisierung unterscheidenden Darstellungen derselben Maschine miteinander verglichen, um die Auswirkung des Detailgrades

auf die Erkennung von möglichen Gefahrenquellen zu bewerten. Daraus ging hervor, dass der angemessene Detailgrad abhängig von der jeweiligen Gefahr ist, da zu viele Details sich ablenkend auf den Nutzer auswirken können. Im Fazit sind Puschmann et al. [2016] der Meinung, dass VR-basierte Risikobewertungen von neu entwickelten schweren Maschinen eine hervorragende Alternative zu den gängigen, weniger immersiven Methoden bilden.

Auch im medizinischen Bereich wird VR zur Evaluierung und Risikoeinschätzung verwendet. Ein Beispiel hierfür ist die VR-Applikation von Bashkanov et al. [2019], in der die Planung und somit auch die Risikoeinschätzung von Leberoperationen ermöglicht wird. Dafür wurde ein virtueller Konferenzraum entwickelt, in dem sich Mediziner mit jeweils eigenem VR-Equipment online zusammenfinden können, um sich gemeinsam u.a. ein vielschichtiges 3D-Modell der Leber des zu operierenden Patienten anzusehen. Evaluiert wurde die Software gemeinsam mit zwei Leberchirurgen, die die Software bereits in der Einzelnutzung, also ohne den Konferenzaspekt, als hilfreich bezeichneten. Die durch VR möglich gemachte räumliche Wahrnehmung der Leberstrukturen habe die Bestimmung von Resektionsbereichen erleichtert.

Neben den soeben aufgezeigten Anwendungsfällen, bei denen die Risikoanalyse innerhalb der VR-Umgebung erfolgt, gibt es auch Anwendungsgebiete, in denen VR indirekt zur Einschätzung von Risiken verwendet wird. Dabei werden spezielle Situationen in VR simuliert und das auftretende menschliche Verhalten von außerhalb bewertet. So wird beispielsweise in der Arbeit von Fromberger et al. [2018] eine VR-Applikation erwähnt, mit der sich das Verhalten von Kindesmissbrauchstätern in einer virtuellen Umgebung bewerten lässt. Im Fazit sind Fromberger et al. [2018] der Meinung, dass virtuelle Risikoszenarien Praktikern die Möglichkeit bieten, das Verhalten von Kinderschändern zu beobachten und ihre Entscheidungen bei unbeaufsichtigten Privilegien, z. B. beim Einkaufen oder unbeaufsichtigten Spaziergängen, zu testen, ohne andere Personen zu gefährden. Die Gültigkeit und Zuverlässigkeit einer solchen Analyse müsse jedoch noch weiter untersucht werden, bevor solche Applikationen im klinischen Umfeld eingesetzt werden könnten.

2.2 Weltraumschrott Risikoanalyse Software

Ein von der ESA entwickeltes und intern verwendetes Programm, welches für die Risikoanalyse von Weltraumschrott bei ihren Missionen verwendet werden kann, ist DRAMA (Debris Risk Assessment and Mitigation Analysis). Gelhaus et al. [2013] schreiben in ihrem Artikel „Upgrade of DRAMA ESA'S Space debris mitigation and analysis tool suite“ über die bereits vorhandenen und neu hinzukommenden Features von DRAMA. Darunter fallen die Berechnung von Kollisionswahrscheinlichkeiten, das Bewerten von möglichen Kollisionsschäden bei einem Zusammenprall, die Berechnung der übrigen Lebenszeit eines Satelliten im Orbit und die Simulation seines Wiedereintritts in die Erdatmosphäre. Eines der neuen Features von DRAMA ist eine graphische Benutzeroberfläche.

Ein weiteres Programm, welches für die Analyse von möglichen Satellitenkollisionen genutzt werden kann, ist FreeFlyer. Dabei handelt es sich um eine *commercial off-the-shelf* Software, welche von *a.i. solutions* entwickelt wurde. FreeFlyer beinhaltet laut a.i. solutions [2020] mehrere Methoden für den Time of Closest Approach

(TCA), der Berechnung von Kollisionswahrscheinlichkeiten und der Berechnung des Abstandes, mit denen sich zwei Objekte verfehlen. Weiterhin lassen sich die Orbits und approximierten Positionen von Satelliten berechnen und visualisieren. Neben diesen beinhaltet FreeFlyer viele weitere Funktionen für diverse Weltraummissionen und wird u.a. von der NASA, der United States Air Force und der United States Space Force verwendet.

In einem Interview mit dem Teamleiter der SSA-Gruppe des Deutsches Zentrum für Luft- und Raumfahrt (DLR) wurde unter anderem erfragt, welche Software das DLR für die Kollisionsvermeidung von Satelliten verwendet. Nach Aussage des Teamleiters ist FreeFlyer nicht verlässlich und präzise genug für die Analyse von möglichen Kollisionsereignissen, weshalb die SSA Gruppe selbst entwickelte Software wie z.B. *ODEM* und *GHOST* für die Propagation von Flugbahnen verwendet und Kollisionsberechnungen von der Flugdynamik Gruppe des DLR erhält, die ebenfalls eigens entwickelte Software verwendet. Zu den intern angewandten Systemen lassen sich nur sehr wenige Informationen finden, sodass im Nachfolgenden lediglich *GHOST* beschrieben wird.

Gps high precision orbit determination software tools (GHOST) ist ein Softwarepaket zur GPS-basierten Bahnbestimmung von Satelliten in der niedrigen Erdumlaufbahn. GHOST umfasst verschiedene Module zur Datenvorverarbeitung, kinematischen Positionierung und reduzierten dynamischen Bahnbestimmung mit Hilfe von weltraumgestützten GPS-Messungen. Satelliten-Laser-Ranging-Daten können darüber hinaus für die Bahnvalidierung verarbeitet werden. Die GHOST-Bibliothek und die Anwendungsprogramme wurden vom Deutschen Raumfahrtkontrollzentrum des DLR (DLR/GSOC) in enger Zusammenarbeit mit dem Delft Institute of Earth Observation and Space Systems (DEOS) an der TU Delft entwickelt (Zywicki [2021]).

2.3 Visualisierung von Weltraumschrott

Colombo et al. [2020] entwickelten im Rahmen einer wissenschaftlichen Arbeit eine VR-Software in Unity mit dem Ziel, die Aufmerksamkeit und das Bewusstsein zum Thema Weltraumschrott zu steigern. Innerhalb der Anwendung kann der Nutzer zu fixen Posititonen im Orbit der Erde teleportieren und dabei 3D-Modelle der 20.000 im Orbit dargestellten Satelliten betrachten. Die Anwendung soll laut Colombo et al. [2020] in den nächsten Jahren weiter auf den Nutzen in der Lehre getestet werden. Bereits durchgeführte Evaluierungen zeigten, dass die Testpersonen die Anwendung als informativ und aufregend empfanden.

Weitere VR-Anwendungen, die Weltraumschrott visualisieren, wurden bei der Recherche zu dieser Arbeit nicht gefunden, weshalb im Nachfolgenden über Web-Anwendungen geschrieben wird.

Eine sehr bekannte und leicht zugängliche Web-Anwendung ist „Stuff in Space“ (stuffin.space). Diese wurde 2015 von einem Studierenden der „University of Texas at Austin“ entwickelt. Hier erhält man einen Überblick über eine große Menge an Satelliten, welche als Partikel dargestellt werden. Diese können ausgewählt werden, woraufhin weitere Informationen und deren Orbits angezeigt werden. Weiterhin lassen sich vorgegebene Satellitengruppen auswählen, um bspw. alle GPS-Satelliten betrachten

zu können. Die verwendeten Daten werden täglich aktualisiert und entstammen der space-track.org Website.

Im Rahmen der Masterarbeit von Zarichin [2017] wurde eine weitere Web-Anwendung zur Visualisierung des Weltraumschrotts entwickelt. Diese ist vom Aufbau *stuffin.space* sehr ähnlich, jedoch werden neben den Satelliten auch die Sonne und der Mond dargestellt. Ein weiterer Unterschied ist die Herkunft der verwendeten Daten, da Zarichin [2017] hierfür Daten von BACARDI (Backbone Catalogue for Relational Debris Information) verwendet, eine vom DLR entwickelte Software, die die Positionen von bekannten Weltraumobjekten speichert und prozessiert (Stoffers et al. [2019]). Durch diese Arbeit ist es nun möglich, eine visuelle Übersicht über die in BACARDI befindlichen Daten zu erhalten. Auf der Website des DLR kann die Anwendung ausprobiert werden.[1]

LeoLabs, ein Unternehmen, welches laut eigener Aussage der erste und einzige Anbieter von kommerziellen Radarverfolgungsdiensten für Objekte im niedrigen Erdorbit ist, hat mit „Low Earth Orbit Visualization“ ebenfalls eine Web-Anwendung entwickelt, welche Daten von bekannten Weltraumobjekten visualisiert. Außerdem werden im „Low Earth Orbit Visualization“ die Bereiche gekennzeichnet, an denen die Objekte geortet werden. Ein weiteres Feature ist die farbliche Kodierung der Satelliten. So lässt sich anhand der Farbe erkennen, wie viel Zeit seit der letzten Ortung vergangen ist. Die „Low Earth Orbit Visualization“ ist öffentlich zugänglich und kann unter dem Link https://platform.leolabs.space/visualization betrachtet werden.

[1] https://www.dlr.de/sc/de/desktopdefault.aspx/tabid-12766/22301_read-51854/, letzter Zugriff: 21.08.2021

3. Grundlagen

In diesem Kapitel werden die Grundlagen dargelegt, die zum Verständnis des Konzepts und der Implementierung relevant sind. Das umschließt die Bestimmung von Satellitenpositionen, die in der Arbeit benötigten Koordinatensysteme, der Ray Marching Algorithmus und VR.

3.1 Satellitenpositionen

Die möglichst exakte Position von Satelliten bestimmen zu können, ist bei der Analyse von Satellitenkollisionen außerordentlich wichtig. Je genauer die Positionsdaten sind, desto sicherer sind die Kollisionsberechnungen, wodurch letztlich Kollisionen, aber auch unnötige Ausweichmanöver verhindert werden können. Das Bestimmen der Position kann entweder durch Messungen oder der Berechnung anhand von mathematischen Modellen erfolgen und wird in verschiedenen Datenformaten festgehalten. Näheres hierzu wird in den folgenden Abschnitten erläutert.

3.1.1 Messungen

Für das Orten und Katalogisieren von Satelliten sind je nach Nation unterschiedliche Organisationen verantwortlich, wobei als Satellit jedes Objekt gilt, welches sich im Orbit um einen Zentralkörper befindet. In den USA ist das Verteidigungsministerium für das Orten von Weltraumschrott zuständig und arbeitet gemeinsam mit der NASA an der Kategorisierung der georteten Objekte. Dabei sei das Verteidigungsministerium nach Aussage der NASA (NASA [2021]) in der Lage, mithilfe ihres „Space Surveillance Networks“ Objekte in niedriger Erdumlaufbahn mit einem Durchmesser von 5cm oder größer zu orten. Derzeit werden von der NASA ca. 27.000 Objekte katalogisiert und deren Bahninformationen größtenteils der Öffentlichkeit zugänglich gemacht. Bei der ESA und dem DLR gibt es sogenannte SSA-Gruppen, welche für das Beobachten von aktiven und inaktiven Satelliten verantwortlich sind (ESA [2021b] und Wermuth [2021]). Das größte Weltraumbeobachtungsradarsystem in Europa ist derzeit TIRA (Tracking and Imaging Radar), welches Objekte in einer Entfernung von 1000 km und einem Durchmesser von 2 cm orten kann (Mehrholz et al. [2002]). Das Frauenhofer

Institut und DLR arbeiten derzeit gemeinsam an German Experimental Space Surveillance and Tracking Radar (GESTRA), womit die deutsche SSA Gruppe in der Lage sein soll, einen Katalog mit qualitativ hochwertigen Daten für alle Objekte im Weltall zu generieren, die eine Flughöhe von weniger als 3000km und einen bestimmten, jedoch nicht genannten, minimalen Radarquerschnitt haben. Wilden et al. [2017]

Neben den zuvor genannten Radarsystemen kann Weltraumschrott auch mithilfe von Teleskopen geortet werden. Hafizoglu und Mailler [2013] schlagen eine Ortung durch eine hohe Anzahl an Teleskopen vor, um kostengünstig die Ortungskapazität zu erhöhen, die bei der stark ansteigenden Zahl von neu dazukommenden Objekten notwendig sei.

3.1.2 TLE - Two Line Elements

Two-line Elements Set (TLE) ist ein Standarddatenformat, um verschiedenste Eigenschaften eines Satelliten zu beschreiben. Technisch gesehen bestehen TLEs aus drei Zeilen, wobei die nullte Zeile den Namen des Satelliten enthält, und deswegen häufig auch mit 3LE abgekürzt wird. „Echte“ *two line elements*, die die nullte Zeile nicht enthalten, werden heutzutage meist mit „2LE“ abgekürzt. Ein Beispiel für den grundlegenden Aufbau eines TLE ist in Abb. 3.1 zu sehen.

3.1: Aufbau eines TLEs anhand eines Beispiels, Quelle: Pavur und Martinovic [2019]

Die Website space-track.org ist der momentane Hauptverteiler von TLE-Daten, die vom amerikanischen Verteidigungsministerium und dem 18th Space Control Squadron an die Öffentlichkeit weitergegeben werden. Da das TLE-Datenformat nun etwas älter ist und für eine geringere Menge an Satelliten ausgelegt war, gibt es Probleme bei der Vergabe von neuen IDs. Nur 5 Ziffern sind im herkömmlichen TLE-Format für die Satelliten-ID reserviert, wodurch nur für 99999 verschiedene Satelliten eine ID vergeben werden kann. Aufgrund der stark ansteigenden Anzahl an getrackten Objekten ist diese Menge nicht mehr ausreichend. Daher wurde das „Alpha-5“ Schema entworfen, in dem die erste Zahl mit einem Buchstaben ersetzt werden kann, um so bis zu 339999 verschiedene IDs verwalten zu können, ohne das Format mit fixer

Breite verbreitern zu müssen. Das *Alpha-5* Schema wird immer häufiger verwendet und muss beim Parsen von TLE-Daten berücksichtigt werden.

3.1.3 SGP Algorithmen

Für die Kollisionsanalyse ist es sehr wichtig, die zukünftigen Positionen von Satelliten zu berechnen, damit Kollisionen vorhergesehen werden können. Dafür bedarf es Algorithmen, die die Positionen und Geschwindigkeiten der Objekte abschätzen können. Als Beispiel kann die Vorhersage der möglichen Ankunftszeit von Autos anhand von Geschwindigkeiten und aktuellem Verkehr dienen. Bei Satelliten verhält sich dies ähnlich, bloß dass diese von verschiedenen Faktoren (z.B. anderen Himmelskörpern) beeinflusst werden. Diese Einflüsse versucht man mit vereinfachten Modellen vorherzusagen.

Im Jahr 1980 wurden von Hoots und Roehrich [1980] im Spacetrack Report No.3 mehrere verbesserte Algorithmen für das Berechnen von TLEs vorgestellt, die im weiteren Verlauf kurz erläutert werden. Außerdem werden der derzeitige Stand und ein paar verbesserte Algorithmen beschrieben, die man in Zukunft anstelle des in der Arbeit implementierten Algorithmus einsetzen könnte. Simplified Perturbations Model (SGP) ist ein Set bestehend aus 5 mathematischen Modellen: SGP, SGP4, SDP4, SGP8 und SDP8. Der wohl wichtigste Algorithmus ist SGP4, welcher für Objekte in der niedrigen Erdumlaufbahn benutzt wird. SGP4 wird auch gerne als Oberbegriff für alle SGP Algorithmen verwendet. SDP4 findet bei Objekten im sogenannten deep-space Anwendung; SGP8 und SDP8 behandeln Sonderfälle. Laut Vallado und Cefola [2012] wurde der SGP4-Algorithmus nach dessen Veröffentlichung im Jahre 1980 zum Standard bei der Verarbeitung von TLEs. Hierbei ist zu berücksichtigen, dass es nicht nur einen einzigen SGP4 Algorithmus gibt, sondern mehrere Versionen davon existieren, die unterschiedlich akkurat sind. In einer Revision des Spacetrack Report No.3 schreiben Vallado et al. [2006] über die Genauigkeit der erwähnten Algorithmen und veröffentlichen einen verbesserten Code, mithilfe dessen man die Position von Satelliten besser abschätzen können soll. Aktuell hat space-track die Version 8.0 des SGP4 Algorithmus auf ihrer Website veröffentlicht und empfiehlt, diese zu nutzen.

3.1.4 Conjunction Data Message (CDM)

Alles, worüber im Nachfolgenden berichtet wird, bezieht sich auf CCSDS 508.0-B-1 [2013], eine Empfehlung eines CDM-Standards vom Consultative Committee for Space Data Systems (CCSDS). Der tatsächliche Aufbau eines CDMs kann vom empfohlenen Standard abweichen.

Conjunction Assessment (CA) oder auch CARA (Conjunction Assessment Risk Analysis) ist das Bestreben, die Bahnen aller bekannten Objekte relativ zur operationellen Satellitenpopulation zu bestimmen, um die Entscheidungsfindung zu erleichtern, ob präventive Maßnahmen zur Vermeidung möglicher Kollisionen ergriffen werden sollten. CDM ist ein Datenformat zur Erleichterung des Austauschs zwischen Urhebern jener CA-Daten und Satellitenbesitzern/-betreibern. Dieser Austausch dient dazu, die Eigentümer/Betreiber von Satelliten über „conjunctions“, also Überschneidungen zwischen Objekten im Weltraum, zu informieren, um eine einheitliche Warnung durch

verschiedene Organisationen zu ermöglichen, die unterschiedliche CA-Techniken einsetzen. CDMs sind üblicherweise XML oder KVN Dateien und im ASCII Format kodiert. Sie enthalten Informationen einer *conjunction* zwischen zwei Objekten, wobei die Informationen aus den folgenden Punkten bestehen:

(a) Objekt1/Objekt2 Position und Geschwindigkeiten zum Zeitpunkt des TCA.

(b) Objekt1/Objekt2 Kovarianzen zum TCA in Bezug auf ein objektzentriertes Bezugssystem. Hierbei wird das Radial Tangential Normal (RTN) Koordinatensystem verwendet, welches in Abschnitt 3.2.3 erläutert wird.

(c) Die relative Position / Geschwindigkeit von Objekt2 in Bezug auf ein auf Objekt1 zentriertes Bezugsystem.

(d) Kollisionswahrscheinlichkeit der beiden Objekte.

(e) Informationen darüber, wie die vorangegangen Daten ermittelt/berechnet wurden.

Neben der Kollisionswahrscheinlichkeit ist insbesondere die Kovarianzmatrix von großer Relevanz. Mithilfe der Kovarianzmatrix lässt sich die mögliche Positionsabweichung zu einem gegebenen Konfidenzintervall berechnen. Diese Eigenschaft wird im Rahmen der Arbeit angewandt, um Ellipsoide zu generieren, die jeweils den Raum kennzeichnen, innerhalb dessen sich die Satelliten mit einer gewissen Wahrscheinlichkeit befinden. Die Implementierung hiervon wird in Abschnitt 5.1.2 beschrieben.

3.2 Koordinatensysteme

Im Nachfolgenden werden verschiedene Koordinatensysteme erläutert, die für das Verständnis dieser Arbeit notwendig sind.

3.2.1 Kartesisches Koordinatensystem

Das kartesische Koordinatensystem ist ein orthogonales Koordinatensystem, d.h., dass die Koordinatenachsen senkrecht (orthogonal) zueinander stehen. Für die Achsen (im dreidimensionellen Raum) werden überlicherweise die Variablen x, y und z verwendet. Im mathematischen Kontext liegen zumeist die x- und y-Achse in der Ebene und die z-Achse fungiert als Höhenanzeige, womit es sich um ein rechtshändiges Koordinatensystem handelt. Innerhalb der Entwicklungsumgebung *Unity*, welche im Rahmen der Arbeit verwendet wird, sind die y- und z-Achse jedoch im Vergleich zum mathematischen Standard vertauscht, da es sich hierbei um ein linkshändiges Koordinatensystem handelt. Dies muss innerhalb der Arbeit bei den angewandten mathematischen Formeln (z.B. Koordinatentransformationen) immer wieder berücksichtigt werden.

Abbildung 3.2: Polar- und Azimutwinkel (θ) und (φ) des Punktes P sowie dessen Abstand (r) vom Ursprung, Quelle: Constantin und Johnson [2017]

3.2.2 Kugelkoordinaten

In Kugelkoordinaten, oder auch sphärischen Koordinaten, wird ein Punkt im dreidimensionalen Raum mithilfe seines Abstandes vom Ursprung (r) und zwei Winkeln, dem Polarwinkel (θ) und dem Azimuthwinkel (φ), angegeben. Das Koordinatensystem wird in Abb. 3.2 illustriert. Im Verlauf der Bachelorarbeit werden Kugelkoordinaten sowohl bei der Fortbewegung des Nutzers als auch beim Berechnen der Sonnenposition relativ zum Datum und der Uhrzeit verwendet.

3.2.3 RTN Koordinatensystem

RTN ist ein am Raumfahrzeug zentriertes dreidimensionales kartesisches Koordinatensystem, dessen Achsen in Abhängigkeit vom Satelliten, seinem Zentralkörper und seines Geschwindigkeitvektors aufgespannt werden. Dabei ist R (Radial) der Einheitsvektor in radialer Richtung, der vom Zentrum des Zentralkörpers nach außen zeigt, T (Transversal) der Einheitsvektor senkrecht zum R-Vektor in Richtung der Geschwindigkeit des Satelliten und N (Normal) der Einheitsvektor senkrecht zur Inertialbahnebene (in Richtung des Satellitendrehimpulses), der das rechtshändige Koordinatensystem vervollständigt. In Abb. 3.3 werden die Achsen beispielhaft dargestellt. Die Kovarianzmatrizen der CDMs (siehe Abschnitt 3.1.4) beschreiben die Positionsabweichung von Satelliten im RTN Koordinatensystem, weshalb das RTN Koordinatensystem im Rahmen der Arbeit immer wieder verwendet wird.

Abbildung 3.3: RTN Koordinatensystem eines Satelliten

3.3 Ray Marching und Sphere Tracing

Bei Ray Marching handelt es sich laut Alex Benton [2017] um eine Alternative zu Ray Tracing. Hierbei werden endlich viele Schritte entlang eines jeden Strahls genommen, wobei die Schrittgröße konstant bleibt. Nach jedem Schritt wird überprüft, ob nun eine Kollision des Schritts mit einem Objekt stattgefunden hat. Sollte eine Kollision stattgefunden haben, wird die Distanz zurückgegeben. Mit Hilfe des Ray Marching Algortihmus werden somit die Distanzen einzelner Strahlen zu den jeweiligen Objekten in Blickrichtung der Beobachtungskamera näherungsweise berechnet, womit sich wiederum die Farbe der Pixel bestimmen lässt. Ray Marching wird bspw. beim Rendern von nicht einheitlichen Volumenmetriken oder impliziten Funktionen eingesetzt. Nach Alex Benton [2017] ist das klassische Ray Marching gut für Höhenfelder (zweidimensionale skalare Felder, die ein Höhenrelief beschreiben) geeignet, benötigt aber oftmals zu viele Schritte und kann bei einer zu groß gewählten Schrittgröße Objekte überspringen (siehe Abb. 3.4). Sphere Tracing ist eine Ray Marching Variante, welche die Performanz des klassischen Ray Marchings erheblich steigern kann (Alex Benton [2017]). Hierfür werden sogenannte Signed Distance Functions (SDFs) verwendet. Bei SDFs handelt es sich um Funktionen, die aus einer Position den Abstand von dieser Position zum nächstgelegenen Teil eines Objekts zurückgeben. So wird beim Sphere Tracing mithilfe der SDFs die kürzeste Distanz zu allen Objekten d berechnet und statt mit einer fixen Größe nun mit der Größe der Distanz d entlang des Strahls vorangeschritten und erneut die kürzeste Distanz zu allen Objekten berechnet. Dies wird so lange wiederholt, bis die Distanz entweder ein vorgegebenes Distanzmaximum überschreitet und somit als „miss“ gilt oder eine ebenfalls vorgegebene Oberflächendistanz unterschreitet und somit als „hit“ gilt (siehe

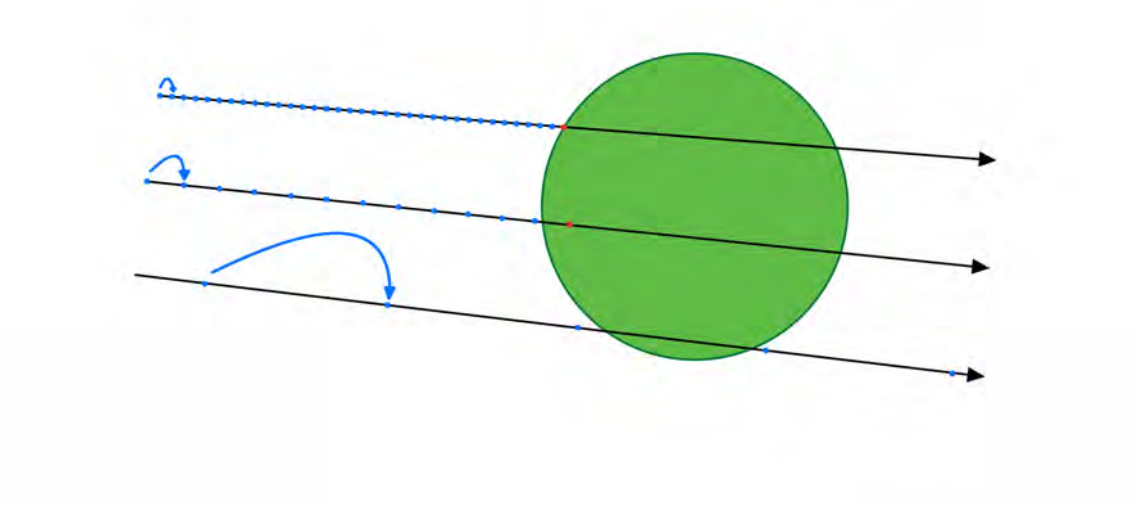


Abbildung 3.4: Ray Marching mit (von oben nach unten) einer zu geringen, einer optimal und einer zu groß gewählten Schrittgröße.

Abb. 3.5). Ein Problem von Sphere Tracing sind Strahlen, die in geringer Entfernung

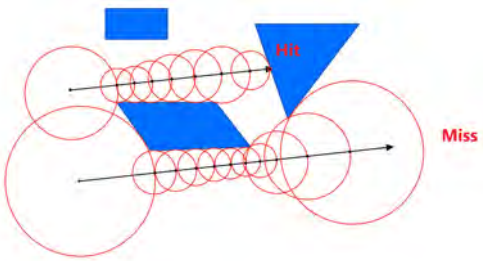


Abbildung 3.5: Sphere Tracing: Beispiel mit einem Hit und einem Miss.

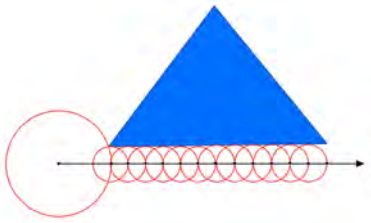


Abbildung 3.6: Sphere Tracing: Strahl verläuft parallel zum Objekt.

parallel zu einer Objektoberfläche verlaufen (siehe Abb. 3.6). Hierbei werden sehr viele kleine Schritte genommen, ohne dass es zu einer Kollision kommt. Um dieses Problem zu lösen, wird oftmals eine maximale Anzahl von Schritten festgelegt, nach deren Überschreitung abgebrochen wird und der Strahl ein „miss“ zurückgibt. Laut Hart [1994] bietet Sphere Tracing ein Werkzeug zur Untersuchung einer größeren Vielfalt impliziter Oberflächen als bisher möglich war. Im Rahmen der Bachelorarbeit wird Sphere Tracing für die Visualisierung der Überschneidung zweier Ellipsoide verwendet.

3.4 Virtual Reality

Bei VR handelt es sich um eine virtuelle Umgebung, die von einer Software geschaffen wird und dem Benutzer so präsentiert wird, dass dieser sie für eine reale Umgebung hält und akzeptiert. Hierfür werden üblicherweise Head Mounted Displays (HMDs) verwendet, die mit zwei Displays, also einem Display pro Auge, ausgestattet sind. Zwischen Display und Auge befinden sich oftmals Linsen, weshalb die Geräte auch oft als „Brille “ bezeichnet werden. Die VR-Brillen verfolgen stets die Kopfposition und -orientierung des Nutzers und drehen/bewegen die Kamera je nach Bewegung

des Nutzers. Neben dem Kopf und der Position des Anwenders können auch weitere Körperteile, wie z.B. die Hände und Füße, getrackt werden. Die nach Chavan [2016] relevantesten Anwendungsgebiete von VR sind das Fortbilden und Trainieren der Anwender in unterschiedlichen Bereichen: das Verkaufswesen, Videospiele, diverse Medien, und die Infrastrukturplanung. Außerdem habe VR großes Potential in der Wissenschaft und Medizin. In einer von Paes et al. [2017] durchgeführten Studie wurde aufgezeigt, dass die Testpersonen eine verbesserte räumliche Wahrnehmung von virtuellen Umgebungen in VR aufwiesen, wenn die virtuelle Umgebung besonders immersiv gestaltet war. Insbesondere Distanzen konnten innerhalb einer immersiven Umgebung besser abgeschätzt werden.

4. Konzept

In diesem Kapitel werden sowohl die initialen Konzepte als auch das schließlich umgesetzte Konzept erläutert. Dabei wird auf die verschiedenen Funktionen eingegangen, die bei der Analyse von möglichen Satellitenkollisionen helfen sollen.

4.1 Initiale Konzepte

Am 04.02.2021 fand eine Zwischenpräsentation der Arbeit statt, in der verschiedene Konzepte vorgestellt und mit den Vorstellungen des DLR abgeglichen wurden. In diesem Abschnitt werden die vorgeschlagenen Konzepte und im nachfolgenden Abschnitt schließlich das Konzept der Implementierung ausführlich erläutert. Innerhalb der Zwischenpräsentation wurden zunächst die Projektdetails inklusive der geplanten Features geschildert und anschließend drei Herausforderungen sowie deren Lösungsansätze vorgestellt.

Die geplanten Features der VR-Applikation waren die folgenden:

- Berechnung der Position und des Orbits von Satelliten anhand von TLE-Daten,
- Zeitgleiche Visualisierung aller Satelliten,
- Realistische Szenerie,
- Fortbewegung des Nutzers im Raum,
- Manipulation der Zeitachse,
- Filtern von Satelliten,
- Durchsuchen und Auswählen von CDM und
- Analysieren von CDM-Warnungen.

Details zu diesen Features werden in Abschnitt 4.2 beschrieben. Die zuvor erwähnten Herausforderungen und Lösungsansätze werden nachfolgend beschrieben.

4.1.1 Analysieren von Kollisionswarnungen

Messungen und das Berechnen von Positionen von Satelliten gehen mit Ungenauigkeiten einher, daher müssen diese Abweichungen in der Analyse berücksichtigt werden. Hierfür wurde vorgeschlagen, das CDM-Datenformat bzw. dessen Kovarianzmatrizen zu verwenden. Aus der Berechnung der Abweichung mithilfe der Kovarianzmatrizen entstehen Ellipsoide, die abhängig vom gewählten Quantil unterschiedlich groß sind und die Bereiche darstellen, innerhalb derer sich die Satelliten mit einer gewissen Wahrscheinlichkeit befinden. Diese Ellipsoide sollten dann, wie in Abb. 4.1 zu sehen, mit einem Wireframe-Shader visualisiert werden. Da die Ellipsoide lediglich

Abbildung 4.1: Präsentierte Skizze der Positionsabweichungsellipsoide.

die Positionsabweichung zu einem gegeben Zeitpunkt aufzeigen, nämlich zu dem Zeitpunkt, für den die Kovarianzmatrizen berechnet wurden, wurde weiterhin ein „Positionsabweichungs-Trichter“ vorgeschlagen, der die mit der Zeit wachsenden Ungenauigkeit der Positionsberechnung visualisieren sollte (siehe Abb. 4.2). Diese Idee wurde verworfen, weil die Abweichung zur tatsächlichen Position bei jedem Pertubationsalgorithmus unterschiedlich stark und nur schwer approximierbar ist. Eine weitere Idee war die farbliche Hervorhebung der Überschneidung beider Ellipsoide, wie in Abb. 4.3 dargestellt. Zuletzt wurde ein Modus vorgeschlagen, in dem sich die Satelliten in einem wiederholenden Zeitraffer immer wieder von *kurz vor* bis *kurz nach* dem TCA begegnen.

4.1.2 Realistische Repräsentation von Satelliten

In diesem Bereich wurden die Problematiken erläutert, die auftreten würden, wenn man alle Objekte maßstabsgetreu darstellen würde. Dazu gehörte auch folgendes technisches Problem: Hat ein Würfel in der VR-Umgebung eine interne Skalierung (1, 1, 1), so wird dieser in VR als Würfel mit ein Meter langen Kanten wahrgenommen.

Abbildung 4.2: Skizze von der geplanten Darstellung des „Abweichungs-Trichters“.

Befindet sich die VR-Brille an der Position (1, 0, 0), würde der Nutzer den Eindruck erhalten, einen Meter vom Ursprung entfernt zu sein. Unter der Annahme, dass das Zentrum der Erde im Koordinatenursprung und ein Satellit sich in erdnahem Orbit (bspw. 1000km Höhe) über dem Nordpol befindet, so wäre dieser an der Position (0, 7371000, 0). Da die jeweiligen Koordinaten der Position innerhalb von Unity ausschließlich als Gleitkommazahl gespeichert und verarbeitet werden können, würde solch eine Position das Problem Namens „spatial jitter“ herbeiführen. Dabei beginnt die Kamera aufgrund der Gleitkommazahl Rundungsfehler zu zittern Thome [2005]. Aus diesem Grund wurde eine 1:10.000 Skalierung vorgeschlagen. Mit dieser Skalierung ist es möglich, geostationäre Satelliten zu untersuchen, ohne dass *spatial jitter* auftritt. Weiter entfernte Satelliten sind kaum vorhanden und aufgrund ihrer geringen Anzahl und damit einhergehenden geringen Kollisionsanalyse nicht relevant. Neben der 1:10.000 Skalierung wäre eine weitere Reduktion der Skala möglich, jedoch nicht sinnvoll, da Abstände zwischen Satelliten hierdurch nur noch kleiner ausfallen würden und eine mögliche Kollision damit noch schwieriger zu erkennen wäre.

Eine weitere genannte Herausforderung war die gleichzeitige Darstellung aller Objekte. Hierfür wurden mehrere mögliche Lösungen vorgestellt:

- Lösung 1: Nur die nächsten X Satelliten darstellen
- Lösung 2: Partikeldarstellung
- Lösung 3: Partikel + selektierte Objekte darstellen
- Lösung 4: Ausschließlich gefilterte Objekte werden dargestellt
- Lösung 5: Kombination der vorangegangenen Lösungen

Die bevorzugte und schließlich umgesetzte Lösung besteht darin, mehrere Lösungen zu kombinieren. Ein 3D-Modell ist bei der Analyse von Satelliten sogar hinderlich,

Abbildung 4.3: Skizze der farblichen Überschneidungsmenge.

da das Aussehen der meisten Satelliten nicht bekannt ist und ein zu großes/kleines Modell dazu führen könnte, dass eine Kollision fälschlicherweise vorher- bzw. nicht vorhergesagt wird. Für die allgemeine Betrachtung der Objekte im Orbit um die Erde können die 3D-Modelle zu einer erhöhten Immersion führen. Jedoch reicht in den meisten Fällen eine Partikeldarstellung aus. Mehr zur Umsetzung befindet sich in Abschnitt 4.2.3.

4.1.3 Weitere Ideen

Während der Diskussionsrunde der Zwischenpräsentation wurden weitere Ideen und Vorschläge zur Implementierung geäußert. Eine Idee war das Clustern. Neben dem Zusammenführen von Orbits könnte eine sogenannte Region of Interest (ROI) definiert werden, wobei nur in der ROI Orbits, Modelle und weitere Details der Satelliten angezeigt würden. Die ROI könnte ein dreidimensionaler Bereich sein, der untersucht wird, ein definierter Zeithorizont (Analyse eines Zeitintervalls), oder sie könnte automatisch aus der Position und Blickrichtung des Nutzers ermittelt werden.

Eine weitere vorgeschlagene Idee bestand darin, die Möglichkeit zu erhalten, nach der Auswahl von Kollisionsobjekten diese aus der Sicht des Satelliten zu betrachten.

Der letzte Vorschlag war es, Satelliten, die sich in der Simulationsumgebung näher kommen, farblich hervorzuheben.

4.2 Umgesetztes Konzept

Die realisierte Anwendung „VRSpaceDebris“ besteht aus zwei verschiedenen Bereichen, einem Hauptbereich und einem Analysebereichs. Innerhalb des Hauptbereichs erhält der Nutzer einen Überblick über alle Satelliten. Der Analysebereich enthält

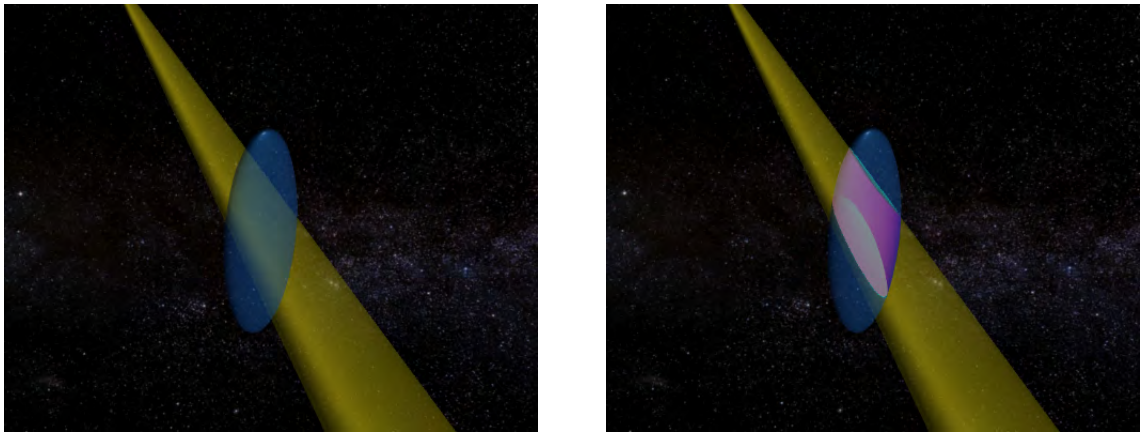
Abbildung 4.4: Skizze der Informationsanzeige der Satelliten.

ein Beispielszenario, an dem eine Kollisionswarnung analysiert werden kann. Der Anwender hat jederzeit die Möglichkeit, zwischen den beiden Bereichen zu wechseln. Die Aufteilung in zwei verschiedene Bereiche ermöglicht dem Nutzer nach dem Hinzufügen weiterer Datensätze beliebige CDMs zu analysieren und zwischen diesen zu wechseln.

4.2.1 Berechnung von Position und Orbits

Eine wichtige Voraussetzung für das Analysieren von möglichen Kollisionsereignissen besteht darin, die Positionen sämtlicher Satelliten zu beliebigen Zeitpunkten möglichst genau zu bestimmen. Dafür empfiehlt es sich, immer möglichst aktuelle Messdaten zu verwenden, da die berechnete Position mit voranschreitender Zeit immer mehr von der tatsächlichen Position abweicht (siehe Abschnitt 3.1.3). In der TLE-Sammlung, die in dieser Arbeit verwendet und von space-track bereitgestellt wurde, befinden sich knapp über 20.000 Objekte, die verarbeitet werden müssen. Um Zeit beim Starten der VR-Anwendung zu sparen, werden die Berechnungen der Positionen und Orbits der 20.000 Objekte multithreaded durchgeführt.

Zum genaueren Verständnis werden abschließend noch Details des gewählten Propagators (Lösungsfunktionen bestimmter partieller Differentialgleichungen, die in diesem Fall die Positionen von Satelliten von x nach y propagieren) erläutert. Hierbei handelt es sich ausschließlich um den SGP4 Algorithmus. Um möglichst akkurate Ergebnisse zu erzielen, wären die fehlenden SGP Algorithmen SDP4, SGP8 und SDP8 erforderlich, welche die Positionen von „Deep Sky“ Objekten sowie die Sonderfälle (z.B. Satelliten mit absinkendem Orbit) behandeln. Für einen konzeptionellen Beweiß reicht der SGP4 Algorithmus aus. Bei einem realen Gebrauch sollte der verwendete Algorithmus mit aktuellen Pertubationsmodellen ausgetauscht werden,



(a) Ohne Ray Marching Visualisierung. **(b)** Mit Ray Marching Visualisierung.

Abbildung 4.5: Überschneidung zweier Poisitionsabweichungsellipdoide innerhalb von *VRSpaceDebris.*

z.B. mit dem *SGP4 propagator v8.0*, welcher am 09.11.2020 auf der space-track Website veröffentlicht wurde.

4.2.2 Positionsabweichungen und Überschneidungen darstellen

Im vorangegangen Kapitel wurde die Berechnung der Position von Satelliten thematisiert. Da bei den SGP Modellen - wie auch schon am Namen zu erkennen - Vereinfachungen vorgenommen werden, kann die errechnete Position deutlich von der tatsächlichen Position abweichen. Diese Abweichungen werden adäquat visualisiert, um eine möglichst gute Einschätzung der Kollisionswahrscheinlichkeit zu ermöglichen. Dafür müssen zunächst die Kovarianzmatrizen der CDMs ausgewertet werden. Mit deren Hilfe und einem geeigneten Quantil können die Ausdehnungen des Fehlerellipsoids berechnet werden. Außerdem bedarf es zweier Rotationen, die Rotation des kartesischen Koordinatensystems hin zum RTN System und die Rotation des Objektes innerhalb des RTN Koordinatensystems. Weiterhin müssen die Vektoren der RTN-Achsen anhand der Position des Objekts und dessen Geschwindigkeitsvektors bestimmt werden, wobei die Positionen und die Geschwindigkeiten der Objekte bereits in den CDMs im ECEF System (x,y,z) angegeben werden.

Das Betrachten von möglichen Überschneidungen der Positionsabweichungsellipsoide ist ein wesentlicher Bestandteil in der Analyse von möglichen Kollisionsereignissen. Je größer das Überschneidungsvolumen im Verhältnis zum restlichen Volumen der Ellipsoide, desto größer ist die Kollisionswahrscheinlichkeit zum gegebenen Zeitpunkt. Außerdem kann hierdurch festgestellt werden, in welche Richtung am ehesten ausgewichen werden sollte. Daher werden diese Überschneidungen innerhalb von VRSpaceDebris farblich hervorgehoben. Ohne diese farbliche Hervorhebung ist es sowohl bei einer Desktop Anwendung als auch in VR herausfordernd, die tatsächlichen Ausmaße der Überschneidung zu erahnen (siehe Abb. 4.5).

4.2.3 Szenerie

Im Zentrum der *MainScene* befindet sich ein 3D-Modell der Erde, welche von dem Abbild der Milchstraße umrahmt wird. Um dem Anwender der Software eine

Möglichkeit zu geben, die Veränderung von Zeit wahrzunehmen, wird neben der veränderten Position der Satelliten auch die Erde - in Abhängigkeit von Zeit und Datum - korrekt von der Sonne beleuchtet. Eine Möglichkeit, dies umzusetzen, wäre das Sonnensystem nachzubauen. Für ein sehr realistisches Verhalten müsste sich die Erde alle 365,25 Tage um die Sonne und die Erde sich alle 23 Stunden, 56 Minuten und 4 Sekunden um ihre Rotationsachse drehen. Außerdem müsste die Rotationsachse um etwa 23,4° geneigt sein, wobei diese Neigung alle 41.000 Jahre zwischen 22,1° und 24,5° oszilliert. Des Weiteren müssten sich alle Satelliten und die VR Position des Anwenders korrekt mitbewegen und rotieren. Der Aufwand hierfür würde den Nutzen übersteigen, weshalb eine erdzentrierte Lösung geeigneter ist und verwendet wird. Zur Darstellung von Satelliten werden standardmäßig Partikel verwendet und lediglich bei einem Besuch die 3D-Modelle angezeigt. Die Anzeige aller 3D-Modelle der Satelliten zugleich ist nicht notwendig, da dies bei einer Übersicht eher hinderlich ist und eine unnötige Belastung der Grafikkarte mit sich ziehen würde. Dies wurde bereits in Abschnitt 4.1.2 thematisiert. Wie ebenfalls in Abschnitt 4.1.2 geschildert, werden Satelliten, die Erde und sämtliche Positionen und Abstände mit einer 1:10.000 Skalierung dargestellt. Allerdings nur in der *MainScene*. Innerhalb der *CDMExampleScene* wird ein Verfahren Namens „Floating Origin“ eingesetzt. Hierbei befindet sich der Nutzer stets im Koordinatenursprung. Sollte sich der Nutzer fortbewegen, so werden in Wirklichkeit alle anderen Objekte um die bewegte Entfernung verschoben, während der Nutzer weiterhin im Koordinatenursprung bleibt. Visuell ist kein Unterschied zu der herkömmlichen Bewegung der Beobachtungskamera sichtbar, allerdings löst *Floating Origin* das in Abschnitt 4.1.2 beschriebene *spatial jitter* Problem. Für die *CDMExampleScene* wird ein Maßstab von 1:1.000 verwendet. Ein größerer Maßstab wäre durch *Floating Origin* zwar möglich, aber aufgrund der Größe der Positionsabweichungsellipsoide nicht sinnvoll. Diese sind in dem Beispielszenario zwischen 300m und 11km lang und wären für den Nutzer nur schwer zu überschauen. Durch den 1:1.000 Maßstab sind die Ellipsoide zwischen 30 cm und 11 m groß. Dies ermöglicht neben der Überschaubarkeit auch eine einfache Umrechnung in die Originalgröße (1m entspricht 1km), wodurch die Größeneinschätzung vereinfacht werden soll.

4.2.4 Interaktionsmöglichkeiten

Die Interaktion mit *VRSpaceDebris* erfolgt mithife von zwei HTC Vice Controllern. *VRSpaceDebris* kann mit einem Controller bedient werden, da beide Controller die gleiche Tastenbelegung erhalten. Hierdurch ist *VRSpaceDebris* sowohl für Links- als auch Rechtshänder geeignet. Die möglichen Interaktionen werden nachfolgend aufgelistet:

- in Kugelkoordinaten durch die Szene reisen,
- Menü öffnen und schließen,
- Fortbewegungsgeschwindigkeit anpassen und
- In einen anderen Bereich wechseln.

Innerhalb des Hauptbereichs:

- Satelliten suchen und besuchen,

- nach Satelliten filtern,
- Simulationsgeschwindigkeit anpassen und
- Performanzoptimierungen vornehmen.

Innerhab des Analysebereichs:

- Konfidenzintervalle der Ellipsoide einstellen,
- RTN-Achsen ein- und ausschalten,
- zwischen verschiedenen Analysemodi wechseln und
- viele weitere Interaktionen je nach gewähltem Analysemodus.

In anschließenden Abschnitten werden diese Interaktionen näher erläutert. Die Implementierung und einige Bilder zu diesen Funktionen werden in Abschnitt 5.3 dargelegt.

4.2.4.1 Allgemeine Interaktionen

In diesem Abschnitt werden alle Funktionen erläutert, die unabhängig von der Szene vorhanden sind.

Die erste Funktion ist die Möglichkeit, sich im Weltraum fortzubewegen. Dabei erfolgt die Bewegung des Nutzers in Kugelkoordinaten, indem dieser entweder die beiden Winkel verändert und sich somit kreisförmig um die Erde bewegt oder den Radius und somit die Distanz zur Erde verändert. Dies ist keine herkömmliche Technik, da für (VR-)Bewegungen im dreidimensionalen Bereich üblicherweise eine Bewegung in Blick- oder Controllerrichtung verwendet wird. Die Fortbewegung mit Kugelkoordinaten wurde ausgewählt, da hiermit das Verfolgen von Satelliten entlang ihrer ellipsenförmigen Flugbahnen vereinfacht wird. Außerdem lässt sich die Fortbewegungsgeschwindigkeit auf verschiedene Arten verändern, damit der Nutzer sowohl schnell einen Überblick erhalten kann, aber auch seine Position präzise bestimmen kann, wenn dieser das Geschehen von einer bestimmten Position aus betrachten möchte. Der Abrufort vieler weiterer Funktionen ist das Menü, welches vom Nutzer zu jeder Zeit geöffnet und geschlossen werden kann. Sobald das Menü geöffnet wird, erscheint ein Laserstrahl aus dem zuletzt verwendeten Controller, mit dem man das Menü bedienen kann. Je nach Bereich befinden sich im Menü unterschiedliche Funktionen, jedoch gibt es in beiden Bereichen die Möglichkeit, zum jeweils anderen Bereich zu wechseln.

4.2.4.2 MainScene Interaktionen

Die *MainScene* soll dem Anwender von *VRSpaceDebris* einen Überblick über sämtliche Satelliten geben, die in den gewählten Daten vorhanden sind. Um dies zu ermöglichen, befindet sich im Menü eine vollständige Liste aller Satelliten. Diese Liste kann mittels Suchleiste anhand vom Namen oder der ID der Satelliten durchsucht werden. Zur Eingabe dient eine Tastatur, die erscheint, sobald die Suchleiste angeklickt wurde. Hierbei wird bereits nach Teilwörtern gesucht, sodass ggf. Zeit bei der Eingabe des vollständigen Namens gespart werden kann. Die in der Liste befindlichen Satelliteneinträge können angeklickt/ausgewählt werden, woraufhin weitere

Informationen zum Satelliten erscheinen. Außerdem kann der ausgewählte Satellit „besucht“ werden, wodurch der Nutzer zu dem Satelliten teleportiert wird und nun ein 3D-Modell des Satelliten sowie dessen Orbit gezeigt werden. Neben der Suche nach einzelnen Satelliten können diese auch nach bestimmten Gruppen gefiltert werden. Dabei werden nur noch die zu der Filtergruppe gehörigen Satelliten angezeigt und es können ihre Orbits ein- und ausgeblendet werden. Diese Funktion soll ebenfalls bei der Übersicht helfen und ist insbesondere dann von Nutzen, wenn nur Satelliten von bestimmten Betreibern beobachtet werden sollen. Die vorgegebenen Filter lassen sich kombinieren, sodass mehrere Filtergruppen zur selben Zeit angezeigt werden können. Schaltet der Nutzer alle Filter aus, so werden wieder alle Satelliten sichtbar. In den Optionen der *MainScene* kann der Nutzer die Simulationsgeschwindigkeit einstellen und dadurch die Fluggeschwindigkeit aller Satelliten beschleunigen oder verlangsamen. Mit der Zeit verändert sich auch die Beleuchtung der Erde von der Sonne, um die vergangene Zeit widerzuspiegeln. Zuletzt kann der Anwender einen Wert names „batch_size“ verändern, welcher die Auslastung der CPU beeinflusst. Damit der Anwender den optimalen Wert finden kann, wird neben dem Wert die aktuelle Bildwiederholrate angezeigt. Da dies technische Hintergründe hat, wird die „batch_size“ im Implementierungskapitel näher erläutert. Der Sinn des einstellbaren Werts ist es zu gewährleisten, dass der Nutzer auf unterschiedlichen Geräten *VRSpaceDebris* ohne Bildverzögerungen verwenden kann und somit weniger unter „Motion Sickness“ leidet.

4.2.4.3 CDMExampleScene Interaktionen

In der *CDMExampleScene* steht das Analysieren einer Kollisionswarnung anhand der in der CDM befindlichen Daten im Vordergrund. Hierfür kann der Nutzer im Menü einige Informationen zu den Objekten und der Gesamtsituation ablesen, wie z.B. den TCA und die Kollisionswahrscheinlichkeit. Auch in der *CDMExampleScene* gibt es ein Optionsmenü. Dort lässt sich unter anderem die Anzeige der RTN-Achsen der Objekte ein- und ausschalten, wodurch der Nutzer eine Orientierungshilfe erhält, da bspw. mithilfe der T-Achse die ungefähre Flugrichtung der Objekte abgeleitet werden kann. Weiterhin kann der Nutzer das Konfidenzintervall der Positionsabweichungsellipsoide verändern, wodurch für unterschiedliche Wahrscheinlichkeiten untersucht werden kann, ob sich die Ellipsoide überschneiden, und somit eine Kollision auftreten könnte. Die Überschneidung der Ellipsoide wird wie in Abschnitt 4.2.2 beschrieben visualisiert. Ohne diese Visualisierung ist es kaum erkennbar, ob die Ellipsoide sich tatsächlich überlappen oder lediglich hinteinander liegen, ohne diese von allen Seiten zu betrachten (siehe Abb. 4.5). Zuletzt können in den Optionen drei verschiedene Analysemodi ausgewählt werden. Im ersten Modus „Explore TCA“ befinden sich beide Ellipsoide im TCA. Dieser Modus ist standardmäßig ausgewählt. Im zweiten Modus „Timelapse Loop“ befinden sich beide Ellipsoide in einer Zeitschleife mit vom Nutzer beliebig auswählbarem Start- und Endzeitpunkt. So kann der Nutzer immer wieder das Aneinander-Vorbeifliegen der beiden Ellipsoide beobachten. Im dritten und letzten Modus „Time Manipulation“ kann der Nutzer einen beliebigen Zeitpunkt kurz vor oder nach dem TCA einstellen und somit sehen, wo sich die Ellipsoide zu einem gegeben Zeitpunkt befinden. Außerdem kann er in diesem Modus die Ellipsoide mit dem nun auch außerhalb des Menüs sichtbaren Laserstrahl beliebig entlang ihrer Richtungsvektoren verschieben und erhält den zu der Position gehörigen Zeitpunkt.

Dadurch kann der Nutzer die Ellipsoide zu ihrer Überschneidung verschieben und erhält sofort den Zeitpunkt der Überschneidung und damit möglichen Kollisionszeitpunkt der beiden Satelliten. Der TCA ist nämlich lediglich der Zeitpunkt, an dem sich die beiden Objekte laut Vorhersage am nächsten sind und nicht automatisch der Zeitpunkt, an dem sich ihre beiden Positionsabweichungsellipsoide überschneiden.

5. Implementierung

Der im Rahmen der Arbeit entwickelten VR-Applikation wurde der Name *VRSpace-Debris* verliehen und wird im Verlauf dieses Kapitels mit *VRSpaceDebris* referenziert. *VRSpaceDebris* wurde in Unity Version 2020.1.16f1 für die HTC Vive im Auftrag des DLR entwickelt und basiert auf Vorarbeiten von Patrick Saalfeld. Diese Vorarbeiten beinhalteten das Berechnen der Positionen und Orbits der Satelliten, was in Abschnitt 5.1.1 beschrieben wird, und zwei 3D-Modelle – die Modelle, die für die Satelliten und Trümmer verwendet werden.

5.1 Datenverarbeitung

5.1.1 Berechnung von Positionen und Orbits

Die Positionen aller in den TLEs befindlichen Objekte werden mittels der C#-Bibliothek one_Sgp4[2] berechnet. Die Multithreading Berechnungen werden mithilfe des Unity-Assets Thread Ninja[3] durchgeführt. Damit ist es möglich, sogenannte Koroutinen, die nativ in Unity vorhanden sind, aber normalerweise singlethreaded durchgeführt werden, multithreaded abarbeiten zu lassen. Zuerst werden die TLEs aus einer JSON Datei geparst und an den one_sgp Propagator weitergegeben. Dabei werden nicht-geparste TLEs abgefangen und protokolliert. Nun kann die Funktion „runSgp4Cal“ des Propagators aufgerufen werden, die die Eingabeparameter Start- und Stopzeit sowie den Zeitschritt in Minuten beinhaltet. Als Startzeit fungiert die Systemzeit. Die Stopzeit setzt sich aus der Addition von der Startzeit mit der Umlaufzeit T zusammen, wobei T mithilfe der mittleren Bewegung n, welche die durchschnittliche Winkelgeschwindigkeit eines Objekts auf einer elliptischen Umlaufbahn beschreibt, und sich aus der zweiten Zeile des TLE entnehmen lässt, wie folgt berechnet werden kann:

$$T = \frac{2 \cdot \pi}{n}$$

[2] https://github.com/1manprojects/one_Sgp4, letzter Zugriff: 29.03.2021
[3] https://assetstore.unity.com/packages/tools/thread-ninja-multithread-coroutine-15717

Diese Umlaufzeit wird in Minuten umgerechnet und durch eine beliebige natürliche Zahl "steps" geteilt, um so "steps"-viele Punkte auf der Umlaufbahn des Objekts zu erhalten. Je höher steps ist, desto besser wird der Orbit approximiert, wobei die Berechnungszeit linear dazu zunimmt. Am Ende werden alle Positionen des Objekts in ein Array gespeichert. Dieses kann jederzeit von anderen Skripten ausgelesen werden.

5.1.2 RTN Kovarianzen

Sowohl die Ausdehnung als auch die Rotation des Ellipsoids innerhalb des RTN-Koordinatensystems lassen sich mithilfe der Eigenzerlegung der Kovarianzmatrix bestimmen. Dabei legen die Eigenvektoren die Ausrichtung der R, T und N Achsen fest, und die Eigenwerte werden für die Ausdehnung in die jeweiligen Richtungen benötigt. Da die Kovarianzmatrix symmetrisch ist, sind die Eigenvektoren orthogonal zueinander, solange ihre Eigenwerte unterschiedlich sind. Sollten die Eigenwerte gleich groß sein, würde es sich um eine Kugel handeln. Zunächst wird Unitys Koordinatensystem so verwendet, als sei es bereits das fertige RTN Koordinatensystem, also

$$R \mapsto x, T \mapsto y, N \mapsto z$$

Daraufhin wird die Ausdehnung des Ellipsoids auf den jeweiligen Achsen mithilfe der jeweiligen Eigenwerte λ_i und eines geeigneten Quantils c^2 der Chi-Quadrat Verteilung χ^2 wie folgt berechnet:

$$width_i = 2 \cdot \sqrt{c^2 \cdot \lambda_i}$$

Hierbei wird $c^2 = 12,84$ verwendet, also das χ^2 Quantil mit 3 Freiheitsgraden und einer Konfidenz von $99,5\%$. D.h., dass der Satellit sich mit $99,5$ prozentiger Wahrscheinlichkeit innerhalb des errechneten Ellipsoids befindet. Das Quantil kann beliebig ausgetauscht werden, um Ellipsoide mit verschiedenen Konfidenzen zu erhalten. Die ermittelten Werte werden an ein in Abschnitt 5.2.4 beschriebenes Skript übergeben, welches prozedural das Ellipsoid mit den übergebenen Ausdehnungen am Koordinatenursprung $(0, 0, 0)$ generiert. Anschließend muss das Ellipsoid innerhalb des RTN Koordinatensystems nach den Eigenvektoren $\vec{v_i}$ rotiert werden. Wie bereits erwähnt, bestimmen die Eigenvektoren die Rotation des Ellipsoids auf dem RTN Koordinatensystem, wobei momentan noch angenommen wird, dass das kartesische Koordinatensystem bereits das korrekte RTN System ist. Diese Vektoren $\vec{x}(1, 0, 0)$, $\vec{y}(0, 1, 0)$ und $\vec{z}(0, 0, 1)$ müssen in Richtung $\vec{v_1}$,$\vec{v_2}$ und $\vec{v_3}$ rotiert werden. Dafür berechnet man die Rotationsmatrix R wie folgt, wobei $\otimes$ das dyadische Produkt zweier Vektoren repräsentiert:

$$R = \vec{x} \otimes \vec{v_1} + \vec{y} \otimes \vec{v_2} + \vec{z} \otimes \vec{v_3}$$

Zuletzt bestimmt man die Rotation mithilfe der ersten und zweiten Spalte von R. Die nullte Zeile wird von Unity nicht verlangt, da diese bloß das Kreuzprodukt aus den anderen beiden Vektoren ist. Dies sieht in Unity wie folgt aus:

$$rotation = Quaternion.LookRotation(R.GetColumn(2), R.GetColumn(1));$$

Hierbei bestimmt der erste Parameter den sogenannten Vorwärtsvekor und der zweite Parameter den Aufwärtsvektor. Darüber hinaus müssen noch die tatsächlichen R, T

und N Vektoren bestimmt und das Objekt zu diesen Vektoren hin rotiert werden. $\vec{R}$ ist der Einheitsvektor in radialer Richtung, der vom Zentrum der Erde nach außen zeigt, und weil das Zentrum der Erde in *VRSpaceDebris* im Koordinatenursprung liegt, lediglich der Einheitsvektor des Ortsvektors $\overrightarrow{OP}$ zur Position des Satelliten $P(x, y, z)$.

$$\vec{R} = \frac{\overrightarrow{OP}}{\left\|\overrightarrow{OP}\right\|}$$

$\vec{T}$ ist der Einheitsvektor senkrecht zu $\vec{R}$ in Richtung des Geschwindigkeitsvektors $\vec{v}$ des Satelliten. Sollte der Satellit sich bei der Apoapsis, der größten Entfernung zum Zentralkörper, oder der Periapsis, der geringsten Entfernung zum Zentralkörper, befinden, so ist der Geschwindigkeitsvektor $\vec{v}$ bereits $\vec{T}$:

$$\vec{T} = \vec{v}$$

Für alle anderen Fälle wird das Gram-Schmidtsche Orthogonalisierungsverfahren verwendet, um $\vec{v}$ hin zu $\vec{R}$ zu orthogonalisieren. Die Berechnung sieht wie folgt aus:

$$\vec{T'} = \vec{v} - \frac{\langle \vec{R}, \vec{v} \rangle}{\langle \vec{R}, \vec{R} \rangle} \cdot \vec{R}$$

$$\vec{T} = \frac{\vec{T'}}{\left\|\vec{T'}\right\|}$$

Der Normalenvektor $\vec{N}$ lässt sich mithilfe des Kreuzproduktes von $\vec{R}$ und $\vec{T}$ bestimmen.

$$\vec{N} = \vec{R} \times \vec{T}$$

Nachdem $\vec{R}$, $\vec{T}$ und $\vec{N}$ bestimmt wurden, fehlt noch die Rotation vom kartesischen System in den RTN-Koordinatenrahmen. Hierfür wird erneut eine Rotationsmatrix R auf dieselbe Art und Weise wie bei den Eigenvektoren erzeugt und die Rotation zu der bereits vorhanden Rotation addiert. Diese Addition erfolgt mittels Multiplikation der jeweiligen Quaternionen in der korrekten Reihenfolge, da die Multiplikation von Quaternionen nicht symmetrisch ist:

$$rotation := newRotation * oldRotation;$$

5.1.3 Sonnenpositionsberechnung

Zur Berechnung der Sonnenposition wird der Code von Mikael[4] verwendet, mit dessen Hilfe zu beliebiger Zeit der Azimut und die Altitude der Sonne aus der Sicht eines gegebenen Längen- und Breitengrades berechnet werden. Bei dem Code von Mikael handelt es sich lediglich um ein vereinfachtes Modell, welches aber laut Eigenaussage bis zum Jahr 2100 zufriedenstellende Ergebnisse liefern soll. Mit dem Azimut- und Polarwinkel sind zwei der drei Komponenten der Kugelkoordinaten vorhanden, wobei

[4] http://guideving.blogspot.com/2010/08/sun-position-in-c.html, letzter Zugriff: 26.03.2021

der Polarwinkel θ der Komplementärwinkel der Altitude ist und sich daher wie folgt bestimmen lässt:

$$\theta = \frac{\pi}{2} - altitude$$

Die allgemeine Umrechnungsformel von Kugelkoordinaten mit dem Radius r in ein kartesisches Koordinatensystem sieht wie folgt aus:

$$\begin{aligned} x &= r \cdot \sin\theta \cdot \cos\varphi \\ y &= r \cdot \sin\theta \cdot \sin\varphi \\ z &= r \cdot \cos\theta \end{aligned}$$

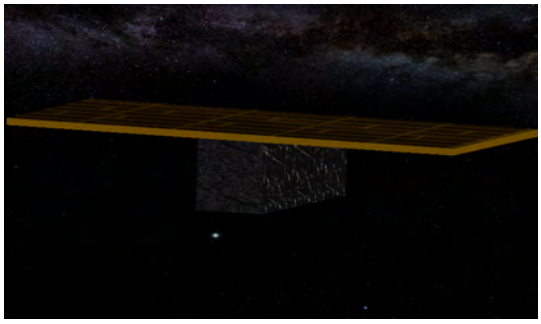
Da für die Beleuchtung der Erde nicht die Position der Sonne, sondern lediglich der zu der Sonne zeigende Richtungsvektor $\vec{v}$, bzw. dessen Gegenvektor $\vec{g}$ benötigt werden, kann für r ein beliebiges $z \in \mathbb{R}, z > 0$ gewählt werden. Dieser verändert nur die Magnituden von $\vec{v}$ und $\vec{g}$. Bei der Anwendung der Gleichung müssen zwei Aspekte beachtet werden. Zum einen beziehen sich die Koordinaten im kartesischen System auf einen Koordinatenrahmen, der vom Beobachtungspunkt aufgespannt wird, und zum anderen sind die y- und z-Achsen in Unity vertauscht.
Da die Null Island des verwendeten Erdmodells nach der x-Achse und die Rotationsachse nach der z-Achse ausgerichtet wurde und bei der Berechnung von Azimut und Anglitude der Längen- und Breitengrad der Null Island (0, 0) verwendet wird, ist es ausreichend, die Achsen in der Berechnung zu vertauschen. Andernfalls müsste die Rotationsmatrix vom verwendeten Koordinatensystem hin zu dem Koordinatenrahmen aus der Sicht des Beobachtungspunktes berechnet werden.
Die Berechnungen in Pseodocode sehen also wie folgt aus:

```
polar := PI / 2 - altitude;
x := 1 * cos(polar);
y := 1 * sin(polar) * cos(azimut);
z := 1 * sin(polar) * sin(azimut);
v := new Vector3(x, y, z);
g := -v;
```

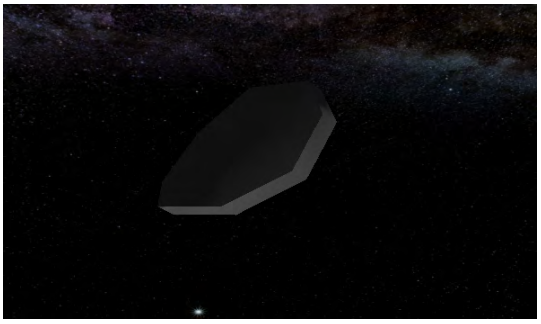
Die enorme Distanz zwischen Sonne und Erde sorgt dafür, dass bei der Erde ankommende Lichtstrahlen annähernd parallel verlaufen, weshalb als Lichtquelle direktionales Licht verwendet und entlang des Gegenvektors $\vec{g}$ ausgerichtet wird.

5.2 Visualisierungen

Innerhalb der MainScene werden alle Satelliten situationsabhängig auf verschiedene Art und Weise dargestellt. Jeder Satellit kann entweder durch ein 3D-Modell oder ein Partikel repräsentiert werden. Dabei hat der Objekttyp des Satelliten Auswirkungen auf das verwendete Modell und die Farbe des Partikels, wobei beim Objekttyp zwischen *Payloads*, *Rocket Bodys*, *Debris* und *unbekannten Objekten* unterschieden wird, was wiederum alles Bezeichnungen sind, die innerhalb der TLE JSON-files von space-track verwendet werden. Im Nachfolgenden wird detailliert auf die verwendeten Modelle, das Partikelsystem, die UI und weitere relevante Visualisierungen eingegangen.



(a) Satellit



(b) Weltraumschrott

Abbildung 5.1: 3D-Modelle von Patrick Saalfeld

5.2.1 Modelle

Sämtliche Objekte innerhalb des Projekt werden in Abhängigkeit ihres Objekttyps durch unterschiedliche Modelle dargestellt. Hierbei wird zwischen *Payloads*, *Rocket Bodys*, *Debris* und *unbekannten Objekten* differenziert. Die Modelle, welche die Payloads (Abb. 5.1a) und Debris (Abb. 5.1b) repräsentieren, sind von Patrick Saalfeld erstellt worden und die Modelle der ISS und des Hubble Space Teleskop entstammen der Website der NASA.[5] Als Erd-Modell dient Planet Earth Free [6] und die Skybox stammt von Adam Bieleckis Milky Way Skybox[7].

5.2.2 Partikel

Die Partikeldarstellung der Satelliten wird durch Unitys ParticleSystem und das selbst entwickelte Skript „ManageParticles“ verwaltet. Das ParticleSystem ist eine sehr effiziente Möglichkeit, eine große Anzahl von Objekten zu animieren. Jedes Partikel im ParticleSystem hat eine gewisse Lebensdauer, Geschwindigkeit, Farbe und noch viele weitere Eigenschaften, die u.a. für das Erzeugen von Rauch-, Feuer- oder Wolkeneffekten verwendet werden. Doch für die Repräsentation der Satelliten werden nicht alle jener Eigenschaften benötigt und sind mitunter sogar hinderlich, wie z.B. die begrenzte Lebensdauer, welche daher auf den größtmöglichen Wert eingestellt wird, da eine unbegrenzte Lebensdauer im ParticleSystem nicht vorgesehen ist. Auch die Position sämtlicher Partikel muss individuell, statt vom Zufall und von vorgegeben Bewegungsgeschwindigkeiten, bestimmt werden, wofür das zuvor erwähnte Skript *ManageParticles* verantwortlich ist. Dieses aktualisiert die Position aller Satelliten inklusive ihrer Partikel im sogenannten Batchverfahren (auch Stapelverfahren genannt). Das bedeutet, dass bei jedem Frame nicht alle Satelliten zugleich abgearbeitet werden, sondern nur eine gewisse Batchanzahl. Dies ist notwendig, da *VRSpaceDebris* mit über 20.000 Satelliten zu viel Zeit für die Aktualisierung der Positionen aufbringen müsste, sodass die Bildwiederholrate unter den für VR Anwendungen empfohlenen 90 Bildern pro Sekunde liegen würde, was nach Ae Ryu und Yoo [2020] zu Desorientierung, Übelkeit und anderen negativen Auswirkungen auf den Benutzer

[5] https://nasa3d.arc.nasa.gov/models, Letzter Zugriff: 14.05.2021

[6] https://assetstore.unity.com/packages/3d/environments/sci-fi/planet-earth-free-23399, letzter Zugriff: 17.05.2021

[7] https://assetstore.unity.com/packages/2d/textures-materials/milky-way-skybox-94001, letzter Zugriff: 17.05.2021

führen kann. Die neue Position wird mithilfe des in Abschnitt 5.1.1 beschriebenen Positionarrays bestimmt. Dabei wird die *CustomUpdate* Funktion, welche Teil des Skripts ist, mit der aktuellen Uhrzeit aufgerufen. Abhängig von der im *GameManager* festgelegten *timeScale* wird somit die aktuelle Position aus der linearen Interpolation zwischen den zeitlich passenden Positionen im Array berechnet. Nicht zuletzt sorgt *ManageParticles* dafür, dass die Lebensdauer der Partikel nicht abläuft und sie die passende Farbe zugewiesen bekommen. Dabei erhalten Payloads eine rote Farbe, Rockey Bodys eine gelbe, Debris eine blaue und unbekannte Objekttypen eine weiße Farbe. Das Ergebnis der Partikeldarstellung kann in Abb. 5.2 angesehen werden. Dabei muss beachtet werden, dass die Partikelgröße innerhalb des Bildes etwas größer wirkt, als sie beim Tragen der VR-Brille wahrgenommen wird.

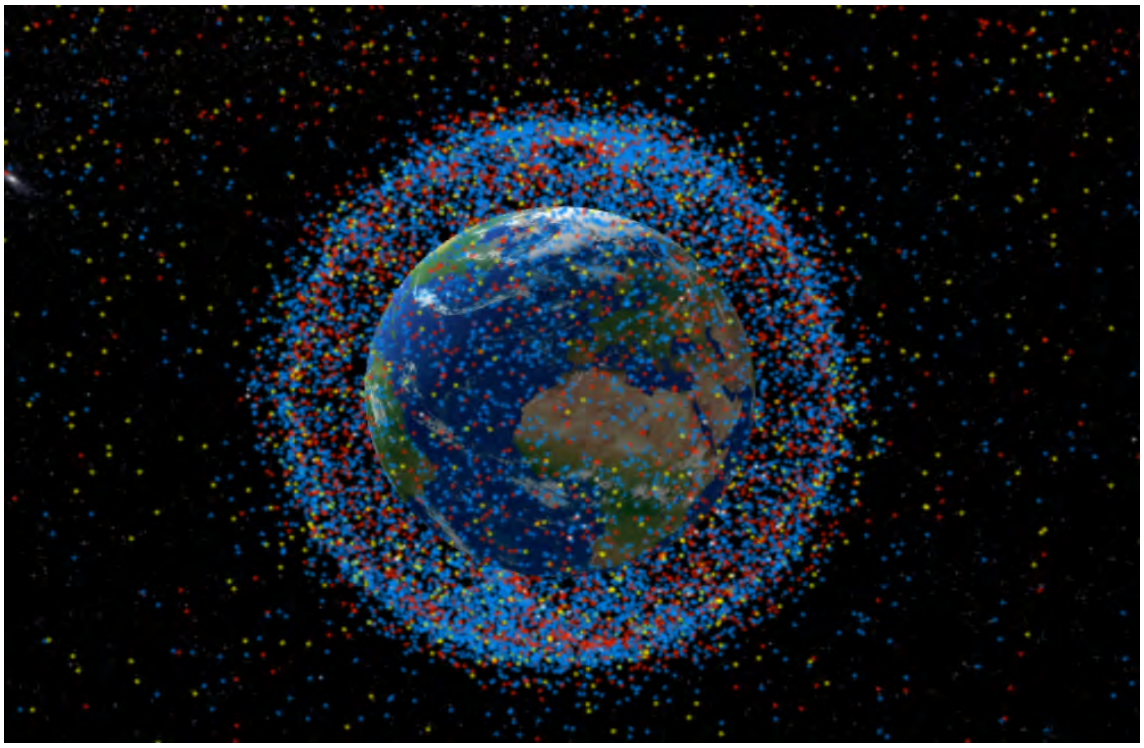
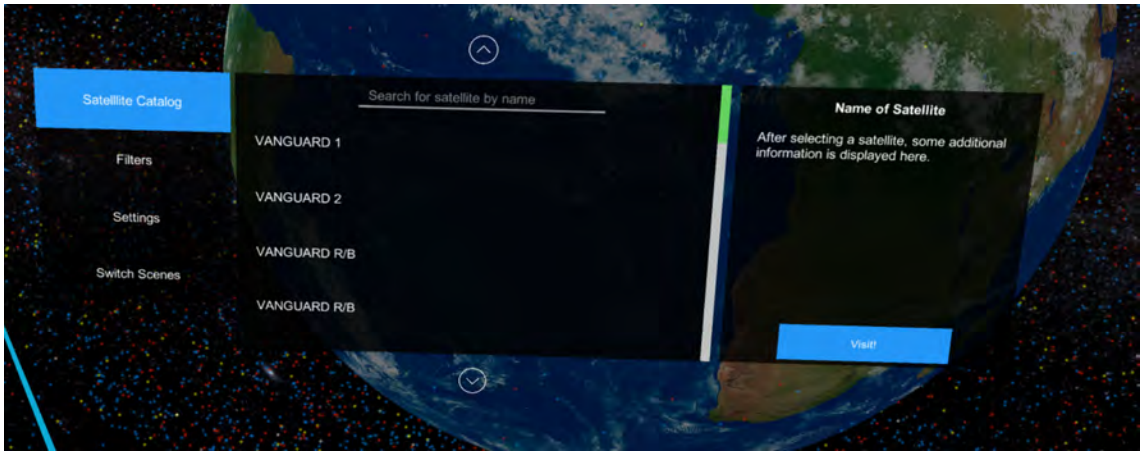


Abbildung 5.2: Ansicht der Erde mit Partikeldarstellung von Satelliten in der niedrigen Erdumlaufbahn innerhalb von *VRSpaceDerbis.*

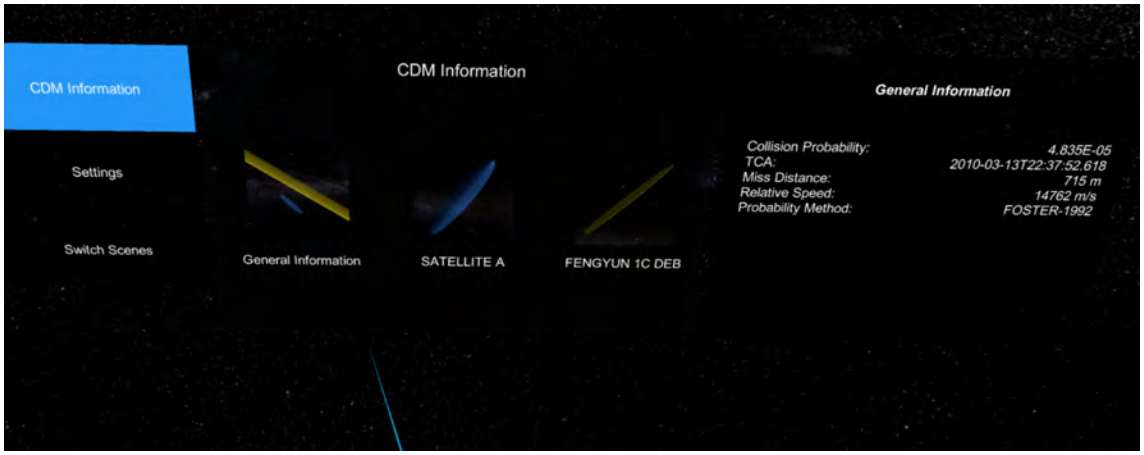
5.2.3 UI

Als User Interface (UI) dient ein mithilfe eines Laserpointers bedienbares Menü, welches sich in Abhängigkeit der Szene und des gewählten Modus dynamisch verändert. Dieser Abschnitt behandelt lediglich das Aussehen und den Aufbau des UI - mehr zu dem Interaktionsaspekt befindet sich in Abschnitt 5.3.2. Das UI wurde mit einer angepassten Version des Unity Assets VR UIKit[8] implementiert.
Das Menü ist dreigeteilt, wobei sich links die Menüpunkte befinden, mittig, abhängig vom ausgewählten Menüpunkt, weitere Auswahlmöglichkeiten bestehen und rechts, abhängig von der in der Mitte gewählten Auswahlmöglichkeit, weitere Informationen

[8]https://assetstore.unity.com/packages/tools/gui/vr-uikit-bootstrap-your-vr-app-with-ease-128236, letzter Zugriff: 24.05.2021



(a) MainScene



(b) CDMExampleScene

Abbildung 5.3: Unterschiede im Menü in Abhängigkeit von der Szene.

angezeigt werden können. Beim Öffnen des Menüs erscheint dieses in Blickrichtung des Anwenders an einer fixen Position und der vom Controller ausgehende Laserstrahl wird sichtbar. Per Default entstammt der Laserpointer dem rechten Controller, dies kann aber zu beliebiger Zeit geändert werden, indem auf dem anderen Controller die Triggertaste gedrückt wird. Die fixierte Position des Menüs ermöglicht dem Anwender, seine Distanz zum Menü nachträglich zu verändern, indem dieser näher herantritt oder sich weiter entfernt. Je nachdem, in welcher Szene sich der Anwender befindet,werden unterschiedliche Menüpunkte angezeigt (siehe Abb. 5.3). Innerhalb der MainScene werden die folgenden vier Menüpunkte zur Verfügung gestellt: Satellite Catalog, Filters, Settings und Switch Scenes. Der Satelliten Katalog ist standardmäßig ausgewählt, sollte aber ein anderer Menüpunkt beim Schließen des Menüs ausgewählt sein, so wird dieser gespeichert und beim erneuten Öffnen des Menüs wieder als ausgewählter Menüpunkt dargestellt. Bei der CDMExampleScene wird neben Settings und Switch Scene auch CDM Information als Menüpunkt gegeben. Ausgewählte Menüpunkte werden blau und deren Unterpunkte grün hervorgehoben.

5.2.4 Positionsabweichungsellipsoide

Im Abschnitt 5.1.2 wurde dargelegt, wie die Ausdehnung und Rotation der Fehler-Ellipsoide berechnet wird. In diesem Abschnitt werden nun die prozedurale Generierung der Ellipsoide sowie die verwendeten Materials/Shader erläutert. Für das prozedurale Generierieren eines Ellipsoids ist das Skript "ProceduralEllipsoid.cs", bzw. dessen Methode „generateEllipsoid“ verantwortlich. Dieses erhält die drei Längen des zu erzeugenden Ellipsoids x_length, y_length und z_length sowie den zu vergebenden Namen und gibt das Ellipsoid als neues GameObject zurück. Mit der Variable n wird die Anzahl der Unterteilungen des Ellipsoids bestimmt. Je größer n, desto mehr Vertices und Dreiecke enthält das Mesh. Zunächst werden die Vertices bestimmt. Der am weitesten rechts gelegene Vertex befindet sich an der Position (0, 0, z_width) und wird zu Beginn der Liste an Vertices hinzugefügt. Anschließend wird jeder Vertex einer Unterteilung ellipsenförmig bestimmt, wobei mit dem am weitesten in x-Richtung befindlichen Vertex mit $y = 0$ angefangen wird. Die Unterteilungen werden von rechts nach links abgearbeitet. Zum Schluss wird der am weitesten links befindliche Vertex (0, 0, -z_width) hinzugefügt. Der dazugehörige Code lautet wie folgt:

```
theta := PI / n;
phi := (2 * PI) / n;

verticies.Add(0, 0, z_width);
for (stack := 1; stack <= n - 1; stack++)
{
    stackRadiusX := sin(theta * stack) * x_width;
    stackRadiusY := sin(theta * stack) * y_width;
    z := cos(theta * stack) * z_width;
    for (slice = 0; slice <= n - 1; slice++)
    {
        x := cos(phi * slice) * stackRadiusX;
        y := sin(phi * slice) * stackRadiusY;
        verticies.Add(x, y, z);
    }
}
verticies.Add(0f, 0, -z_width);
```

Nach der Bestimmung der Vertices werden die dazugehörigen Dreiecke ermittelt. Dies erfolgt dreigeteilt. Zuerst werden alle Vertices, die zu der ersten Unterteilung gehören, mit ihren Nachbarn und dem am weitesten rechts befindlichen Vertex verbunden:

```
for (slice := 0; slice <= (n - 2); slice++)
{
    triangles.Add({ 0, slice + 2, slice + 1 });
}
triangles.Add({ 0, 1, n });
```

Anschließend werden nahezu alle weiteren Dreiecke gebildet. Hierbei werden erneut von rechts nach links alle Unterteilungen abgearbeitet. Genau genommen werden

die Dreiecke zwischen zwei Unterteilungen *stack* und *stack + 1* gebildet. Zur Veranschaulichung des Ablaufs folgendes Beispiel: Sei der erste Vertex des ersten Stacks *A1* und der erste Vertex des zweiten Stacks *B1*, so wird *A1* mit dem zweiten Vertex von *A* (*A2*), und dem zweiten Vertex von *B* (*B2*) zu einem Dreieck verbunden. Weiterhin bilden die Vertices *A1*, *B2* und *B1* ein Dreieck. Nun wird *A2* ebenso gehandhabt. Die daraus resultierenden Dreiecke sind {*A2*, *A3*, *B3*} und {*A2*, *B2*, *B3*}. Dies wird für alle Vertices des Stacks und schließlich für alle außer dem letzten Stack fortgeführt. Dieser muss später nur noch mit dem letzten Vertex zu Dreiecken verbunden werden. In Pseudocode sieht das Beschriebene wie folgt aus:

```
for (stack := 0; stack <= (n - 3); stack++)
{
    for (int slice := 0; slice <= (n - 2); slice++)
    {
        _vert1 := 1 + slice + (n * stack);
        _vert2 := _vert1 + 1;
        _vert3 := 1 + slice + (n * (stack + 1));
        _vert4 := _vert3 + 1;
        triangles.Add({ _vert1, _vert2, _vert4 });
        triangles.Add({ _vert1, _vert4, _vert3 });
    }
    vert1 := n * (stack + 1);
    vert2 := 1 + (n * stack);
    vert3 := n * (stack + 2);
    vert4 := 1 + (n * (stack + 1));
    triangles.Add({ vert1, vert2, vert4 });
    triangles.Add({ vert1, vert4, vert3 });
}
```

Zuletzt werden alle Vertices, die zu der letzten Unterteilung gehören, mit ihren Nachbarn und dem am weitesten links befindlichen Vertex verbunden:

```
lastvert := verticies.Count - 1;
for (slice := 0; slice <= (n - 2); slice++)
{
    vert2 := (n - 2) * n + slice + 1;
    vert3 := (n - 2) * n + slice + 2;
    triangles.Add({ lastvert, vert2, vert3 });
}
triangles.Add({ lastvert, (n - 1) * n, (n - 2) * n + 1 });
```

Nachdem sowohl die Vertices als auch die Dreiecke bestimmt wurden, werden diese dem Mesh übergeben und die Normalen mit Unitys Funktion „RecalculateNormals“ festgelegt. Auf diese Weise werden beide Fehlerellipsoide der Satelliten einer CDM erzeugt.
Der erste Ansatz bestand darin, diese mit einem Wireframe Shader zu versehen, um ein ähnliches Aussehen zu erzeugen, wie es bei FreeFlyer der Fall ist (siehe Abb. 5.4). Hierbei wird allerdings kein Gebrauch von der verbesserten räumlichen Wahrnehmung in VR gemacht und es stellte sich beim Testen heraus, dass sich die Größen der

Abbildung 5.4: Ellipsoide mit Wireframedarstellung innerhalb von FreeFlyer, Quelle: https://ai-solutions.com/wp-content/uploads/2020/09/cap3-image.png, letzter Zugriff: 25.05.2021

Ellipsoide mit einem Wireframe Shader nur schwer approximieren ließen. Daher wird anstelle dessen ein leicht transparenter Shader verwendet, welcher in VR deutlich besser wahrnehmbar ist. In Abb. 5.5 wird das fertige Aussehen der Ellipsoide gezeigt. Dabei muss beachtet werden, dass der transparente Shader in einer 2D Illustration eher ungeeignet ist und in VR anders auf den Betrachter wirkt.

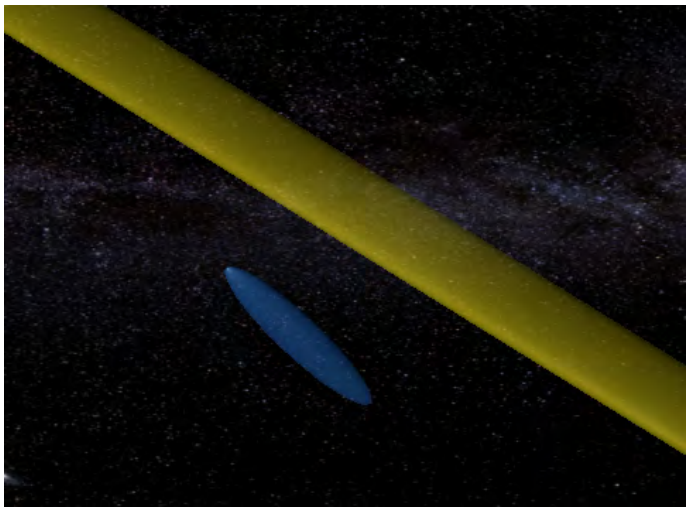


Abbildung 5.5: Prozedural generierte Ellipsoide mit transparentem Shader.

5.2.5 Positionsüberschneidungen durch Ray Marching

Der erste Ansatz verwendete die boolesche Operation „Intersection“ der Constructive Solid Geometry (CSG). Die Ergebnisse mit CSG waren jedoch nicht zufriedenstellend. Die CSG Intersection zweier Ellipsoide benötigte in Unity etwa 900 msec und war somit nicht in Echtzeit berechenbar. Außerdem gab es einige Fälle, bei denen die Berechnungen fehlschlugen oder nicht mit dem erwarteten Resultat übereinstimmten. Daher wurde stattdessen ein Ray Marching Shader implementiert, der nach dem sphere-tracing Prinzip arbeitet und die Überschneidung zweier Ellipsoide rendert. Mehr über den Ray Marching bzw. sphere-tracing Algorithmus kann in den Grundlagen unter Abschnitt 3.3 nachgelesen werden. Für den Ray Marching Algorithmus ist es erforderlich, die SDF der verwendeten geometrischen Objekte zu kennen. Für

Ellipsoide gibt es nach Quilez [2008] keine analytische Abstandsfunktion, die mit Hilfe von Wurzeln und/oder trigonometrischen Funktionen ausgewertet werden kann, weswegen die folgende Approximation Anwendung findet:

```
float GetDist(float3 p1, float3 r1) {
        float k1 = length(p1 / r1);
        float k2 = length(p1 / (r1*r1));
        float d1 = k1 * (k1 - 1.0) / k2;

        return d1;
}
```

Mit Hilfe folgender Funktionen können einige boolesche Operationen implementiert werden. Die Schnittmenge zweier Objekte erhält man, indem die Distanzfunktion die größere beider Distanzen zurück gibt.

$$distance = max(d1, d2);$$

Wenn also die maximale Distanz beider Distanzen zum ersten Mal unter dem Schwellenwert der Ray Marching Funktion liegt, so kollidiert der Strahl mit beiden Objekten, weshalb man sich an der Schnittstelle befindet. Wird d2 negiert, so ist das erste Objekt abzüglich des zweiten Objekts das Resultat.

$$distance = max(d1, -d2);$$

Gibt die Distanzfunktion das Minimum beider Distanzen zurück, so würde man die Vereinigung beider Objekte erhalten.

$$distance = min(d1, d2);$$

Um sowohl den im vorigen Abschnitt beschriebenen transparenten Shader als auch den „RayMarch.shader“, der nur die Schnittstellen hervorhebt und andernfalls durchsichtig ist, einem Ellipsoid hinzufügen zu können, wird ein Klon des GameObjects erzeugt. Dieser Klon erhält den transparenten Shader und wird zum Kindobjekt des Originals, welches wiederum den Ray Marching Shader erhält. Der Ray Marching Shader muss nur bei dem ersten der beiden Ellipsoide liegen, da die Schnittmenge beider auch immer eine Teilmenge des ersten Ellipsoids ist. Innerhalb des Shaders können nur Informationen zu der Position eines Ellipsoids abgerufen werden. Um die Rotation und Position beider Ellipsoide dem Shader zugänglich zu machen, erhalten beide ein Skript namens „EllipsoidShaderInterface.cs“, welches zu jedem Frame aktuelle Informationen über die Position, Rotation und Ausdehnung der Ellipsoide übermittelt. Um Rotationen und verschiedene Positionen der Ellipsoide zu berücksichtigen, wird nicht die Distanzfunktion verändert, sondern der Strahl, von dem aus die Distanz zum Objekt berechnet wird.
Die Farbe der Überschneidung wird anhand der approximierten Normalenvektoren festgelegt, wobei ihre Werte auf das Intervall [0, 1] abgebildet werden.

$$col.rgb = float3(0.5, 0.5, 0.5) + normals * 0.5;$$

5.3 Interaktionen

Dieses Kapitel beschreibt alle möglichen Interaktionen innerhalb von *VRSpaceDebris* und legt ihre Implementierung dar. *VRSpaceDebris* unterstützt die VIVE Controller und im weiteren Verlauf wird vielmals auf dessen Tasten verwiesen. Der Aufbau des Controllers kann in Abb. 5.6 angesehen werden.

Abbildung 5.6: Aufbau eines VIVE Controllers.

5.3.1 Fortbewegung in Kugelkoordinaten

Da Unity kartesische Koordinaten als Positionsdaten verwendet, müssen die Kugelkoordinaten in kartesische Koordinaten transformiert werden. Die Umrechnung lautet wie folgt, wobei „x-Rot“ den Azimutwinkel und „y-Rot“ den Polarwinkel darstellen:

```
newX := radius * sin(yrot * (PI / 180)) * cos(xrot * (PI / 180));
newY := radius * cos(yrot * (PI / 180));
newZ := radius * sin(yrot * (PI / 180)) * sin(xrot * (PI / 180));
```

Die aus der Berechnung resultierende Struktur kann in Abb. 5.7 angesehen werden. Sollte sich der Nutzer also über der Null Island (lon 0, lat 0) befinden, so wäre y-Rot 90 und x-Rot 0. Diese Position kann mithilfe des Trackpads verändert werden. Die Bewegungsgeschwindigkeit ist in kubischer Abhängigkeit von der Berührungsposition am Trackpad. Je weiter außen das Trackpad berührt wird, desto schneller bewegt

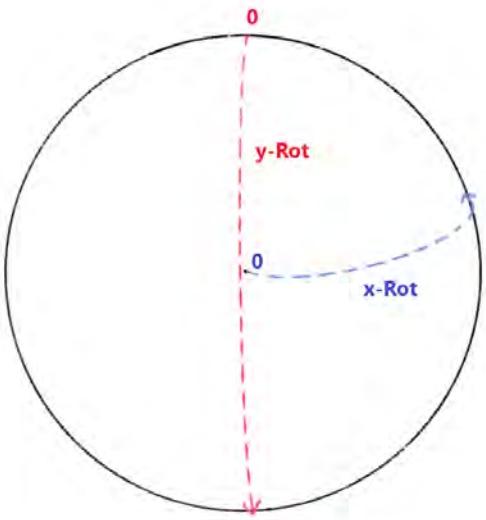


Abbildung 5.7: Verwendete Struktur für die Berechnung von Kugelkoordinaten.

sich der Nutzer in Richtung des Tastendrucks. Um zu gewährleisten, dass die Bewegungsgeschwindigkeit unabhängig von der Bildfrequenz ist, wird die berechnete Veränderung mit deltaTime, dem Intervall in Sekunden zwischen dem letzten und dem aktuellen Bild, multipliziert.

```
xrot += movementOffsetX * deltaTime * pow(feedback[0], 3);
yrot += movementOffsetY * deltaTime * pow(feedback[1] * -1, 3);
```

Bei „feedback“ handelt es sich um das Positionsarray, welches vom Trackpad zurückgegeben wird. Die Werte können in Abb. 5.8 eingesehen werden. Alle Werte

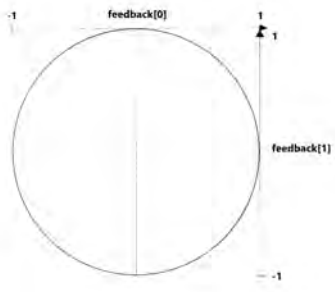


Abbildung 5.8: Werteverlauf je nach Berührungsposition auf dem VIVE Trackpad.

Abbildung 5.9: Knöpfe auf dem Trackpad mit Himmelsrichtungen als Namen.

befinden sich im Intervall [-1; 1], weswegen die Kubierung von *feedback* dazu führt, dass die Veränderung der Bewegungsgeschwindigkeit im Zentrum des Trackpads feiner und auf der Außenseite gröber ist. „movementOffsetX“ und „movementOffsetY“ geben die maximale Winkelveränderung pro Sekunde an. Diese lässt sich in

den Optionen innerhalb des Menüs (siehe Abschnitt 5.3.2.3) verändern. Neben der Berührung des Trackpads besteht weiterhin die Möglichkeit, diesen in die in Abb. 5.9 illustrierten Himmelsrichtungen zu drücken. Dabei wird - je nach gedrücktem Knopf - die maximale Bewegungsgeschwindigkeit verdoppelt und unabhängig von der Berührungsposition verarbeitet. Bei einem Knopfdruck der Taste „Ost“ wird somit das Produkt aus movementOffsetX und deltaTime mit 2 multipliziert und zu xrot addiert.

```
xrot += (2 * movementOffsetX * deltaTime);
```

Möchte der Anwender den Radius ändern, muss er zunächst den Menü-Knopf drücken, um den Bewegungsmodus zu wechseln. Dabei erscheint kurzzeitig eine Meldung, die den Anwender über den derzeitigen Modus informiert und anschließend wieder verblasst. Nun kann das Trackpad zur Vergrößerung oder Verkleinerung des Radius verwendet werden. Wird das Trackpad in der oberen Hälfte berührt, wird der Radius kleiner und der Anwender bewegt sich in Richtung Erde. Bei einer Berührung der unteren Hälfte wird der Radius größer und der Anwender bewegt sich wiederum von der Erde fort. Wie bei der Winkelveränderung steht die Bewegungsgeschwindigkeit entlang des Radius in kubischer Relation zur Berührungsposition am Trackpad.

```
radius -= movementOffsetRadius * deltaTime * pow(feedback[1], 3);
```

Die Variable „movementOffsetRadius“ bestimmt die maximale Bewegungsgesgeschwindigkeit entlang des Radius pro Sekunde und kann mit einem Knopfdruck von „Nord“ oder „Süd“ verdoppelt oder in den Optionen beliebig angepasst werden.
Möchte man sich nach der Anpassung des Radius erneut entlang x-Rot bzw. y-Rot bewegen, so genügt ein weiterer Knopfdruck der Menü Taste, um die Funktionalität des Trackpads wieder umzuschalten.

5.3.2 Menü

Die nachfolgenden Abschnitte zeigen sämtliche Interaktionen auf, die über die in Abschnitt 5.2.3 gezeigten Menüs erreichbar sind. Das Menü unterscheidet sich in Abhängigkeit von der Szene, in der sich der Nutzer befindet, da in den jeweiligen Szenen unterschiedliche Interaktionen benötigt bzw. nicht benötigt werden.

5.3.2.1 Suchen und Besuchen von Satelliten

Ein Feature innerhalb der MainScene besteht darin, jeden in der TLE-Datei enthaltenen Satelliten auswählen zu können, um entweder weitere Informationen zum Objekt zu lesen, oder diesen zu besuchen. Um keine Liste mit über 20.000 Satelliten durchsuchen zu müssen, gibt es die Möglichkeit, Satelliten nach ihrem Namen oder ihrer North American Aerospace Defense Command (NORAD)-ID zu suchen. Dafür muss der Laserpointer auf die Suchleiste gerichtet und der Triggerknopf gedrückt werden. Anschließend erscheint eine Tastatur (siehe Abb. 5.10), die zur Eingabe des Namens/ der ID verwendet werden kann. Die Suchanfrage ist nicht case sensitive und bezieht Teilwörter mit ein. Während der Eingabe eines jeden Zeichens erscheinen sofort alle passenden, jedoch maximal 30, Suchergebnisse in der Liste. In Abb. 5.11 werden die Ergebnisse der Suchanfrage ***ti*** gezeigt.

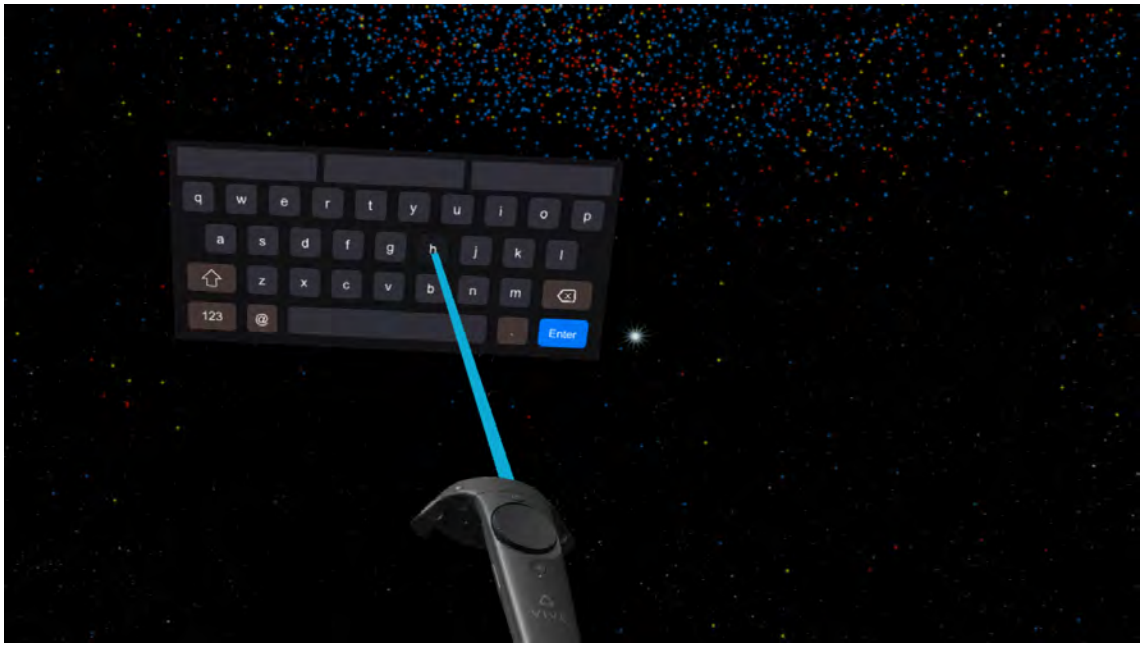


Abbildung 5.10: Tastur innerhalb von *VRSpaceDebris*, die mit dem Laserpointer bedient werden kann.

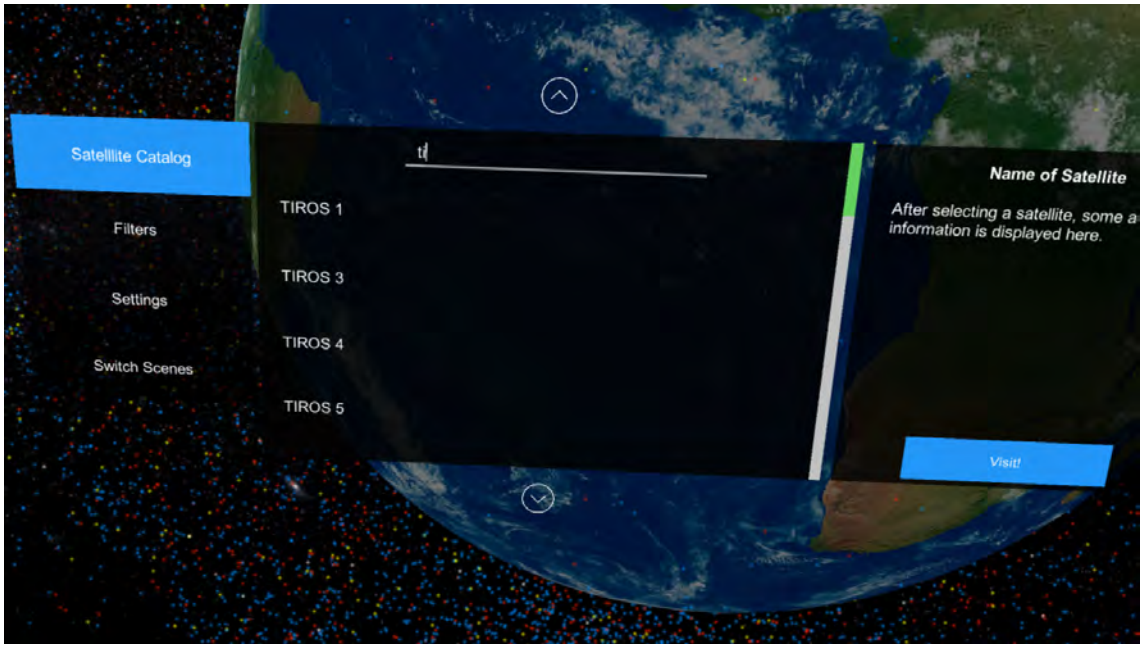


Abbildung 5.11: Suchergebnisse nach der Eingabe von ***ti***.

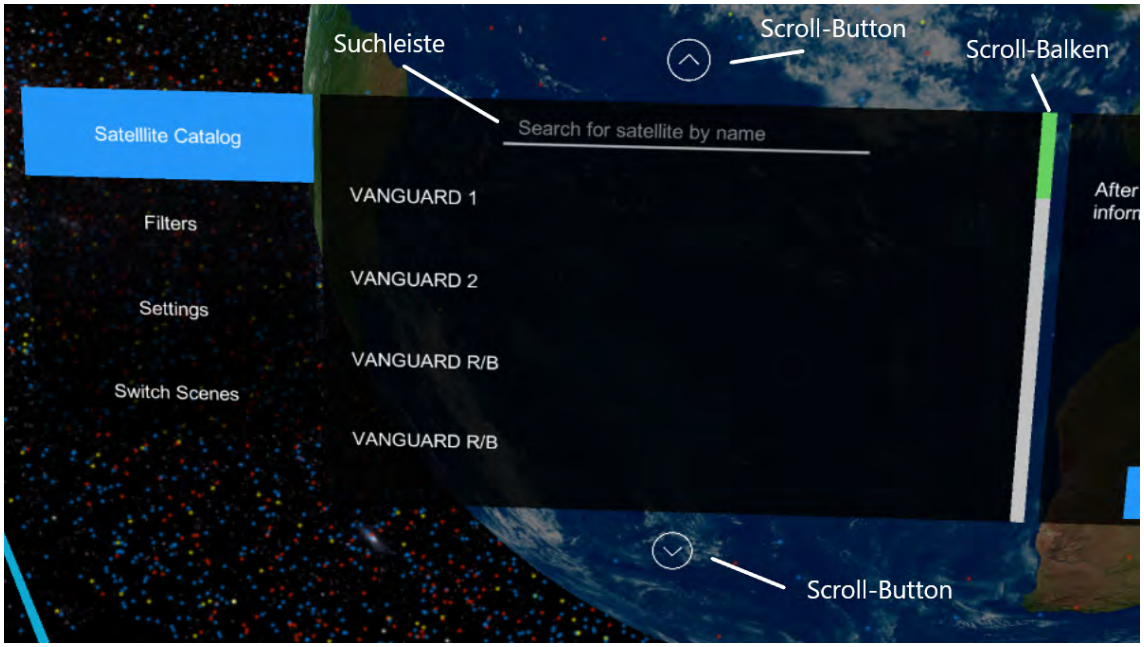


Abbildung 5.12: Verschiedene Möglichkeiten, die Suchergebnisse zu durchsuchen.

Die Liste mit Suchergebnissen wird automatisch nach der NORAD-ID in aufsteigender Reihenfolge sortiert. Nur vier Einträge können auf einmal angesehen werden, daher kann der Nutzer auf verschiedene Weisen durch die Liste scrollen. Zunächst können die Scroll-Button über/unter der Liste mit dem Triggerknopf angeklickt werden, um drei Elemente nach oben/unten zu scrollen. Weiterhin kann der Scroll-Balken mit dem Gedrückthalten des Triggers festgehalten und an eine beliebige Position gezogen werden, um zu der äquivalenten Stelle innerhalb der Liste zu gelangen. Auf dieselbe Weise können die Suchergebnisse festgehalten und gescrollt werden, wobei sich alle anderen Elemente mitbewegen. Zuletzt könnte man die exakte NORAD-ID in die Suchleiste eingeben, um bei gleichnamigen Objekten nur das gesuchte Objekt zu finden. Diese Möglichkeiten werden in Abb. 5.12 hervorgehoben. Sie erfüllen jeweils verschiedene Aufgaben: Der Scroll-Balken ermöglicht ein schnelles, der Scroll-Button ein präzises, aber langsames und das „swipen“ ein angenehmes, aber noch langsameres Durchsuchen der Liste. Einzelne Suchergebnisse können mit einem Klick der Triggertaste ausgewählt werden. Anschließend erscheinen weitere Informationen und es besteht die Möglichkeit, den Satelliten zu besuchen, indem man auf den ***Visit!*** Button klickt. Wenn man den Satelliten besucht, so wird man in die Nähe des Objekts teleportiert und das 3D-Modell des Objekts wird sichtbar. Weil manche Modelle wie die der ISS oder des Hubble Space Teleskops deutlich größer als die üblichen Modelle sind, ist die Entfernung vom Objekt variabel und von dem für das Modell gewählten Faktor abhängig. Die Berechnung der neuen Position sieht wie folgt aus:

```
newPosition := sat.position + (sat.position.normalized * factor);
```

Anschließend wird der y-Wert der neuen Position auf den y-Wert der Satellitenposition zurückgesetzt, sodass sich der Nutzer und der Satellit auf derselben Höhe befinden.

```
newPosition.y := sat.position.y;
```

Da es sich hierbei um kartesische Koordinaten handelt und die PlayerPosition in Kugelkoordinaten verarbeitet wird, müssen diese Koordinaten transformiert werden und die neue x und y-Rot sowie der Radius an den InputHandler gesendet werden. Andernfalls würde sich die Position des Nutzers auf die Position vor der Teleporation zurücksetzen, sobald sich dieser das erste Mal nach der Telportation mit dem Trackpad bewegt.

$$xrot = \arctan\left(\frac{z}{x}\right) * \frac{180}{\pi}$$
$$yrot = \arccos\left(\frac{y}{\sqrt{x^2+y^2+z^2}}\right) * \frac{180}{\pi}$$
$$radius = \sqrt{x^2+y^2+z^2}$$

Mögliche Divisionen mit 0 werden bei der Berechnung von xrot und yrot abgefangen. Neben der Teleportation und dem Anzeigen des 3D-Modells wird beim ***Visit!***-Knopfdruck das Menü geschlossen und der Laserpointer deaktiviert. Des Weiteren wird die Rotation des Nutzers verändert, sodass dieser in seiner Ausgangsrotation genau in Richtung des Satelliten blickt.

```
rotation = Quaternion.LookRotation((-newPosition.x, 0f, -
    newPosition.z), (0f, 1f, 0f));
```

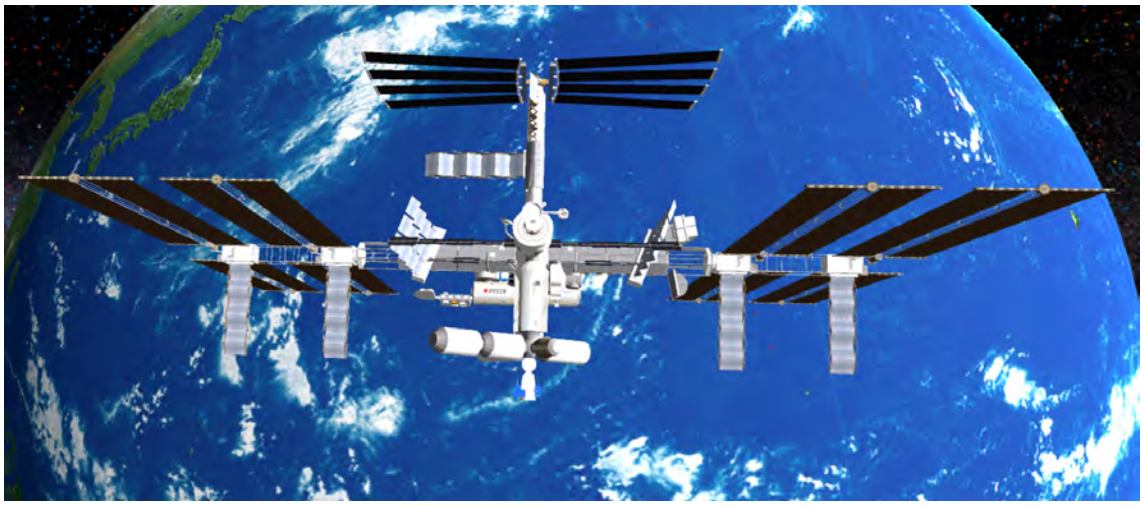
Das Resultat der beschriebenen Besucherfunktion, welches vom Skript „VisitButton.cs“ gemanagt wird, kann in Abb. 5.13 gesehen werden.

5.3.2.2 Filtern von Objekten

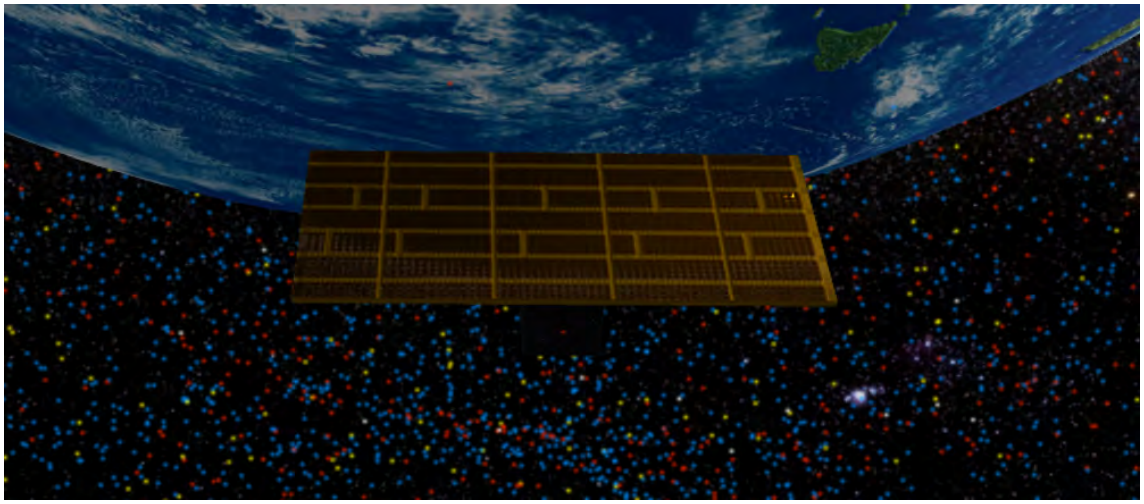
Das Filtern von Objekten ist ein weiteres Feature innerhalb der MainScene. *VR-SpaceDebris* stellt die folgenden drei Filter zur Verfügung:

1. Starlink
2. Galileo
3. Iridium 33 Collision Debris

Die zu den Filtern gehörenden Objekte werden auf Bedarf, d.h. erst nach Aufruf des Filters anhand ihrer Namen ermittelt. Auf diese Weise ließen sich viele weitere Filter einbauen, die nicht nur nach dem Namen, sondern auch nach anderen geparsten Eigenschaften suchen, wie z.B. alle Satelliten, die vor einem bestimmten Datum gestartet wurden. Um einen Filter anzuwählen, muss lediglich der Laserpointer auf die zum Filter gehörige Checkbox (siehe Abb. 5.14) gerichtet und der Trigger gedrückt werden. Nach der Aktivierung eines Filters verschwinden alle nicht dazugehörigen Partikel und per Default werden neben den Partikeln alle Orbits der gesuchten Objekte angezeigt. Diese können jedoch mit der in Abb. 5.14 zu sehenden Checkbox ***Don't show orbits*** ausgeblendet werden. Weiterhin besteht die Möglichkeit, mehrere Filter miteinander zu kombinieren. So können beispielsweise alle Starlink und Galilieo Satelliten gleichzeitig angezeigt werden. Werden alle Filter wieder deaktiviert, so erscheinen die Partikel aller Satelliten und der ursprüngliche Zustand wird wieder hergestellt. Für das Filtern und Ablesen aktivierter Checkboxen ist das Skript „HandleFilters.cs“ verantwortlich. In Abb. 5.15 und Abb. 5.16 befinden sich zwei Beispiele von aktiven Filtern.



(a) ISS



(b) Beliebiger Payload

Abbildung 5.13: Ausgangslage nach dem Besuchen von Satelliten mit verschiedenen 3D-Modellen.

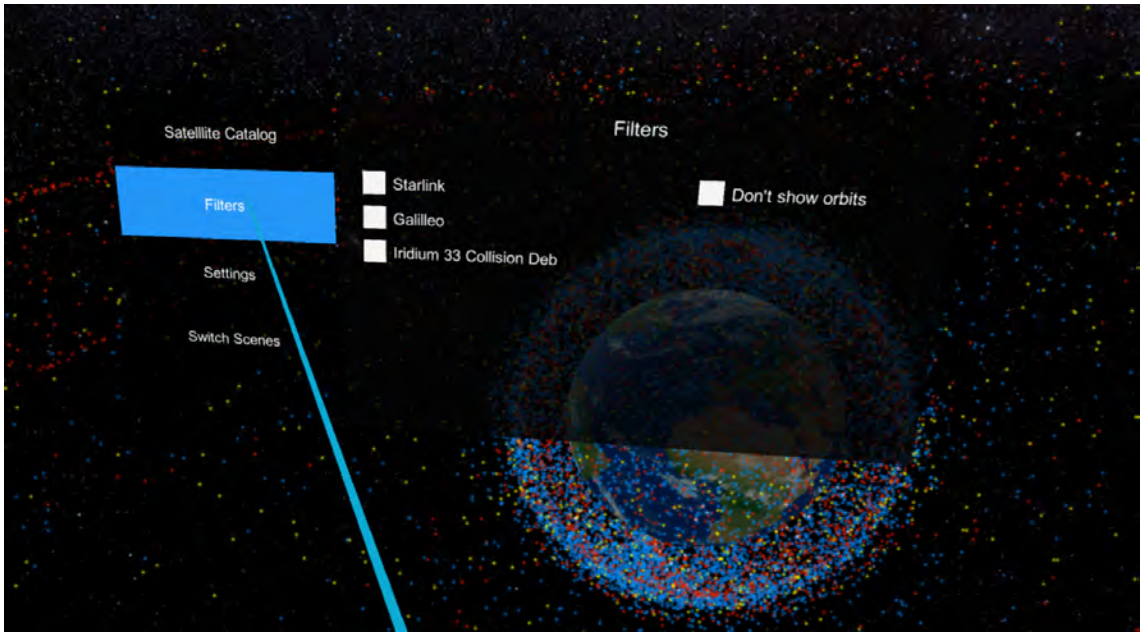


Abbildung 5.14: Verfügbare Filter in *VRSpaceDebris*

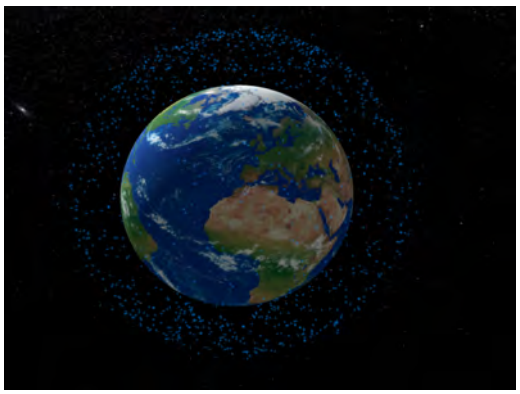


Abbildung 5.15: Iridium 33 und Kosmos 2251 Kollisionsbruchstücke mit ausgeschalteten Obrits.

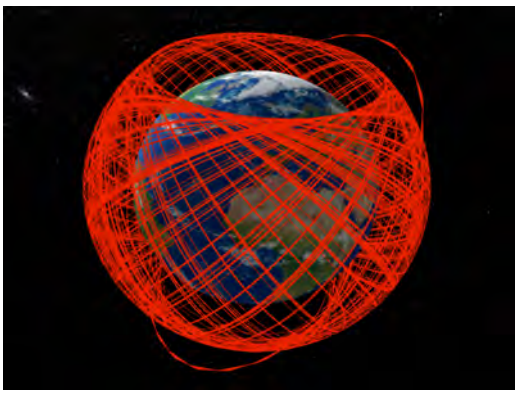


Abbildung 5.16: Starlink Satelliten und ihre Orbits.

5.3.2.3 Optionen

Die möglichen Optionen unterscheiden sich in Abhängigkeit von der Szene, in der man sich befindet. In Abb. 5.17 kann der Aufbau des Optionsfensters in der MainScene angesehen werden. ***Axis Speed*** beeinflusst die Bewegungsgeschwindigkeit entlang der in Abschnitt 5.3.1 beschrieben x und yrot. Die abgebildete Zahl gibt die maximale Winkelveränderung pro Sekunde an. Bei einer Axis Speed von 6 würde eine Drehung um die Erde, unabhängig vom Radius, insgesamt 60 Sekunden andauern, sofern man den Trackpad am äußeren Rand berührt. ***Radius Speed*** bestimmt die Fortbewegungsgeschwindigkeit entlang des Radius, wobei der angezeigte Wert aufgrund der gewählten Skalierung die maximale Geschwindigkeit in 10km/s darstellt. Die ***Time Scale*** bestimmt die Geschwindigkeit, mit der die Zeit abläuft, und somit, wie schnell sich die Satelliten fortbewegen. Während sich bei einer *Time Scale* von 1 alle Satelliten in Echtzeit fortbewegen, bewegen sie sich bei einer *Time Scale*

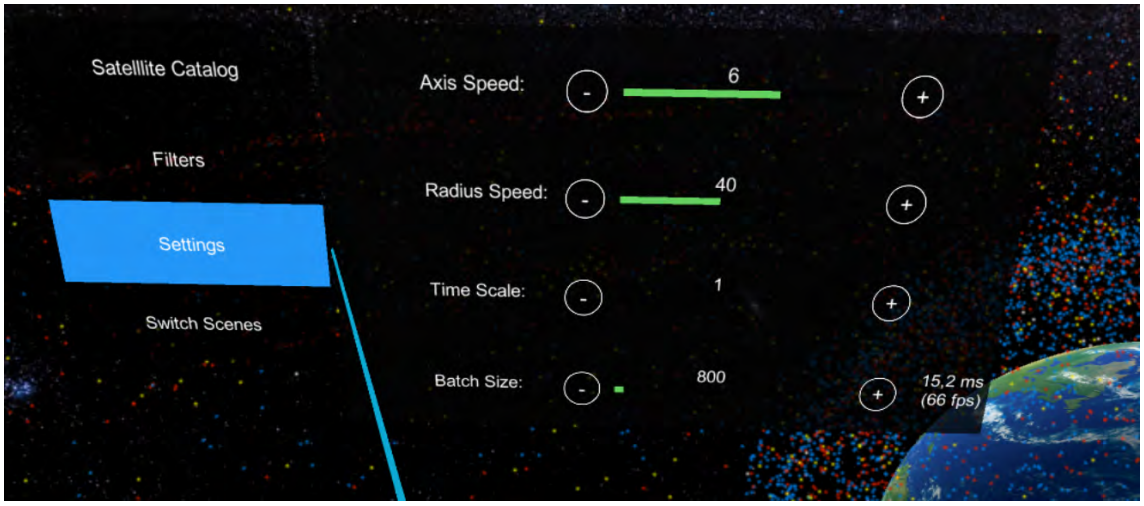


Abbildung 5.17: Mögliche Optionen in der MainScene.

von n n-mal so schnell. Zuletzt kann die in Abschnitt 5.2.2 beschriebene ***Batch Size*** verändert werden. Hiermit kann, je nach Leistungsfähigkeit des Rechners, die Anzahl an Satelliten, die in jedem Frame abgearbeitet werden, angepasst werden. Um die Suche nach der optimalen *Batch Size* zu erleichtern, werden unten rechts die Frametime und Bildfrequenz angezeigt. Alle vier zuvor beschriebenen Werte können entweder mit einem Klick auf den danebenliegenden Plus- und Minusbutton oder an den Leisten selbst verändert werden.
Das Optionsmenü der CDMExampleScene (siehe Abb. 5.18) enthält ebenfalls die Möglichkeit, die Fortbewegungsgeschwindigkeiten des Nutzers zu verändern. Diese sind allerdings deutlich geringer eingestellt, da man sich in der Regel bei dem Betrachten von Positionsabweichungsellipsoiden präziser fortbewegen möchte. Des Weiteren kann das Konfidenzintervall der Positionsabweichungsellipsoide auf eines der vorgegebenen Intervalle (80, 85, 90, 95, 97.5, 99, oder 99.5%) angepasst werden. Nach dessen Anpassung werden die alten Ellipsoide in Echtzeit mit den neuen Ellipsoiden ausgetauscht. Weiterhin lassen sich über die Optionen die RTN-Achsen der Satelliten einblenden. Ein Beispiel hierfür befindet sich in Abb. 5.19, wobei die roten Linien die R-, die grünen Linien die T- und die blauen Linien die N-Achsen repräsentieren. Die letzte Option ermöglicht das Einstellen/Umstellen des verwendeten Modus zur Analyse der möglichen Kollision. Weitere Informationen zu den auswählbaren Modi befinden sich in Abschnitt 5.3.2.4

5.3.2.4 Analyse Modi

Die drei Modi ***Explore TCA***, ***Timelapse Loop*** und ***Time Manipulation*** werden von dem Skript *HandleSettings2.cs* verwaltet. Der aktive Modus kann innerhalb der Optionen in der CDMExampleScene ausgewählt werden. Wird der Modus verändert, so wird die Funktion „HandleModeChange“ aufgerufen, welche dafür sorgt, dass UI-Elemente des vorherigen Modus mit den Elementen des neuen Modus ausgetauscht werden. Da die Funktionen innerhalb der Modi bereits im Konzept erläutert wurden, wird im Folgenden nur die Implementierung von besonderen Funktionen beschrieben.

Abbildung 5.18: Mögliche Optionen in der CDMExampleScene.

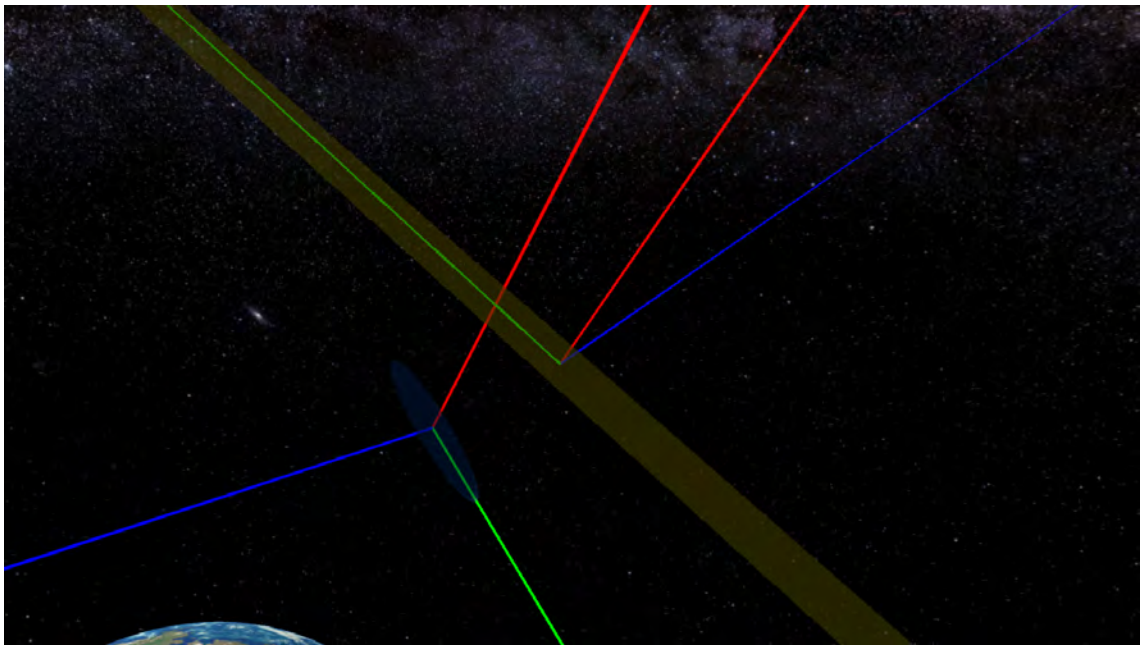


Abbildung 5.19: RTN-Achsen beider Satelliten in der CDMExampleScene.

Zeitschleife

Die Zeitschleife im *Timelapse Loop*-Modus wird mittels Unitys *InvokeRepeating* Funktion ermöglicht. Dabei wird die wird die Funktion *moveObjects* aus dem *LoadAndManagageCDM.cs* Skript alle 20 msec aufgerufen.

```
private void moveObjects()
{
    int index := number_of_repeats % amount;

    ellipsoid1.transform.position := simplifiedPos1[index];
    ellipsoid2.transform.position := simplifiedPos2[index];

    number_of_repeats++;
}
```

Die *simplifiedPos*-Arrays enthalten die Positionsdaten der Ellipsoide. Diese werden durch die *SimplifyPositions* Funktion berechnet, welche immer dann ausgeführt wird, wenn der Nutzer die Startzeit und/oder die Dauer der Zeitschleife verändert.

Ellipsoide per Laserpointer verschieben

Die Verschiebung der Positionsabweichungsellipsoide in dem *Time Manipulation* Modus erfolgt, sobald der Nutzer eines der beiden Ellipsoide mit dem Laserstrahl greift, indem dieser den Trigger-Knopf drückt während er den Laserstrahl auf das Ellipsoid hält. Der festgehaltene Ellipsoid kann nun entlang seines Satelliten-Geschwindigkeitsvektors verschoben werden. Um das Verschieben zu erleichtern, wird der nächstliegende Punkt zwischen dem Laserstrahl und dem Geschwindigkeitsvektor auf letzterem ermittelt, sodass der Nutzer nicht exakt den Geschwindigkeitsvektor treffen muss. Dieser Punkt wird mithilfe der *ClosestPointsOnTwoLines* Funktion aus dem Unity Wiki errechnet. [9] Mit jeder neuen Position wird der dazugehörige Zeitpunkt in Relation zum TCA wie folgt berechnet:

$$timeOffset := (closestPoint.x - position1.x)/velocity1.x;$$

Hiermit werden die Positionen beider Ellipsoide wie folgt berechnet:

```
position1 := position1 + velocity1  * timeOffset;
position2 := position2 + velocity2  * timeOffset;
```

Schließlich wird der neue timeOffset in die UI geschrieben, sodass der Nutzer den eingestellten Zeitpunkt relativ zum TCA ablesen kann. Jener Zeitpunkt kann auch per Schieberegler verändert werden. Die Berechnung der neuen Position erfolgt analog.

[9] http://wiki.unity3d.com/index.php/3d_Math_functions, letzter Zugriff: 21.08.2021

6. Evaluierung

Zur Evaluierung von VRSpaceDebris wurde am 06.07.2021 ein Experteninterview mit Herrn Dr. Hauke Fiedler geführt. Herr Dr. Fiedler hat zunächst Allgemeine Physik und anschließend Astrophysik studiert, worin er promovierte. Später erlangte Herr Dr. Fiedler einen Master in Space Systems Engineering. Er arbeitet seit 20 Jahren in unterschiedlichen Bereichen des DLR. Vor etwa 10 Jahren hat er die Gruppe SSA des DLR aufgebaut und leitet diese seitdem. Zu seinen aktuellen Aufgaben gehört der Aufbau eines Teleskopnetzes (SMARTnet) und die Entwicklung der Prozessierungsmaschine BACARDI. Nach Eigenaussage versucht Herr Dr. Fiedler, möglichst gute Daten zu generieren, sodass gezielter ausgewichen und unnötige Ausweichmanöver vermieden werden können, um letztlich die Gefahr des Weltraumschrotts zu minimieren. Aufgrund seiner Erfahrungen und Verantwortung in der SSA ist Herr Dr. Fiedler ein sehr geeigneter Experte, um den Nutzen einer neuen Analysesoftware zu überprüfen. Im Umgang mit VR hat Herr Dr. Fiedler nach eigenen Angaben jedoch nur wenig Erfahrung gesammelt.

Das Treffen fand online statt, da die Corona-Pandemie kein Präsenztreffen zuließ. Leider war es nicht möglich, Herrn Dr. Fiedler eine VR-Brille zukommen zu lassen, weswegen er die Software nicht selbst ausprobieren konnte. Stattdessen fand zu Beginn eine dreiviertelstündige Live-Demo per Bildschirmübertragung statt. Anschließend wurden einige Fragen gestellt. Dazu gehörten sowohl technische Fragen als auch Fragen, die seine Meinung zu bestimmten Features und dem Nutzen von VRSpaceDebris betrafen. Diese werden in den folgenden Abschnitten zusammengefasst. Ein Transkript des Interviews befindet sich im Anhang.

6.1 Technische Fragen

Zunächst wurde über die vom DLR genutzte Software zur Kollisionsanalyse gesprochen. Herr Dr. Fiedler beschrieb, dass die SSA-Gruppe des DLR in Deutschland für die Bahnproporgation als Software entweder GHOST oder ODEM verwendet, welche beide intern entwickelt wurden. Für die Kollisionsberechnung sei die Flugdynamik-Gruppe verantwortlich, die ebenfalls eigens entwickelte Software verwende, da das

Programm FreeFlyer als zu unpräzise angesehen werde.f Auf die Frage, ob man in dem Beispielszenario (Kollisionsanalyse zweier Satelliten), welches in der Arbeit verwendet wurde, ausweichen müsste, antwortete Herr Dr. Fiedler, dass in der Tat so verfahren werden würde. Allerdings sei die Entscheidungsfindung bei jedem Raumfahrtzentrum und Satellitenbetreiber anders. Manche würden nur nach der Wahrscheinlichkeit gehen, andere würden eine zweite Wahrscheinlichkeit hinzunehmen und die SSA-Gruppe des DLR verwende neben der Wahrscheinlichkeit noch eine radiale Komponente und lasse Erfahrungswerte aus der Vergangenheit in die Entscheidungsfindung einfließen. Ob tatsächlich ausgewichen werde, könne jeder Betreiber für sich selbst entscheiden. Da die Verantwortung im Falle einer Kollision bei den Betreibern liegt, würden diese auch eigenständig über mögliche Ausweichmanöver entscheiden und die Wahrscheinlichkeiten selbst errechnen wollen.

6.2 Fragen zu VRSpaceDebris

Im Anschluss an die technischen Fragen wurden Fragen zu den Features und dem Nutzen von *VRSpaceDebris* gestellt. Die nachfolgenden Fragen sind thematisch sortiert.

Allgemeine Funktionen

Herr Dr. Fiedler bewertete die Darstellung der Fortbewegung des Nutzers in Kugelkoordinaten als sehr gut geeignet, da diese auf natürliche Art und Weise die Ephemeriden der Objekte sehr gut widerspiegelten. Mit anderen Fortbewegungssystemen bräuchte man immer zwei Bewegungen zugleich, um Satelliten zu verfolgen. Das Bewegen in Blick- oder Controllerrichtung sei also nicht sinnvoller. Es würde ihn nicht stören, wenn er sich bei der Fortbewegung um die Erde immer selbst drehen müsse, um die Erde weiterhin im Mittelpunkt zu sehen. Ganz im Gegenteil. Dadurch, dass man sich selber bewegt, könne der eigene Eindruck sogar noch viel intensiver sein. Auf die Frage, ob die Teleportation zu einem Satelliten dazu führen könnte, dass der Nutzer die Orientierung verliert, antwortete Herr Dr. Fiedler, dass man natürlich die Orientierung verliere, weil man nicht wisse, wo man sich gerade befindet und wo das neue Objekt ist. Gerade bei Objekten im erdnahen Orbit könnte dies ein Problem sein. Eine Alternative zur Teleportation könnte das „Hinfahren“ zum Objekt auf einer Dreiecksbahn sein.

Analyse Funktionen

Die drei verschieden Analysemodi, die vorhanden sind, findet Herr Dr. Fiedler gut. Als Ergänzung für die *CDMExampleScene* wären die Komponenten in RTN sinnvoll, wobei die radiale Komponente am entscheidensten sei. Die tangentiale Komponente sei eher uninteressant, da diese am meisten schwankt, weshalb man über diese ohnehin keine Aussage treffen könne. Bei dem Vergleich der Eignung zwischen einem transparenten Shader und einen Wireframe-Shader für die Darstellung von Positionsabweichungswahrscheinlicheiten von Satelliten entschied sich Herr Dr. Fiedler für den in *VRSpaceDebris* implementierten transparenten Shader. Dieser sei optisch wesentlich eindrücklicher und sofort intuitiv greifbar. Beim Betrachten des Wireframe-Shaders könne dessen Rückseite für Verwirrung sorgen. Hier müsste man

das Wireframe zweifarbig gestalten, damit man einen Mehrwert vom Wireframe-Shader hat. Die Überschneidungsvisualisierung der Ellipsoide hält Herr Dr. Fiedler für sinnvoll und würde sie auf jeden Fall so beibehalten. Ebenso die Möglichkeit, die Ellipsoide per Hand fortzubewegen. Ergänzend dazu schlug er vor, die relativen Abstände beider Ellipsoide zueinander anzuzeigen, sodass man erkennen könnte, wenn man das Ellipsoid vor und zurück bewegt. Auf die Vor- und Nachteile der VR-Anwendung im Vergleich zu einer Desktop-Anwendung angesprochen, erläuterte Herr Dr. Fiedler, dass er Erstere gerade im Bereich der medialen Aufmerksamkeit für sehr relevant halte, weil man eine solche Software bspw. einem Politiker zeigen könne. Das halte er für sehr wesentlich. *VRSpaceDebris* sei eine Software, welche man auch auf Konferenzen wie dem International Astronautical Congress (IAC) präsentieren könne, was er auch tun würde. Wenn es aber darum ginge, Manöver zu berechnen, sei die Analyse in VR zu langsam. Am Desktop würde man die Zahlen sehen, wissen welche Zahlen sinnvoll sind und könne die Zahlen sofort einschätzen. Das ginge viel schneller als das mit einer VR-Brille und einem Laserstrahl zu machen sei. Da sei die Desktop-Anwendung mit den ganzen Algorithmen, die man dafür entwickelt hat, sinnvoller, zumal einige Berechnungen nicht automatisiert werden könnten. Das Schnelligkeitsproblem könnte laut Herrn Dr. Fiedler mit einer funktionierenden Sprachsteuerung gelöst werden. Ein Weiteres Problem sei allerdings, dass nicht alle Manöverberechnungen automatisiert werden könnten. Abschließend ist Herr Dr. Fiedler der Meinung, dass er nach der Vorstellung auch ohne weitere Einweisung mit VRSpaceDebris zurecht käme, da es nur 7 Knöpfe zu bedienen gäbe. Von daher glaubt er, dass er keine große Schwierigkeiten haben würde und er sehr viel Spaß daran hätte.

6.3 Zusammenfassung

Nahezu alle Funktionen von *VRSpaceDebris* wurden von Herrn Dr. Fiedler positiv bewertet. Lediglich bei der Teleportation zu ausgewählten Satelliten gäbe es laut Herrn Dr. Fiedler eine bessere Alternative – eine flüssige Fortbewegung entlang einer Dreiecksbahn. Dies könnte das Orientierungsproblem nach der Teleportation lösen. Keine der entwickelten Funktionen wurde als überflüssig bezeichnet. Ein paar weitere Ideen und Verbesserungsvorschläge wurden genannt, wie z.B. das Anzeigen der relativen Distanz zwischen den beiden Positionsabweichungsellipsoiden oder der Darstellung der radialen Komponente.

Die Forschungsfrage, ob eine VR-Anwendung einen Beitrag zur Kollisionsanalyse von Satelliten leisten kann, wurde mit nein beantwortet, da die Arbeit am Desktop schneller sei. Dies könnte zwar mit einer Sprachsteuerung behoben werden, jedoch bliebe weiterhin das Problem, dass nicht alle Kollisionsberechnungen automatisiert werden können und in VR nur schwer durchführbar wären. Allerdings nannte Herr Dr. Fiedler einen anderen Vorteil: die mediale Aufmerksamkeit. *VRSpaceDebris* sei hervorragend für die Präsentation auf Messen und Konferenzen geeignet und könne damit die benötigte Aufmerksamkeit auf die Weltraumschrottproblematik lenken. Somit wäre es möglich, dass *VRSpaceDebris* dennoch einen Beitrag zur Lösung des Weltraumschrottproblems leistet – wenn auch auf eine andere Weise als ursprünglich gedacht.

7. Fazit und zukünftige Arbeiten

Das Ziel dieser Bachelorarbeit war es zu ermitteln, ob VR für die Kollisionsanalyse von Satelliten geeignet ist. Dazu wurde zunächst gemeinsam mit einigen Mitarbeiterinnen und Mitarbeitern des DLR ein Konzept erörtert. Anschließend wurde eine VR-Applikation in Unity entwickelt, welche das zuvor erörterte Konzept erfüllt. Innerhalb dieser Applikation kann man sowohl alle im Datenset vorhandenen Objekte im simulierten Orbit der Erde betrachten und filtern als auch CDMs mithilfe verschiedener Modi analysieren. Zuletzt wurde die Applikation bei einem Experteninterview gemeinsam mit Herrn Dr. Fiedler, dem Leiter der SSA-Gruppe des DLR, evaluiert. Dabei wurden einzelne Features sowie der Gesamtnutzen der entwickelten Applikation bewertet.

Das Ergebnis der Evaluierung ist, dass die gewünschten Funktionen zwar sehr gut umgesetzt wurden, VR jedoch für die Analyse von Satellitenkollisionen nicht geeignet ist. Dadurch, dass die Analysegeschwindigkeit gering ist und einige Berechnungen nicht automatisiert werden können, sondern von Hand durchgeführt werden müssen, ist die Arbeit am Desktop nach wie vor praktikabler. Jedoch stellte sich heraus, dass die entwickelte VR-Anwenndung sehr gut für mediale Aufmerksamkeit der Weltraumschrottproblematik geeignet sein könnte, da hiermit z.B. Politiker Kollisionsgefahren erkennen und in einer immersiven Umgebung die Vielzahl an Satelliten im Orbit der Erde beobachten können.

Aufgrund der Corona-Pandemie fand die Evaluierung online statt und Herr Dr. Fiedler konnte die Software nicht selbst ausprobieren. Bei der Bildschirmübertragung kamen die Vorteile von VR nicht so sehr zur Geltung, weshalb eine weitere Evaluierung durchgeführt werden könnte. Allerdings ist es fraglich, ob sich an der Beantwortung der Forschungfrage etwas ändern würde, da die Problematik der Analysegeschwindigkeit und die Tatsache, dass nicht alle Berechnungen automatisiert werden können, bestehen bleiben.

Abschließend werden im nachfolgenden Abschnitt Aspekte diskutiert, die auf dieses Projekt aufbauend im Rahmen zukünftiger Arbeiten durchgeführt werden könnten.

Zukünftige Arbeiten

Da das maßgeblichste Problem der VR-Applikation die Analysegeschwindigkeit ist, könnte in einer anderen Arbeit diese mithilfe einer Sprachsteuerung beschleunigt werden und einige der entwickelten Interaktionen ersetzen. Weiterhin wäre es für die Analyse sinnvoll, weitere Informationen der Satelliten zu parsen und anzuzeigen, wie z.B. Apogäum und Perigäum des Satelliten.

Um das Datenset zu erweitern, wäre das Hinzufügen weiterer Satellitenkataloge, wie z.B. den russischen „Vimpel-Katalog “[10], möglich. Diese müsste man aber mit den TLE-Daten abgleichen, um redundante Weltraumobjekte zu vermeiden.

Der Nutzen der medialen Aufmerksamkeit könnte weiter ausgebaut werden, indem mehr Fokus auf die Immersion der VR-Anwendung gesetzt wird. Zusäztzlich könnte man auf Messen und Konferenzen mithilfe der VR-Applikation auf das Weltraumschrott-Problem aufmerksam machen und Studien zum Weltraumlagebewusstsein durchführen.

[10] spacedata.vimpel.ru, letzter Zugriff 24.08.2021

Anhang

A.1 Transkription des Interviews mit Hauke Fiedler

Christopher Olson: Einmal am Anfang die Frage - Welche Software verwenden Sie momentan, um Kollisionen abschätzen zu können oder analysieren lassen zu können? Wahrscheinlich auch FreeFlyer?

Hauke Fiedler: Nein! Das ist nicht verlässlich genug, und das ist auch nicht präzise genug. Und da kennt man auch teilweise die Störung nicht. Das gleiche wie .. Da gibt's noch eine andere Software aus den USA, die verwenden wir auch nicht. Einfach deswegen auch, weil bei diesem anderen US Produkt, da werden die Module teilweise zusammengebaut von verschiedenen Abteilungen und die Schnittstellen - da weiß ein Programmierer nicht, was die anderen machen und da stimmen teilweise die Schnittstellen nicht überein und da wissen wir auch, dass teilweise falsche Werte verwendet werden. Wir haben eigene Software, die bei uns hier im Haus entwickelt worden ist. Sowohl für die Bahnproporgation - da haben wir auch eine hochpräzise Software, die auch teilweise lizenziert wird - die heißt dann auch entweder ODEM oder GHOST, je nachdem welche sie haben wollen. ODEM ist die Standardsoftware für ganz normale Bahnproporgation und für die ganzen Kollisionsberechnungen, das macht bei uns alles die Flugdynamik. Der Herr Kahle - ich mach das nicht - und die haben da auch eigene Sachen, die auch hier im Haus selber entwickelt worden sind, wo sie auch Kollisionen selber ausrechnen. Für´die Flugdynamik ist so eine Animation wenig hilfreich. Die rechnen hauptsächlich in der Biplane Manöver aus, das ist praktisch die Bahnebene von diesen zwei Satelliten, wo die sich auch treffen. In dieser Biplane rechnen sie dann auch die Manöver aus, wie die aussehen könnten. Eventuell finde ich rasch einen Vortrag, dann kann ich Ihnen das mal kurz zeigen. Haben Sie noch eine andere Frage in der Zwischenzeit?

Christopher Olson: Ja gerne! Gleich mehrere. Einmal wollte ich fragen, wie Sie denn die Fortbewegung in Kugelkoordinaten finden, oder ob Sie meinen, dass eine direkte Bewegen - entweder über Blickrichtung, oder Richtung des Controllers - für sinnvoller halten würden?

Hauke Fiedler: Nein, Kugelkoordinaten sind sogar sehr gut! Deswegen sind sie sehr gut, weil sie natürlich die Ephemeris (Positionswerte sich bewegender astronomischer Objekte) ist natürlich in Kepler oder oskullierenden Elementen oder sowas und dann sind Kugelkoordinaten da sehr gut. Das passt. Das hat mich von Anfang an überzeugt. Das fand ich sehr gut. Wenn Sie da auf x,y,z gehen, ich glaube da haben Sie sehr große Schwierigkeiten, da brauchen Sie immer zwei Bewegungen gleichzeitig, um die Satelliten zu verfolgen. Und das macht die Sache natürlich sehr zäh dann.

Christopher Olson: Wie würden Sie denn Ihren Beruf bezeichnen? Sie sind ja beim SSA verantwortlich, oder?

Hauke Fiedler: Ganz genau! Ich mach Situational Space Awareness. Das ist Weltraumlage, Weltraum Lagebewusstsein. Und womit ich mich hauptsächlich beschäftige, das ist einmal Space Debris im Allgemeinen. Wo ist er? Wo wird er in Zukunft sein? Und wir haben festgestellt, dass Kataloge, die öffentlich zugänglich sind - sei es jetzt von Russland oder von den USA - dass die weder präzise noch vollständig genug sind, um richtige Kollisionsanalysen rechnen zu können. Deswegen haben wir angefangen, ein Sensornetzwerk aufzubauen. Wir haben derzeit eine Station in Südafrika und Australien. Die werden ferngesteuert von uns betrieben. Die Beobachtungsdaten werden korreliert zu Bahninformationen, und diese Bahninformationen sollen dann zu einem späteren Zeitpunkt - Ende des Jahres wenn es gut läuft, vermutlich aber erst Anfang des nächsten Jahres - sollen die dann verwendet werden, um Kollisionsanalysen zu machen. Sie haben zwei verschiedene Sensortypen - das sind Teleskope und Radare. Teleskop das kostet deutlich unter einer Million. Die Station, inkl Betrieb über 10 Jahre. Und ein Radar - wenn sie da ein Anständiges haben wollen - da sind Sie eher bei einer Milliarde Euro.

Christopher Olson: Da wird ja auch die GESTRA entwickelt?

Hauke Fiedler: Ja, das kommt immer drauf an, wie Sie das System.. Das GESTRA, das ist German Experimental Space and Surveillance Tracking Radar, das ist richtig. Damit können Sie vereinzelnde Sachen tracken. Die Größe darf ich Ihnen nicht nennen. Was sinnvoll wäre, ist, dass man da auf bis ca. 2-3cm runter geht, und das schafft GESTRA nicht. Die USA bauen da gerade den space fence auf leolabs hat mit dem Kiwi Radar, was in Neuseeland ist, damit können sie 2-3cm große Objekte tracken, verfolgen und auch katalogisieren und das ist auch wo man hin muss. Radar ist allerding immer viel teurer als ein optisches Gerät. Sie sehens hier bei mir im Hintergrund (Zoom Hintergrundbild), das ist unsere Teleskopstation in Australien, das ist ein 3,5m Dom mit nem 50cm und einem 25cm Teleskop. Das kann man noch relativ leicht einfach gut kaufen und betreiben. Das ist noch tatsächlich so, dass die Ausweichmanöver, die bei uns stattfinden, die werden derzeit mit den Daten die uns zur Verfügung stehen - bei einem Kilometer weicht man da schon aus. Das sind natürlich sehr viele Manöver. Wenn man sagt, man kommt bei einem Objekt im GEO auf einen Kilometer nahe - Kilometer ist sehr viel, bei uns schaffen wir es mit unseren Daten, dass wir Bahninformationen haben. Anforderung war unter 500m und wir sind jetzt deutlich unter 100 m und damit hat man natürlich ganz andere Herangehensweise. Man muss nicht mehr so viel ausweichen, wenn man weiß, dass die Daten präziser sind, man spart dabei sehr viel Treibstoff, man spart Manöver usw. und ein Satellitenbetreiber, der Fernsehsatelliten betreibt hat gesagt, ein Ausweichmanöver kostet ungefähr ein Monat Revenue (Gewinn) von der Lebenszeit her. Das sind 3 Millionen Euro. - damals.. Das sind schon ein paar Jahre her. D.h. Sie können sich vorstellen, wenn Sie jetzt die Anzahl der Manöver um einen Faktor 10 verringern können, das macht jeder.

Christopher Olson: In dem Szenario, was ich in der Arbeit verwendet habe, da hätte man jetzt zum Beispiel ausweichen müssen, oder?

Hauke Fiedler: Die Wahrscheinlichkeit war jetzt 10^5 da kommt es noch auf die radiale Komponente drauf an. Wenn die radiale Komponente sehr gering ist, dann weicht man - also wir würden ausweichen. Das macht auch jedes Raumfahrtzentrum und Satellitenbetreiber anders. Es gibt Leute, die rechnen mit zwei Wahrscheinlichkeiten. Die nehmen eine assumed probability noch dazu, verändern dann die Radien der Ellipse, um dann zu sehen wie sich die zwei Wahrscheinlichkeiten zu einander verhalten. Das machen die Franzosen. Dann gibt es welche, die gehen rein nach der Wahrscheinlichkeit 10^-5 , es wird nicht ausgewichen, 10^-4 es wird ausgewichen. Und dann gibt es noch andere, die sagen, wir nehmen noch eine radiale Komponente + Wahrscheinlichkeit + Erfahrungswert aus der Vergangenheit hinzu, das ist z.B. das, was wir bei uns hier machen. Da gab es einmal den Fall, ein GTO Objekt, d.h. es kam senkrecht von oben, unser Satellit ist eher kreisförmig geflogen. Die GTOs, die haben ja sehr hole Exzentrizität und da hat der Kollege der dafür verantwortlich gesagt: Rechnerisch sollte nichts passieren, wir machen ein Ausweichmanöver. Weil er einfach gesagt hat, das ist ihm zu kritisch. Also von daher. Wer wann wie ausweicht, das ist immer eine sehr sehr spezielle Frage.

Christopher Olson: Und das kann auch immer jeder selbst entscheiden, nehm ich an? Also jeder Betreiber?

Hauke Fiedler: Ganz genau, weil der Betreiber hat die Verantwortung und da hilft das auch nicht, wenn sie z.B. die Weltraumlagezentren haben, die für die Hoheitlichen Satelliten zuständig sein sollten. Wenn die sagen “hier die Wahrscheinlichkeit ist ..” “weicht nicht aus” oder “weicht aus” dann sagen wir “ihr übernehmt die Verantwortung” und dann sagen sie “nein, ihr seid der Betreiber”, und dann sagen wir “gut, dann rechnen wir das auch selber”. Und das ist etwas was nicht funktioniert. Sie können einem Betreiber nicht - oder einem Piloten, das ist ja ganz genau das gleiche. Der Pilot vom Flugzeug hat immer noch die letztendliche Entscheidungsgewalt, der entscheidet, ob er ausweicht oder nicht ausweicht, da kann der Fluglotse so viel sagen wie er will, wenn er sagt “ich weich jetzt aus” dann macht er das und trägt auch die Verantwortung.

Christopher Olson: Gut, dann hätte ich noch ein paar Fragen zum Thema VR. Haben Sie denn schon Erfahrung mit VR, oder haben Sie schonmal eine VR Brille aufgehabt?

Hauke Fiedler: Einmal, aber das ist schon ein bisschen länger her. Und von daher würde ich sagen: Ganz, ganz wenig Erfahrung.

Christopher Olson: Ok. Und hätten Sie jetzt das Gefühl, wenn Sie jetzt, nachdem ich Ihnen ein wenig gezeigt habe, wie man alles anwenden könnte und wenn ich Ihnen jetzt die Brille aufsetzen würde, ob Sie dann damit zurecht kämen, oder bräuchten Sie dann noch eine Einweisung noch dazu?

Hauke Fiedler: Ah, ich glaub, da kommt man schon zurecht. Und man hat ja heutzutage sehr sehr viel mit irgendwelchen Mobiltelefonen, die ja auch irgendwie sehr intuitiv sind, (zu tun). Und es war glaube wie viele Knöpfe? Warens 7 insgesamt?

Christopher Olson: Genau, wenn man bei Trackpad 3 Knöpfe zählen möchte, entschuldige 4 Knöpfe zählen möchte, dann sind es insgesamt - ne dann sind es nur 6 Knöpfe.

Hauke Fiedler: Von daher, ich glaube nicht, dass ich da relativ große Schwierigkeiten hab. Würde mir sehr viel Spaß machen!

Christopher Olson: Ja, leider hat das nicht geklappt, aber vielleicht beim nächsten Mal ja dann! Ich werd das Programm auf jeden Fall rüberschicken zum DLR, vielleicht bekommen Sie ja dann nochmal die Möglichkeit, das dann auszuprobieren. Also man kann es sowohl links als auch rechtshändig bedienen und braucht eben auch nur einen Controller dafür. Es gibt keine Unterschiede zwischen die Controllern. Wenn man zwei Controller hätte, könnte man mit beiden gleichzeitig etwas machen, aber halt eben nur die selben Befehle ausführen. Ich kann jetzt leider nicht nach Motion Sickness oder sowas fragen, weil Sie jetzt die Brillle nicht aufhatten. Da gibt es bei VR nämlich oft ein Problem, das nennt sich Motion Sickness. Das ist wie beim Autofahren, wo der Kopf wahrnimmt, dass man sich bewegt, aber die Beine sich nicht bewegen und der Kopf daher komplett durcheinander kommt und man sich eben übel fühlt. Das habe ich versucht zu vermeiden, indem ich keine automatisierten Kamerabewegungen (eingebaut) habe. Wenn man sich mit Kugelkoordinaten herum bewegt, dann bleibt die Sicht gleich. Das einzige Problem ist, dass man irgendwann nicht mehr die Erde anschaut, wenn man um die Erde herumreist, außer man dreht sich selbst. Aber ich habe jetzt aus eigener Erfahrung gemerkt, dass das zumindest die Motion Sickness verringert. Wenn man da jetzt die ganze Zeit gedreht wird, obwohl ich mich gar nicht drehe, dann spielt der Kopf manchmal ein bisschen verrückt.

Hauke Fiedler: Weiß ich nicht, ob ich das haben würde. Ich werde z.B. weder seekrank noch höhenkrank. Macht mir relativ wenig aus. Und jetzt auch vom Zuschauen - es ruckelt ein bisschen, aber

Christopher Olson: Ich glaube, das ist vor allem durch Zoom bedingt, bei der VR Brille, da muss man noch dazu sagen, dass man ein größeres FOV hat, also ein größeres Field of View - man hätte jetzt auch deutlich mehr gesehen, also ich musste auch immer ein bisschen aufpassen, dass die Dinge in der Mitte zu sehen waren, damit auch Sie die Objekte sehen konnten. Und es ruckelt auch nicht. Es läuft auch bei etwas 120 Bildern die Sekunde, das sind auch 30 (Bilder) mehr, als die Brille gerade angezeigt hat. Von daher ist das schon die Maximalmenge. Das ist jetzt leider der Übertragung geschuldet. Worauf ich eigentlich hinaus wollte ist, würde es Sie stören, wenn man sich immer selbst drehen um die Erde drehen müsste?

Hauke Fiedler: Nein, gar nicht. Das fände ich nicht als störend. Ganz im Gegenteil. Dadurch, dass man sich selber bewegt glaube ich, dass sogar der eigene Eindruck noch viel intensiver ist. Ich finde das sehr sehr spannend.

Christopher Olson: Ok, sehr gut. Dann hätte ich noch eine Frage zu den Ellipsoiden. Da habe ich jetzt einen transparenten Shader verwendet - man konnte eben durchgucken durch die Ellipsoide. Ich hatte auch mal einen Ansatz mit einem Wireframe Shader, also so ähnlich wie es bei FreeFlyer verwendet wird. Kennen Sie das? Ansonsten könnte ich das mal kurz raussuchen?

Hauke Fiedler: Wo man dann so einen Mesh hat?

Christopher Olson: Genau, ein Mesh außenrum.

Hauke Fiedler: Da finde ich den transparenten besser. Und zwar einfach deswegen. Es ist einfach optisch wesentlich eindrücklicher, sofort intuitiv greifbarer. Beim Mesh hat

man es dann vielleicht so, dass man irgendwie durchgucken kann und die Rückseite sieht, da müsste man das ganze zweifarbig machen, damit man da einen Mehrwert hat. Weiß nicht ob FreeFlyer das macht, aber das ist, was ich machen würde. Und von daher finde ich die transparenten Ellipsen (Ellipsoide) super!

Christopher Olson: Ich glaube FreeFlyer macht das nicht, die sind dann immer einfarbig.

Hauke Fiedler: Ja, und damit hat ich sag mal wenn man so ein Standbild sieht dann kann man erst durch das Auge es vor und zurückklappen lassen.

Christopher Olson: Zeigt Bild. Also so siehts bei FreeFlyer aus bspw. Ich habe dann in VR gemerkt, dass man es gar nicht so gut wahrnehmen konnte, was vorne ist und wie Sie schon sagten, man bräuchte dann vermutlich verschiedene Farben, um das gut darzustellen. Für wie sinnvoll halten Sie denn die Überschneidungsvisualisierung? Da hatten wir glaube ich auch schon bei meiner Zwischenpräsentation kurz drüber gesprochen.

Hauke Fiedler: Ja doch, das ist sinnvoll. Auf jeden Fall. Würd ich beibehalten. Dass man dann auch beim TCA direkt hinfahren kann, und dann auch sehen können "was ist jetzt der TCA" und evtl. noch Time of highest Probability. Weiß ich nicht, ob das sinnvoll ist. Obwohl der ist vermutlich ziemlich dicht beim TCA dabei, von daher glaube ich nicht, dass das ein Mehrwert bringt. Ich würds so lassen.

Christopher Olson: In diesem Fall war der glaube ich 0,36sec von dem TCA entfernt. Aber der wird auch glaube ich gar nicht in den CDMs mit angegeben, oder? Also zumindest nicht in dem CDM, den ich hatte.

Hauke Fiedler: Nein, nein.

Christopher Olson: Wie würde man den denn. Kann man den denn noch irgendwie dazu berechnen?

Hauke Fiedler: Keine Ahnung, das fiel mir einfach dazu ein. (lacht)

Christopher Olson: Ich wüsste jetzt nämlich gerade noch nicht, wie ich da rankomme, deswegen frage ich. (lacht)

Hauke Fiedler: Ich würd das so lassen, das ist völlig in Ordnung. Ich kann Ihnen aber in der Zwischenzeit selber etwas zeigen. (Zeigt und erläutert einen Vortrag vom DLR über die Analyse von einer Kollision in der Biplane).

Christopher Olson: (nach der Präsentation) sehr schön, danke! Dann würde ich jetzt zu den letzten 2-3 Fragen kommen. Meinen Sie, dass die Teleportation zu einem Satelliten dafür sorgen könnte, dass man die Orientierung verliert, oder glauben Sie, dass die Erde als Referenz dazu hilft?

Hauke Fiedler: Die Erde als Referenz hilft, man verliert natürlich die Orientierung, weil man nicht weiß, wo man gerade ist und wo das neue Objekt ist. Das kann natürlich auf der anderen Seite sein. Und gerade, wenn Sie zu LEO Objekten gehen, die ja relativ erdnahe sind, wie z.B der ISS oder auch das, was drunter ist, kann es natürlich sein, dass Sie über den Blickwinkel so viel Erde sehen, dass Sie gar nicht mehr wissen, wo bin ich, wo komm ich her. Da kann ich mir vorstellen, dass, wenn man

da irgendwo hinfahren würde auf einer Dreieck Turie, das kann auch relativ schnell gehen, dass das vielleicht eine sinnvolle Anpassung sein könnte. Man müsste das ausprobieren und dann auch wirklich beides im direkten Vergleich ausprobieren und sehen, was dann tatsächlich geschickter ist. Die Teleportation erstmal klar notwendig auf jeden Fall, wie man da jetzt hinkommt Teleportation oder Dreieck Turie (Dreieck Tour?) muss man erstmal ausprobieren.

Christopher Olson: Genau. Das wäre noch die andere alternative Möglichkeit gewesen, um zu Objekten zu kommen. Und wie empfinden sie die drei verschieden Modi die es gibt? Würde Ihnen da noch ein Modus fehlen, oder sind die Modi auch sinnvoll jeweils. Was halten Sie denn generell von ihnen?

Hauke Fiedler: Das fand ich, das war gut. Wenn es jetzt darum geht, dass man da noch mehr haben möchte weiß ich nicht, ob das sinnvoll wäre, das zu implementieren, also was man auf jeden Fall noch machen kann, wenn Sie noch etwas implementieren wollen. Ich schau mal kurz. Das war bei dem Feld, da würd ich auf jeden Fall noch die Crossbar ID mitbringen. (In Gedanken) Nein das ist nicht drin. Das ist auch nicht da drin. Was interessant sein kann bei der Information vom Satelliten ist Apogee und Perigee, Typ, aktuelle Höhe (inklination) und eventuell noch die Geschwindigkeit. Das würd ich vielleicht noch reinschreiben.

Christopher Olson: Ok.

Hauke Fiedler: Und die Crossbar ID, die würd ich noch zusätzlich zur NORAD ID dazunehmen.

Christopher Olson: Welche ID?

Hauke Fiedler: International Designator oder Crossbar ID (Könnte auch anders geschrieben sein, habe nur den International Designator in der TLE gefunden, ist aber scheinbar nur ein anderer Name dafür). Das ist das gleiche, steht auch im TLE drin. Das wären dann z.B. 2009-50b oder so was.

Christopher Olson: Sie meinen jetzt bei der Satellitenansicht, nicht bei der CDM Analyse.

Hauke Fiedler: Nicht beim CDM, genau, sondern rein bei der Satellitenansicht wo Sie sagen, da hatten Sie ja dieses Infofenster aufgemacht und das waren nur 3 Einträge. 2 oder 3.

Christopher Olson: 2

Hauke Fiedler: 2?

Christopher Olson: Also der Satellit Name, wenn man den dazuzählen möchte, Objekttyp und die NORAD ID. Also das war auch nur ein Showcase, um zu zeigen, dass man alles mögliche aus den TLE Daten parsen könnte. Aber gut, da könnten Sie mir eben sagen, was da besonders sinnvoll ist, weil ich mir da nicht ganz sicher war, was da alles reingehören würde.

Hauke Fiedler: Also, was ich auf jeden Fall sinnvoll finde ist Apogee und Perigee, damit man weiß, wie ist die Exzentrizität, was hat man da für Höhenänderungen. Dann die Inklination und vielleicht Geschwindigkeit. Momentan aktuelle Geschwindigkeit oder so. Das könnte sinnvoll sein.

Christopher Olson: Ja.

Hauke Fiedler: Ansonsten bei den CDMs, das hatte ich schon gesagt mit den Komponenten in RTN, wobei da natürlich die radiale am entscheidendsten ist. Aber wenn sie alle drei bringen, dann hat man gleich alle drei. Die Tangentiale Komponente ist glaube ich eher uninteressant, weil die schwankt am größten. Über die kann man eh keine Aussage treffen, von daher. Aber die Radiale, das wäre sehr gut.

Christopher Olson: Welche Vorteile sehen Sie denn im Vergleich zu einer Desktop Anwendung bei einer VR Anwendung?

Hauke Fiedler: Ich möchte Sie da jetzt nicht enttäuschen.

Christopher Olson: Genau, dann sagen sie ruhig auch die Nachteile, die benötige ich auch.

Hauke Fiedler: Ich sehe das gerade so für Veranstaltungen für PR Veranstaltungen und sowas halte ich das für sehr relevant. Weil das ist was man einem Politiker zeigen kann, wo er sagt "Oh ja, hier gibt es eine Kollision, hier seh ich die zwei Ellipsen. Oh ja, meine Herren, da müssen wir jetzt was machen - Ihr braucht unbedingt Geld, hier habt ihr viel Geld damit ihr das viel besser machen könnt". Das halte ich für sehr wesentlich. Und das ist auch der Teil, wo man auch auf Konferenzen wie im IAC oder sowas, wo man sowas präsentieren kann und wo ich das auch präsentieren würde. Wenn Sie bei mir im Team wären, würde ich sagen "Hinfahren, präsentieren, fertig". Das wär überhaupt gar keine Frage für mich. Wenn es jetzt darum geht, dass Sie Manöver rechnen und sagen "ok, ich hab hier jetzt 3-4 Alternativ Manöver, ich mach das zum späteren/früheren Zeitpunkt" da sind Sie am Desktop da sehen Sie die Zahlen, da wissen Sie, welche Zahlen sinnvoll sind, sie können die Zahlen einschätzen. Das geht viel schneller! Das mit einer VR Brille zu machen oder sowas. Ich glaube, bis sie da soweit sind, dass Sie sagen "ok, jetzt habe ich alles". Die Tastatur, die Finger sind einfach viel schneller, als sie irgendwas mit dem Laser einstellen können. Und präziser. Und von daher sehe ich da doch eine Trennung, wenn es wirklich um das operationale Berechnen und die tatsächlichen Manöver geht. Da ist die Desktopanwendung mit den ganzen Algorithmen, die man da entwickelt hat usw., das ist da sinnvoller. Ich habe Ihnen von dieser BACARDI Maschine erzählt, wo wir die ganzen Berechnungen machen. Wo wir auch so ein Screening machen "Wer trifft wen", wo wir dann auch die Kollisionsberechnungen machen da wird z.B. Berechnungen von der Biplane oder sowas - das wird zum Teil automatisiert werden, aber nicht ganz. Die Flugdynamiker haben gesagt "ja das und das, diese 5 Schritte 5-6 Schritte automatisch, das müssen wir eh immer machen: Absolut sinnvoll". Wenn es da um die Manöverplanung geht, die wir dann wieder zurückspielen in das System, um zu sehen, was kommt dabei raus ist dieses Manöver sinnvoll oder nicht oder wir haben hier eine handvoll Manöver, die wir reinstecken wollen, das ist dann wieder eine rein manuelle Betreiber Operator Sicht, wo es dann auch.. wo ich sag na klar, man kann prinzipiell alles automatisieren, wo die aber ein paar sehr gute Argumente haben uns sagen: "Nein, nicht alles". Und solche Sachen.

Christopher Olson: Genau die ließen sich noch schwer in VR wirklich umsetzen. Aber wie wäre es denn z.B. mit einer Sprachsteuerung? Könnte man dann schneller Dinge bedienen, wenn man das einfach über die Sprache mitteilen könnte an das System und das System dann die Arbeiten automatisiert durchführt?

Hauke Fiedler: Ja. Auf jeden Fall. Also eine Sprachsteuerung - wenn sie dann funktioniert

Christopher Olson: Sieht man ja bei Autos heutzutage manchmal. (lacht)

Hauke Fiedler: (lacht) genau. “Anrufen, Nachricht schicken oder sonstwas?” - “Anrufen!” - “Anrufen, Nachricht schicken oder” - “Anrufen!”

Christopher Olson: Genau!

Hauke Fiedler: Düt düt, weg.. Genau. Also das ist dann natürlich relativ hinderlich. Ansonsten, wenn so eine Sprachsteuerung funktioniert - Sie funktioniert ja auch teilweise schon sehr gut, gerade bei den Mobiltelefonen oder sowas. Wenn man da die großen Anbieter hat, Alexa und wie sie sonst alle heißen. Das funktioniert dann sehr gut. Und ich glaube, dass man sich da dann auf ein paar Schlüsselwörter festlegen kann, die man dann auch sehr gut umsetzen kann.

Christopher Olson: Dann hab ich jetzt noch eine letzte Frage, und zwar, ob Sie die Möglichkeit, die Ellipsoiden per Hand fortzubewegen als sinnvoll erachten.

Hauke Fiedler: Auf jeden Fall. Auf jeden Fall. Und was dann natürlich auch glaube ich gut wäre, wenn man die Abstände in einem Koordinatensystem hätte, wo man dann auch sieht, wenn man dann das Ellipsoid vor und zurück bewegt, wie sich dann die relativen Abstände verhalten. Und zwar z.B. der Gesamtabstand, dass man z.B sieht, ich hab jetzt TCA ist 60m und ich bin jetzt noch 3km weg, 2, 1, 560m und dann gehts wieder rauf. So was könnte vielleicht auch noch

Christopher Olson: Und das am besten auch am Ellipsoid selbst visualisieren und nicht im Menü, oder?

Hauke Fiedler: Ganz genau, ganz genau. Entweder am Ellipsoid selber oder irgendwo im Gesichtsfeld wo man’s dann sieht.

Christopher Olson: Wären dann noch weitere Informationen am Ellipsoid sinnvoll, die man bereits dort sehen kann?

Hauke Fiedler: Ich glaub nicht. Ich glaube, wenn man da noch den Abstand dazu hat, das könnte vielleicht noch ganz sinnvoll sein. Die Sache ist natürlich die, dass man nicht versuchen sollte, das Ganze zu überfrachten. Wenn man dann nämlich von der Information überschlagen wird, wenn man da nämlich zu viel hat, dann hilft einem das auch nicht mehr weiter.

Christopher Olson: Genau. Ja, dann danke ich Ihnen, das waren eigentlich alle meine Fragen, außer ich hab jetzt irgendwas übersehen Patrick (Betreuer, hat mitgeschrieben)?

Patrick Saalfeld: (Fragt nochmal nach dem Lebenslauf).

Hauke Fiedler: Ursprünglich bin ich Astrophysiker. Hab dann nochmal erstmal Allgemeine Physik studiert, dann Astrophysik - darauf hab ich promoviert. Und hab dann Jahre später nochmal einen Master of Space System Engineering gemacht. Habe im Radar Institut bei HR gearbeitet, hab dann für den Vorstand gearbeitet DLR, und bin jetzt bei RB - Raumflugbetrieb und Astronautentraining heißt das offiziell - DLR / RB ist das und hab hier eine Gruppe SSA aufgebaut vor 10 Jahren. Ja

ziemlich genau 10 Jahren, im September sind es 10 Jahre und baue jetzt das Teleskop Netz SmartNet auf und die Prozessierung Maschine BACARDI und versuche damit Weltraumschrott zu minimieren, indem wir gute Daten haben, wo wir dann eben Ausweichen können.

Patrick Saalfeld: (Lobt seinen Lebenslauf)

Hauke Fiedler: Ja, macht Spaß! Und falls Sie beide - die Einladung gilt für beide - wenn Sie hier mal nach Oberpfaffenhofen kommen, geben Sie mir vorher Bescheid, dann kann ich Ihnen die Kontrollräume hier zeigen.

Christopher Olson: Das klingt ja super, das ist bei München oder?

Hauke Fiedler: Das ist bei München, ganz genau. Ungefähr 15km westlich. Es geht auch ein S-Bahn raus. Von der S–Bahn kann man dann nochmal mit dem Bus fahren oder 20 min zu Fuß gehen, der Weg ist direkt an der Hauptstraße, das ist nicht gerade das schönste. Ich würde da versuchen, ein Bus zu nehmen. So die Pandemie mal dann vorbei sein sollte und Sie hier in der Gegend sind. Sie haben meine E-Mail Adresse, schreiben Sie mir eine E-Mail, dann kann ich Ihnen das zeigen, weil das ist wirklich sehr interessant. Wir hier Betrieb vom Columbus Modul von der ISS. Das europäische Modul, das wird bei uns betrieben. Und da gibt es so eine Besucherbrücke, wo man dann unten direkt reinschauen kann, wo das dann auch eben alles sieht, wie da gerade gearbeitet wird. Live-Schalte zur ISS!

Christopher Olson: Vielen Dank für die Einladung!

Hauke Fiedler: Ja gern!

Patrick Saalfeld: (Fragt wegen Pandemie)

Hauke Fiedler: Ja genau, für Besucher ist seit März letzten Jahres geschlossen und bei uns gibt es immer die .. die Regelungen, dass derzeit ist bis 15. August noch alles zu und man muss dann einfach sehen, wie das dann alles ist.

Christopher Olson: Genau. Ok, dann danke Ihnen für die .. für Ihre Zeit - war ja dann doch deutlich länger, als ich eigentlich angekündigt hatte, aber ich hoffe, das war nicht schlimm.

Hauke Fiedler: Ganz im Gegenteil. Und wenn Sie herkommen, bringen Sie die Brille mit!

Christopher Olson: Auf jeden Fall. Mach ich dann! Ja und - falls Sie keine weiteren Fragen haben, würde ich sagen - hören wir noch voneinander, vielleicht auch bei meiner Abschlusspräsentation, wenn Sie Lust haben, können Sie natürlich auch gerne mit sich das Ganze anhören. Dann würde ich Ihnen dann auch noch ein Link zu schicken.

Hauke Fiedler: Das wär super. Ich kann nicht versprechen, ob ich dabei sein kann oder nicht, weil teilweise ist bei mir die Zeit recht eng, recht knapp bemessen. Aber das ist, falls es mir möglich sein sollte, würde ich sehr gern dabei sein. Auf jeden Fall.

Christopher Olson: Genau, also nur falls Sie Zeit haben. Machen Sie sich da keinen Stress.

Hauke Fiedler: Wär spannend!

Christopher Olson: Genau, dann würde ich noch mit Patrick hier drin bleiben, um kurz zu quatschen.

Hauke Fiedler: Dann nochmal vielen Dank, war wirklich sehr spannend. Und wirklich sehr, sehr eindrücklich. Herzlichen Glückwunsch.

Christopher Olson: Dankeschön!

Hauke Fiedler: Dann wünsche ich noch einen schönen Tag, bis demnächst!

Christopher Olson: Wünsche ich Ihnen ebenfalls!

Literaturverzeichnis

Gae Ae Ryu und Kwan-Hee Yoo. Key factors for reducing motion sickness in 360° virtual reality scene: Extended abstract. In *The 25th International Conference on 3D Web Technology*, Web3D '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381697. doi: 10.1145/3424616.3424723. URL https://doi.org/10.1145/3424616.3424723. (zitiert auf Seite 31)

a.i. solutions. Freeflyer capabilities q2 2020. https://ai-solutions.com/wp-content/uploads/2021/01/FreeFlyer_Capabilities_Q2_2020.pdf, 2020. Letzter Zugriff: 26.07.2021. (zitiert auf Seite 6)

Alex Benton. Gpu ray marching. URL: https://www.cl.cam.ac.uk/teaching/1718/AdvGraph/5.%20GPU%20Ray%20Marching.pdf, 2017. (zitiert auf Seite 14)

Oleksii Bashkanov, Patrick Saalfeld, Hariharasudhan Gunasekaran, Mathews Jabaraj, Bernhard Preim, Tobias Huber, Florentine Hüttl, Werner Kneist, und Christian Hansen. Vr multi-user conference room for surgery planning. In *18. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie e.V.*, pages 264–268, 2019. URL https://www.youtube.com/watch?v=c4YneeMREvM, YouTube. (zitiert auf Seite 6)

Pietro C. Cacciabue und Carlo Cacciabue. *Guide to Applying Human Factors Methods*. SpringerVerlag, 2004. ISBN 1852337052. (zitiert auf Seite 5)

CCSDS 508.0-B-1. Conjunction data message. Recommended standard, CCSDS, June 2013. URL https://public.ccsds.org/Pubs/508x0b1e2c1.pdf. (zitiert auf Seite 11)

S. Chavan. Augmented reality vs. virtual reality: Differences and similarities. 2016. (zitiert auf Seite 16)

Camilla Colombo, Nicoletta Di Blas, Ioannis Gkolias, Pier Luca Lanzi, Daniele Loiacono, und Erica Stella. An educational experience to raise awareness about space debris. *IEEE Access*, PP:1–1, 05 2020. doi: 10.1109/ACCESS.2020.2992327. (zitiert auf Seite 7)

Adrian Constantin und R. Johnson. Large gyres as a shallow-water asymptotic solution of euler's equation in spherical coordinates. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 473:20170063, 04 2017. doi: 10.1098/rspa.2017.0063. (zitiert auf Seite 13)

ESA. Space debris: assessing the risk. URL: https://www.esa.int/About_Us/ESOC/Space_debris_assessing_the_risk, 2005. Letzter Zugriff: 22.08.2021. (zitiert auf Seite 1)

ESA. Esa commissions world's first space debris removal. URL: https://www.esa.int/Safety_Security/Clean_Space/ESA_commissions_world_s_first_space_debris_removal, 2019. Letzter Zugriff: 22.08.2021. (zitiert auf Seite 2)

ESA. The cost of space debris. URL: https://www.esa.int/Safety_Security/Space_Debris/The_cost_of_space_debris, 2020. Letzter Zugriff: 22.08.2021. (zitiert auf Seite 2)

ESA. Space debris by the numbers. URL: https://www.esa.int/Safety_Security/Space_Debris/Space_debris_by_the_numbers, 2021a. Letzter Zugriff: 22.08.2021. (zitiert auf Seite 1)

ESA. Ssa programme overview. URL: https://www.esa.int/Safety_Security/SSA_Programme_overview, 2021b. (zitiert auf Seite 9)

Peter Fromberger, Kirsten Jordan, und Jürgen L. Müller. Virtual reality applications for diagnosis, risk assessment and therapy of child abusers. *Behavioral Sciences & the Law*, 36(2):235–244, 2018. doi: https://doi.org/10.1002/bsl.2332. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bsl.2332. (zitiert auf Seite 6)

Johannes Gelhaus, Noelia Sánchez-Ortiz, Vitali Braun, Christopher Kebschull, Raúl Domínguez-González, Carsten Wiedemann, Holger Krag, und Prof Vörsmann. Upgrade of drama esa's space debris mitigation and analysis tool suite. 01 2013. (zitiert auf Seite 6)

Feyza Merve Hafizoglu und Roger Mailler. Telescope management for satellite tracking: A decentralized approach. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '13, page 1213–1214, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450319935. (zitiert auf Seite 10)

John C. Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12:527–545, 1994. (zitiert auf Seite 15)

F. R. Hoots und R. L. Roehrich. Spacetrack report no. 3: Models for propagation of NORAD element sets. Technical report, Aerospace Defense Center, Peterson Air Force Base, 1980. (zitiert auf Seite 11)

TS Kelso et al. Analysis of the iridium 33-cosmos 2251 collision. *Advances in the Astronautical Sciences*, 135(2):1099–1112, 2009. (zitiert auf Seite 1)

Donald J. Kessler und Burton G. Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, 83(A6):2637–2646, 1978. doi: https://doi.org/10.1029/JA083iA06p02637. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JA083iA06p02637. (zitiert auf Seite 1)

Donald J Kessler, Nicholas L Johnson, JC Liou, und Mark Matney. The kessler syndrome: implications to future space operations. *Advances in the Astronautical Sciences*, 137(8):2010, 2010. (zitiert auf Seite 1)

J.-C. Liou. An analysis of the fy-1c, iridium 33, and cosmos 2251 fragments. *ARES Biennial Report 2012 Final*, pages 75–76, 2014. URL https://ntrs.nasa.gov/citations/20150003820. (zitiert auf Seite 1)

D. Mehrholz, Ludger Leushacke, W. Flury, Rüdiger Jehn, H. Klinkrad, und M. Landgraf. Detecting, tracking and imaging space debris. *Fraunhofer FHR*, 109, 02 2002. (zitiert auf Seite 9)

NASA. Space debris and human spacecraft. URL: https://www.nasa.gov/mission_pages/station/news/orbital_debris.html, 2021. (zitiert auf Seite 9)

Daniel Paes, Eduardo Arantes, und Javier Irizarry. Immersive environment for improving the understanding of architectural 3d models: Comparing user spatial perception between immersive and traditional virtual reality systems. *Automation in Construction*, 84:292–303, 2017. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2017.09.016. URL https://www.sciencedirect.com/science/article/pii/S0926580517308361. (zitiert auf Seite 16)

James Pavur und Ivan Martinovic. The cyber-asat: On the impact of cyber weapons in outer space. pages 1–18, 05 2019. doi: 10.23919/CYCON.2019.8756904. (zitiert auf Seite 10)

Patrick Puschmann, Tina Horlitz, Volker Wittstock, und Astrid Schütz. Risk analysis (assessment) using virtual reality technology - effects of subjective experience: An experimental study. *Procedia CIRP*, 50:490–495, 2016. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2016.04.115. URL https://www.sciencedirect.com/science/article/pii/S2212827116303031. 26th CIRP Design Conference. (zitiert auf Seite 5 und 6)

Inigo Quilez. Ellipsoid sdf. https://iquilezles.org/www/articles/ellipsoids/ellipsoids.htm, 2008. Letzter Zugriff: 25.05.2021. (zitiert auf Seite 37)

Waleed Salem. Combining vr technology and human factors methods for supporting risk analysis. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6, 2008. doi: 10.1109/ICTTA.2008.4530380. (zitiert auf Seite 5)

Martin Stoffers, Michael Meinel, Martin Weigel, Martin Siggel, Hauke Fiedler, Kathrin Rack, und Yi Wasser. Bacardi: A system to track space debris. In *ESA NEO and DEBRIS DETECTION CONFERENCE - EXPLOITING SYNERGIES* -, Februar 2019. URL https://elib.dlr.de/126572/. (zitiert auf Seite 8)

C. Thome. Using a floating origin to improve fidelity and performance of large, distributed virtual worlds. In *2005 International Conference on Cyberworlds (CW'05)*, pages 8 pp.–270, 2005. doi: 10.1109/CW.2005.94. (zitiert auf Seite 19)

Marit Undseth, Claire Jolly, und Mattia Olivari. Space sustainability. (87), 2020. doi: https://doi.org/https://doi.org/10.1787/a339de43-en. URL https://www.oecd-ilibrary.org/content/paper/a339de43-en. (zitiert auf Seite 1)

David Vallado und Paul Cefola. Two-line element sets - practice and use. *Proceedings of the International Astronautical Congress, IAC*, 7:5812–5825, 01 2012. (zitiert auf Seite 11)

David Vallado, Paul Crawford, Richard Hujsak, und T.S. Kelso. Revisiting spacetrack report# 3: Rev. 08 2006. (zitiert auf Seite 11)

Martin Wermuth. Space situational awareness (ssa). URL: https://www.dlr.de/rb/desktopdefault.aspx/tabid-10156/17320_read-41664/, 2021. (zitiert auf Seite 9)

H Wilden, C Kirchner, O Peters, N Ben Bekhti, R Kohlleppel, A Brenner, und T Eversberg. Gestra-technology aspects and mode design for space surveillance and tracking. In *Proceedings of the 7th European Conference on Space Debris*, 2017. (zitiert auf Seite 10)

Aleksandra Zarichin. Space debris: History, analysis and implementation of a web-based visualization system. Master's thesis, TH Köln, Januar 2017. URL https://elib.dlr.de/117558/. Arbeit betreut durch Dr. Martin Siggel. (zitiert auf Seite 8)

Jerzy Zywicki. Dlr - raumflugbetrieb und astronautentraining - gps high precision orbit determination software tools (ghost). https://www.dlr.de/rb/desktopdefault.aspx/tabid-10749/10536_read-23371/, 2021. Letzter Zugriff: 26.07.2021. (zitiert auf Seite 7)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Weiterhin wurde die Arbeit bisher weder in einem anderen Prüfungsverfahren vorgelegt noch anderweitig veröffentlicht. Textpassagen, die wörtlich oder dem Sinn nach auf anderen Quellen beruhen, sind als solche kenntlich gemacht.

Magdeburg, 25. August 2021