Landesamt für Digitalisierung, Breitband und Vermessung

Hochschule München University of Applied Sciences

Bachelorarbeit

Entwicklung eines Godot-basierten Serious Games zur Visualisierung von Geodaten des Landesamtes für Digitalisierung, Breitband und Vermessung

Bachelorarbeit

Fakultät für Geoinformation

Hochschule für angewandte Wissenschaften München

Erster Betreuer: Prof. Dr. Sven Fuhrmann (HM)

Zweiter Betreuer: Thomas Meier (LDBV)

Angefertigt von: Konstantin Zeller

Matrikelnr.: xxx

E-Mail: xxx

Studiengang: Bachelor of Engineering Kartographie | Geomedientechnik (B.Eng.)

Poing, den 12.05.2025

Kurzfassung (Abstract)

Geodaten sind heutzutage frei verfügbar und bergen ein enormes Potenzial für Bildung, Visualisierung und gesellschaftliche Nutzung. Doch vielen Menschen – insbesondere jungen Zielgruppen – ist weder deren Existenz noch deren Mehrwert bewusst. Diese Arbeit beschäftigt sich mit der Frage, wie Serious Games als innovative Vermittlungsform dazu beitragen können, Geodaten niedrigschwellig, spielerisch und motivierend zu erschließen. Mit Serious Games sind interaktive Spiele gemeint, die neben dem Unterhaltungswert gezielt zur Wissensvermittlung eingesetzt werden. Im Zentrum steht die Entwicklung eines kindgerechten 3D-Spiels mit der Game Engine Godot, in dem die Spieler in einer offenen Spielwelt virtuelle „Aliens“ einsammeln, die jeweils mit geobezogenem Wissen verknüpft sind. Bei der Gestaltung wurden pädagogische Prinzipien wie exploratives Lernen, Interaktivität und Motivation gezielt in die Spielmechanik integriert. Technische und didaktische Herausforderungen – insbesondere im Umgang mit der jungen, aber offenen Engine Godot – wurden ebenso thematisiert wie mögliche Weiterentwicklungen. Ziel war es, ein funktionierendes Beispiel zu schaffen, das sowohl die Potenziale von Geodaten als auch die Stärke spielerischer Wissensvermittlung demonstriert.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In den letzten Jahren haben sich die technischen Möglichkeiten zur Entwicklung interaktiver 3D-Anwendungen rasant weiterentwickelt. Leistungsfähige 3D-Grafik ist längst nicht mehr teuren Spezial-Workstations vorbehalten; preisgünstige PC-Grafikkarten liefern heute die schnellsten und realistischsten Simulationen, wie Lewis und Jacobson (Jacobson, 2002, S. 27) hervorheben. Hochentwickelte Render-Pipelines und interaktive Physik-Module, die früher nur in aufwendig ausgestatteten VR-Laboren zum Einsatz kamen, sind mittlerweile integrale Bestandteile moderner Game Engines und somit frei verfügbar. Gleichzeitig wächst die Bandbreite an Visualisierungsplattformen stetig. Fortschritte in der Computergrafik machen es möglich, räumliche Szenarien nicht nur in klassischer 3D-Perspektive darzustellen, sondern auch als immersive VR-Walkthroughs, in CAVE-Umgebungen oder als mobile Anwendungen. Fritsch (Fritsch, 2003, S. 281) betont, dass insbesondere geo-bezogene Disziplinen von diesen Entwicklungen profitieren, da sie zunehmend auf eine anschauliche Darstellung komplexer räumlicher Zusammenhänge angewiesen sind. Diese technischen Fortschritte haben auch zur rasanten Entwicklung des Forschungsfelds der Serious Games beigetragen. Laamarti et al. (Laamarti et al., 2014, S.3) zeigen in einer umfassenden Untersuchung, dass die Zahl wissenschaftlicher Publikationen zu Serious Games seit Ende der 1990er-Jahre exponentiell gestiegen ist – ein deutliches Indiz für das zunehmende Interesse der Forschungsgemeinschaft. Parallel dazu hat auch die Industrie diesen Trend aufgegriffen: Der Markt für Serious Games hat sich stark erweitert und wurde bereits 2015 auf ein Volumen von 10 Milliarden Euro geschätzt. Diese Entwicklungen unterstreichen das enorme Potenzial von Serious Games – sowohl als didaktisches Werkzeug als auch als innovatives Medium zur Vermittlung komplexer Inhalte, etwa im Bereich der Geodaten.

1.1 Forschungsproblem

Das Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) stellt mittlerweile eine Vielzahl frei verfügbarer Geodaten bereit. Das Potenzial dieser Daten ist jedoch vielen Menschen nicht bewusst – sei es, weil das Wissen über deren Nutzungsmöglichkeiten fehlt oder weil die Daten außerhalb von Fachkreisen kaum bekannt sind. Obwohl das LDBV selbst Visualisierungsaufgaben übernimmt, mangelt es an ausreichend spezialisierten Fachkräften, um das volle Potenzial dieser Daten für Bildung, Öffentlichkeitsarbeit oder innovative Anwendungen zu erschließen. Gerade im Kontext digitaler und interaktiver Vermittlungsformen stellt sich daher die Frage:

Wie kann ein Serious Game dazu beitragen, das Bewusstsein und Verständnis für frei verfügbare Geodaten des LDBV bei einer jungen Zielgruppe zu fördern?

1.2 Zielsetzung

Ziel dieser Arbeit ist die Konzeption und Entwicklung eines Serious Games, das insbesondere Kindern, aber auch interessierten Jugendlichen, die Potenziale frei verfügbarer Geodaten des Landesamts für Digitalisierung, Breitband und Vermessung (LDBV) auf spielerische Weise näherbringt. Das Spiel soll Neugierde wecken, Entdeckergeist fördern und Freude am Erkunden vermitteln. Gleichzeitig soll es grundlegendes Wissen über verschiedene Arten von Geodaten wie digitale Orthophotos(Luftbildkarten), historische Karten, Geländemodelle und 3D-Gebäudedaten vermitteln.

Um dieses übergeordnete Ziel zu erreichen, verfolgt die Arbeit mehrere Teilziele:

Erstens soll durch interaktive Spielmechaniken wie Sammeln, Erkunden und Rätsellösen eine aktive und motivierende Wissensvermittlung erfolgen. Die Spielerinnen und Spieler sollen dabei auf unterhaltsame Weise lernen, welche Arten von Geodaten existieren, wie sie entstehen und wofür sie genutzt werden können. Zweitens soll das Spiel einen Beitrag zur Aufklärung leisten, indem es anschaulich den Mehrwert amtlicher Geodaten demonstriert. Auf diese Weise wird das öffentliche Bewusstsein für die Angebote des LDBV gestärkt und deren gesellschaftliche Relevanz sichtbarer gemacht.

Ein weiteres Ziel ist die Förderung des geoinformationsbezogenen Nachwuchses. Durch eine positive erste Begegnung mit Geodaten in einem spielerischen Kontext soll langfristig das Interesse an entsprechenden Ausbildungswegen und Berufsfeldern geweckt werden. Nicht zuletzt soll im Rahmen dieser Arbeit auch die technische Machbarkeit eines solchen Spiels gezeigt werden. Dabei wird demonstriert, dass moderne Game Engines eine leistungsfähige, ressourcenschonende und praxisnahe Grundlage für geodatenbasierte Lernanwendungen bieten können. Diese Erkenntnisse sollen eine Grundlage für zukünftige Entwicklungen in diesem Bereich schaffen.

Das Spiel ist speziell für Kinder im Alter von etwa 6 bis 12 Jahren konzipiert und verfolgt eine bewusst geschlechterneutrale und religionsneutrale Ausrichtung, um möglichst viele Zielgruppen anzusprechen. Die Spieldauer beträgt etwa fünf Minuten und enthält eine abgeschlossene, verständliche Handlung, die auch bei kürzeren Spielzeiten ein vollständiges Erlebnis bietet. Dadurch eignet sich das Projekt besonders für den Einsatz bei Messen, Ausstellungen oder Bildungspräsentationen, bei denen ein kompaktes, anschauliches und leicht zugängliches Format gefragt ist.

2 Literatur Review

2.1 Serious Games

2.1.1 Definition

Eine einheitliche, allgemein akzeptierte Definition von *Serious Games* existiert bislang nicht. In der wissenschaftlichen Literatur wie auch in der Industrie variieren die Auffassungen darüber, was ein Serious Game genau ausmacht. Dies liegt vor allem daran, dass Serious Games aus verschiedenen Komponenten bestehen – etwa Lerninhalten, Spielmechaniken und Interaktionsformen – die je nach Disziplin unterschiedlich gewichtet und interpretiert werden.
Einige Vertreter der Industrie betonen etwa, dass ein Serious Game zwingend ein echtes Unterhaltungselement mit einer praktischen, zweckgerichteten Dimension verbinden müsse (Laamarti et al., *2014, S.5*).

Definition „Serious Game“ (nach Loh et al.)
Diese Formulierung gilt heute als eine der am weitesten verbreiteten Definitionen von Serious Games:
Ein Serious Game ist ein digitales Spiel, dessen Hauptziel nicht Unterhaltung ist, sondern ein ernsthafter Nutzen – etwa für Training, Bildung oder Gesundheitsversorgung (Loh, Sheng & Ifenthaler, 2015, S. 5)

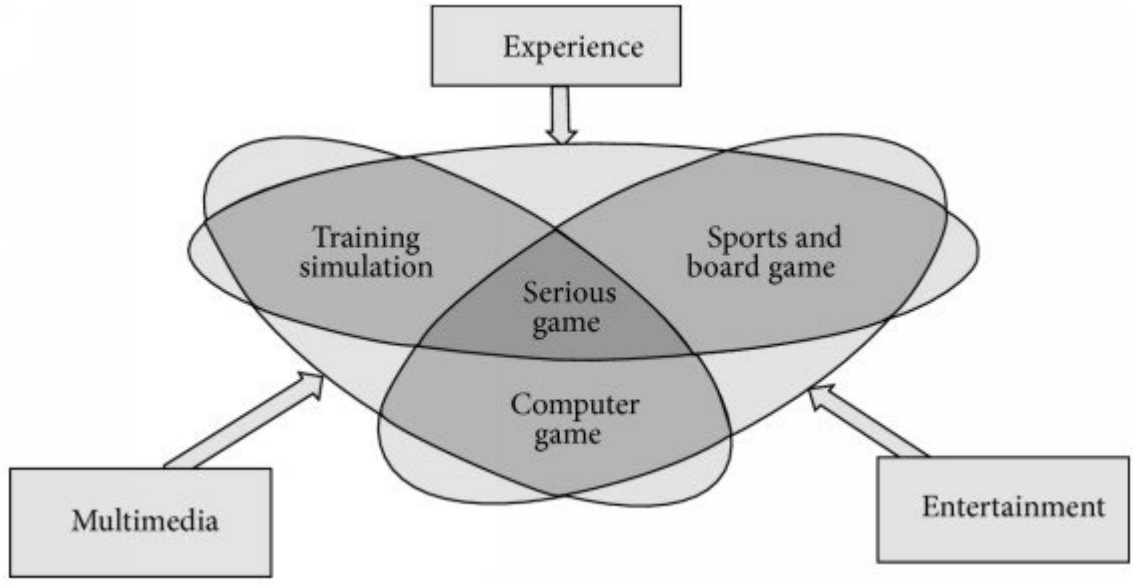


Abbildung 1: Aufbau eines Serious Games (Laamarti et al., An Overview of Serious Games, S. 4)

2.1.2 Ziel von Serious Games und wie sie das erreichen

Serious Games verfolgen eine Vielzahl übergeordneter Zielsetzungen, die je nach Anwendungsfeld variieren können. Häufig stehen dabei die Vermittlung von Wissen, das Training bestimmter Fähigkeiten sowie die Förderung von Verhaltens- und Einstellungsänderungen im Vordergrund. So werden sie etwa im schulischen oder beruflichen Kontext eingesetzt, um Lerninhalte anschaulich zu vermitteln oder praktische Kompetenzen zu trainieren – etwa in der Medizin oder Technik. Darüber hinaus können sie genutzt werden, um gesundheitsförderliches Verhalten anzuregen, nachhaltige Lebensweisen zu unterstützen oder therapeutische Prozesse zu begleiten. Ein zentrales Merkmal solcher Spiele besteht darin, dass Lern- und Handlungsziele möglichst nahtlos in die Spielmechanik eingebunden werden. Dadurch erfolgt der Wissenserwerb nicht nur kognitiv, sondern auch motiviert und erfahrungsbasiert – was die langfristige Wirkung deutlich erhöht (vgl. Kerres, Bormann & Vervenne, 2009, S.1-2).

Wie erreichen Serious Games ihre Ziele?

Serious Games nutzen eine Vielzahl von Methoden und Mechanismen, um ihre spezifischen Ziele zu erreichen. Dabei greifen sie bewusst auf psychologische, pädagogische und spieltheoretische Prinzipien zurück:

1. Motivation durch Spielmechaniken

Einer der größten Vorteile von Serious Games liegt in ihrer Fähigkeit, durch klassische Spielmechaniken wie Belohnungssysteme, Herausforderungen, Levelstrukturen oder Narrative eine hohe intrinsische Motivation bei den Spielern aufzubauen (Emmerich & Bockholt, 2018, S.287). Motivation wird damit zum Motor des Lern- und Veränderungsprozesses.

2. Lernen durch Interaktion und aktives Erleben

Serious Games unterscheiden sich von klassischen Lehrmethoden vor allem durch ihre interaktive Struktur: Lerninhalte werden nicht passiv aufgenommen, sondern aktiv durch eigenes Handeln erschlossen. Spieler übernehmen Rollen, treffen Entscheidungen und erleben deren Konsequenzen unmittelbar im Spielverlauf. Dieses „Handeln im Spiel“ fördert das sogenannte situierte Lernen, bei dem Wissen direkt im Anwendungskontext aufgebaut wird. Laut Kerres, Bormann und Vervenne (2009, S.2) entsteht durch die Verbindung von Herausforderung, Handlung und Rückmeldung eine besonders motivierende Lernsituation, in der Inhalte nicht nur verstanden, sondern erlebt und verinnerlicht werden.

3. Feedback und Adaptivität

Effektive Serious Games bieten direktes, kontinuierliches Feedback zu den Handlungen der Nutzer. Dieses Feedback unterstützt nicht nur die Lernkurve, sondern erlaubt es auch, Fehler risikofrei auszuprobieren und daraus zu lernen. Viele moderne Serious Games sind zudem adaptiv und passen Schwierigkeitsgrad oder Inhalte an die Fähigkeiten des Spielers an, was die Motivation und die Effektivität weiter steigert.(Kerres 2009, S.2-3)

4. Narrative Einbettung

Geschichten spielen in Serious Games eine zentrale Rolle, um Lerninhalte sinnhaft und emotional zugänglich zu machen. Durch erzählerische Rahmenhandlungen

lassen sich auch abstrakte oder komplexe Themen greifbar darstellen. Narrationen schaffen Identifikation, fördern das Verständnis und erleichtern die Erinnerung an zentrale Inhalte. Gleichzeitig ermöglichen sie es, Lernziele subtil in den Spielverlauf zu integrieren, ohne dass sie belehrend wirken. Besonders bei Themen mit sozialem oder emotionalem Bezug kann die narrative Gestaltung dazu beitragen, Empathie und Reflexionsfähigkeit zu fördern. Damit wird die Geschichte zum didaktischen Werkzeug – sie motiviert nicht nur, sondern strukturiert auch den Lernprozess auf einer emotional anschlussfähigen Ebene.(Kerres 2009, S.9-10)

5. Transfer in die Realität

Ein entscheidender Erfolgsfaktor von Serious Games ist es, dass die im Spiel erlernten Inhalte oder Verhaltensweisen in das reale Leben übertragen werden können. Dies wird durch realistische Szenarien, problemorientiertes Lernen und Übungen in einem sicheren Rahmen gefördert (Girard, 2013 , S. 213). Der Transfer von Spielinhalten auf Alltagssituationen ist eines der zentralen Evaluationskriterien für die Wirksamkeit von Serious Games.

Trotz des großen Potenzials von Serious Games bestehen mehrere Herausforderungen, die das Erreichen ihrer angestrebten Ziele erschweren können. Eine zentrale Schwierigkeit liegt in der Heterogenität der Zielgruppen: Spielerinnen und Spieler bringen unterschiedliche Vorkenntnisse, Fähigkeiten und Erfahrungen mit, was zu Über- oder Unterforderung führen kann, wenn diese Unterschiede nicht angemessen berücksichtigt werden (Emmerich & Bockholt, 2018, S. 268–269). Darüber hinaus ist es oft kompliziert, die Wirkung von Serious Games zuverlässig zu erfassen – insbesondere dann, wenn es um abstrakte Zielsetzungen wie das Fördern von Wohlbefinden oder die Veränderung von Verhaltensweisen geht (Emmerich & Bockholt, 2018, S. 270). Auch die Frage nach der Nachhaltigkeit der erzielten Effekte ist bislang nur unzureichend beantwortet, da viele Evaluationsstudien lediglich kurzfristige Ergebnisse betrachten, obwohl langfristige Wirkungen für viele Einsatzfelder entscheidend wären (ebd.). Zusätzlich stehen Serious Games im direkten Vergleich zu etablierten didaktischen oder therapeutischen Methoden, wodurch sie sich in ihrer Wirksamkeit erst behaupten müssen (Girard et al., 2013, S. 213).

Zusammenfassend lässt sich sagen, dass Serious Games gezielt auf das Erreichen von Lern-, Trainings- oder Veränderungszielen ausgerichtet sind, indem sie die motivierende Kraft von Spielen mit pädagogischen und psychologischen Prinzipien kombinieren. Sie ermöglichen es, Inhalte interaktiv, emotional ansprechend und nachhaltig zu vermitteln. Der Erfolg hängt jedoch stark von einer durchdachten Gestaltung, der Zielgruppenpassung und einer soliden Evaluation ab. Nur wenn Serious Games sowohl kurzfristige als auch langfristige Wirkungen erzielen und den Transfer in den Alltag sicherstellen, können sie ihr volles Potenzial entfalten und sich als ernstzunehmende Alternativen oder Ergänzungen zu traditionellen Methoden etablieren.

2.1.3 Didaktik bei Serious Games und Einsatz in der Bildung

Der Erfolg von Serious Games im Bildungsbereich hängt entscheidend von ihrer didaktischen Qualität ab. Damit solche Spiele effektiv Wissen vermitteln, Fähigkeiten fördern oder Problemlösestrategien anregen können, müssen sie auf durchdachten pädagogischen Grundlagen basieren. Wesentlich ist dabei, dass Lerninhalte nicht lediglich in eine spielerische Oberfläche verpackt werden, sondern fest mit dem Spielverlauf und den Spielzielen verwoben sind – ein Prinzip, das als „intrinsische Integration“ bezeichnet wird. Auf diese Weise wird Lernen nicht als externe Aufgabe, sondern als integraler Bestandteil des Spiels erlebt. Wichtige didaktische Elemente sind etwa klar formulierte Lernziele, kontinuierliches Feedback, motivierende Spielmechaniken wie Belohnungssysteme oder Geschichten sowie ein flexibler Schwierigkeitsgrad. Besonders effektiv sind Spiele, die nicht nur kognitive Fähigkeiten ansprechen, sondern auch emotionale und soziale Lernprozesse fördern. Die enge Verzahnung von pädagogischer Planung und spielerischem Design ist dabei entscheidend für den Lernerfolg.(Emmerich, 2016, S. 267-268)

Serious Games finden zunehmend Einzug in verschiedene Bildungsbereiche, da sie komplexe Lerninhalte nicht nur vermitteln, sondern durch Handeln erfahrbar machen können. Gerade im schulischen Kontext bieten sie die Möglichkeit, abstrakte oder schwer verständliche Themen auf anschauliche Weise zu visualisieren. Emmerich und Bockholt (Emmerich, 2016, S. 267-268) betonen, dass insbesondere durch die Verbindung von Interaktion und inhaltlicher Tiefe ein nachhaltiger Lerneffekt erzielt werden kann. Ob es um mathematische Konzepte, naturwissenschaftliche Abläufe

oder Sprachtraining geht – Serious Games ermöglichen es Lernenden, Inhalte durch wiederholte Anwendung in einem motivierenden Kontext zu verinnerlichen. Dabei profitieren besonders solche Spiele, die das Lernen eng mit den Spielmechaniken verzahnen und so nicht nur Wissen vermitteln, sondern auch Problemlösefähigkeiten und Transferdenken fördern.

Auch in der Hochschullehre und der beruflichen Weiterbildung spielen Serious Games eine zunehmend wichtige Rolle. Besonders in praxisnahen oder sicherheitsrelevanten Bereichen – wie etwa der Medizin, der Luftfahrt oder im Katastrophenschutz – ermöglichen sie es, komplexe und potenziell risikoreiche Situationen gefahrlos zu simulieren. Girard et al. (Girard, 2013, S.207-208) heben hervor, dass solche digitalen Trainingsumgebungen nicht nur die fachlichen Kompetenzen der Lernenden fördern, sondern auch deren Entscheidungsfähigkeit und Reaktionsschnelligkeit verbessern können. Durch die realitätsnahe Gestaltung der Szenarien wird ein intensives und nachhaltiges Lernerlebnis ermöglicht, das herkömmliche Lehrmethoden sinnvoll ergänzt.

Neben den didaktischen Chancen sind jedoch auch Herausforderungen zu beachten. Die Balance zwischen Unterhaltungswert und inhaltlicher Tiefe ist nicht trivial: Ein Spiel, das motiviert, aber keine nachhaltigen Lerneffekte erzielt, verfehlt sein Ziel ebenso wie ein Spiel, das zwar inhaltlich korrekt, aber spielerisch unattraktiv ist. Hinzu kommt, dass Lehrkräfte und Bildungseinrichtungen oft zunächst für die Potenziale von Serious Games sensibilisiert und entsprechend geschult werden müssen, um sie sinnvoll in bestehende Lernkontexte zu integrieren (Kerres, 2009, S.4).

Zusammenfassend zeigt sich, dass Serious Games in der Bildung ein hohes Potenzial besitzen, vorausgesetzt sie sind didaktisch fundiert konzipiert, passgenau auf den jeweiligen Einsatzbereich abgestimmt und werden von pädagogischem Fachpersonal gezielt eingesetzt. Nur dann können sie nicht nur motivieren, sondern auch messbare Lernfortschritte erzielen und langfristige Bildungsziele unterstützen.

2.2 Game Engine

2.2.1 Was ist eine game engen und wie ist diese aufgebaut

Der Begriff *Game Engine* beschreibt das technische Grundgerüst, das für die Entwicklung und Ausführung digitaler Spiele verwendet wird. Eine Game Engine ist modular aufgebaut und besteht aus verschiedenen Komponenten, die grundlegende

Funktionen wie Eingabe, Grafik, Physik, Sound und Netzwerkkommunikation ermöglichen, ohne dabei die eigentliche Spielmechanik (Game Logic) oder die Spielumgebung (Leveldaten) direkt zu definieren (Lewis & Jacobson, 2002, S. 28-29).

Moderne Game Engines beinhalten in der Regel Module zur 3D- oder 2D-Darstellung (Rendering Engine), zur Audiowiedergabe, zur Steuerung physikalischer Prozesse (z. B. Kollisionsabfragen, Gravitation) sowie zur Netzwerkintegration für Mehrspieler-Funktionalität. Diese Bestandteile arbeiten eng mit externen Systemen wie Betriebssystem, Grafiktreiber (z. B. OpenGL oder DirectX) und dem Server zusammen, der die Synchronisation virtueller Welten bei Mehrspielerumgebungen übernimmt (Lewis & Jacobson 2002, S.29).

Die ältere Fachliteratur beschreibt Game Engines eher als eine Sammlung von unterstützenden Entwicklungswerkzeugen. Dazu zählen beispielsweise ein *Compiler*, ein *Debugger*, ein *Integrated Development Environment (IDE)*, ein *Software Development Kit (SDK)* sowie eine oder mehrere *Application Programming Interfaces (APIs)*. Diese Komponenten unterstützen Entwickler bei der Erstellung und Fehlerbehebung von Spielen, stellen aber nicht direkt die Engine selbst dar (Novak 2012: 342–343).

Zudem ist es wichtig zu beachten, dass es nicht die eine einheitliche Game Engine gibt. Der Aufbau und Funktionsumfang variieren je nach Einsatzzweck – etwa für 2D-Spiele, mobile Anwendungen oder komplexe 3D-Welten. Dies bedeutet, dass jede Engine individuell zusammengestellt ist und unterschiedliche Module und Werkzeuge mitbringt (Peter, 2023).

2.2.2 Heutige Game Engens

Für die Entwicklung eines Spiels ist die Wahl der passenden Game Engine ein entscheidender Schritt. Grundsätzlich gibt es zwei Herangehensweisen: Entweder man entwickelt eine eigene Engine – was den Vorteil bietet, dass nur genau die Funktionen implementiert werden, die tatsächlich benötigt werden – oder man greift auf eine bestehende Game Engine zurück. Eine eigene Engine zu entwickeln, erfordert jedoch sehr tiefgehende Programmierkenntnisse, viel Zeit und Ressourcen. Gerade

für Einzelentwickler oder kleine Teams ist dies in der Regel nicht praktikabel (Peter 2023).

Daher greifen die meisten Entwickler auf etablierte Engines zurück. Die bekanntesten Vertreter auf dem Markt sind Unreal Engine und Unity. Beide bieten umfangreiche Möglichkeiten für professionelle Spielentwicklung und werden auch in der Industrie eingesetzt. So wurde beispielsweise das bekannte Battle Royal Spiel *Fortnite* mit der Unreal Engine entwickelt. Unreal ist besonders für seine hochwertigen 3D-Grafiken und seine leistungsfähige Rendering-Technologie bekannt. Allerdings gilt sie auch als weniger anfängerfreundlich und komplexer im Einstieg, was insbesondere für kleinere Bildungsprojekte oder Einsteiger*innen eine Hürde darstellen kann. (vgl. animations-and-more.com; unrealengine.com)

Auch Unity ist weit verbreitet und unterstützt sowohl 2D- als auch 3D-Entwicklung. Die Engine verfügt über eine große Auswahl an Assets und Plugins. Zwar ist Unity in der Basisversion kostenlos, jedoch wurde in den letzten Jahren zunehmend Kritik an der Lizenzpolitik laut. Für bestimmte Umsatz- oder Installationszahlen fallen teils erhebliche Gebühren an, was besonders für Entwickler mit geringem Budget problematisch sein kann. (vgl. animations-and-more.com; unity.com)

Vor diesem Hintergrund fällt die Wahl auf Godot, eine moderne, quelloffene und einsteigerfreundliche Engine, die sich zunehmend als Alternative zu den großen Marktführern etabliert. Weitere Details zur Godot-Engine finden sich im Kapitel *Softwaretools und Entwicklungsumgebungen*.

2.3 Navigation, Orientierung & Interaktion in virtuellen Umgebungen

2.3.1 Orientierung in Virtuellen 3D-Welten

Die Fähigkeit, sich in einer Umgebung zurechtzufinden, wird in der Forschung unter dem Begriff „Wayfinding“ zusammengefasst, welcher als räumliches Problemlösen definiert wird. Dabei handelt es sich um einen komplexen Prozess, der sowohl kognitive als auch wahrnehmungsbasierte Fähigkeiten umfasst (Arthur & Passini, 1992, S. 25). Menschen orientieren sich grundsätzlich, indem sie mentale Repräsentationen ihrer Umgebung – sogenannte kognitive Karten – erstellen. Diese entstehen nicht durch die Wahrnehmung eines einzigen Blickwinkels, sondern durch

die Integration verschiedener Perspektiven. Das menschliche Gehirn setzt somit einzelne visuelle Eindrücke zu einem Gesamtbild zusammen (ebd., S. 23).

Wayfinding besteht aus drei miteinander verknüpften Teilprozessen: der Entscheidungsfindung und Handlungsplanung, der Ausführung dieser Entscheidungen im Raum sowie der Informationsverarbeitung, die sowohl Umweltwahrnehmung als auch kognitive Interpretation beinhaltet (ebd., S. 25). Besonders in unbekannten oder komplexen Umgebungen kann dieser Prozess mit Unsicherheit behaftet sein – beispielsweise, wenn visuelle Informationen fehlen oder uneindeutig sind. In solchen Situationen wird die Fähigkeit zur flexiblen Problemlösung entscheidend (ebd., S. 28).

Für die Orientierung greifen Menschen auf verschiedene Strategien zurück. Dazu gehört unter anderem die Nutzung markanter Orientierungspunkte wie Landmarken oder Lichtführungen, die als visuelle Anker dienen und die räumliche Einordnung erleichtern (ebd., S. 37). Zusätzlich unterscheiden Forscher zwischen zwei Arten von mentalen Karten: der egozentrischen Route (eine Abfolge von Wegpunkten mit Richtungsänderungen) und der allozentrischen Survey-Karte (eine abstrakte, koordinatenbasierte Darstellung von Raumbeziehungen). In unbekannten Umgebungen beginnen Menschen typischerweise mit dem Erfassen von Landmarken, um darauf aufbauend weitere Wege zu erschließen (ebd.).

Im Kontext von 3D-Welten – wie sie in Computerspielen oder virtuellen Simulationen vorkommen – gestaltet sich dieser Prozess besonders herausfordernd. Da der Raum digital generiert und oftmals weitläufig oder komplex ist, ist eine gezielte Informationsführung durch Designentscheidungen von großer Bedeutung. Visuelle Elemente wie Lichtquellen, Farbkontraste oder eindeutige Strukturen unterstützen die kognitive Orientierung der Spielenden. Fehlen diese Hinweise, kann es zu Desorientierung oder kognitiver Überlastung kommen, besonders wenn Nutzer aktiv nach Informationen suchen, diese aber nicht eindeutig oder auffindbar sind (ebd., S. 34).

Spieler entwickeln im Laufe des Spiels eigene Strategien, um sich in 3D-Räumen zurechtzufinden. Dazu gehört etwa das Merken bestimmter Objekte oder das Verfolgen auffälliger Wegeführungen. Die Art und Weise, wie ein Spiel visuelle

Informationen strukturiert, beeinflusst somit direkt das Navigationsverhalten der Spielenden.

2.3.2 Benutzerführung / User Experience in 3D-Umgebungen

Die Benutzerführung in 3D-Spielumgebungen ist ein zentrales Element des Game Designs – sie entscheidet maßgeblich darüber, wie leicht Spieler ins Spiel finden und wie motiviert sie bleiben. Ziel ist es, die Spielenden so zu leiten, dass sie sich stets orientieren können, ohne das Gefühl zu bekommen, bevormundet oder überfordert zu sein. Gerade in offenen 3D-Welten, die durch Komplexität und Freiheit geprägt sind, ist die Herausforderung groß, eine Balance zwischen Freiheit und Zielorientierung herzustellen.

Eine zentrale Regel im Spieldesign besagt: Ein Spiel muss schnell und intuitiv zugänglich sein. Lange Erklärungen oder komplexe Einstiege wirken abschreckend und gefährden den Spielfluss (vgl. Salen & Zimmerman 2003, zitiert in Kerres et al. 2008, S.4). Stattdessen sollte Lernen idealerweise innerhalb des Spiels selbst stattfinden – durch aktives Ausprobieren, Entdecken und Interagieren. Diese Form des impliziten Lernens basiert auf dem Prinzip, das Wissen und Handlungskompetenzen durch wiederkehrende Erfahrungen aufgebaut werden. Lernprozesse werden so in den Spielfluss eingebettet, dass sie vom Spielgeschehen selbst getragen und gefestigt werden, ohne dass sie den Spielspaß unterbrechen.

Um solche Prozesse zu ermöglichen, sollte das Spiel durch gestalterische Hinweise führen, anstatt mit erklärenden Elementen zu unterbrechen. Michael Kerres et al. (Kerres, 2008, S. 5) sprechen in diesem Zusammenhang vom didaktisch-immersiven Spieldesign: Dabei sollen Spieler durch die Umgebung selbst lernen – etwa durch visuell auffällige Objekte, subtile Hinweise durch NPCs -Non-Player-Character, die vom Computer und nicht von einem menschlichen Spieler gesteuert werden - oder Levelstrukturen, die sie schrittweise an komplexere Aufgaben heranführen. Ziel ist es, den Wechsel in den expliziten Lernmodus (Novak, 2012, S. 10) – etwa durch Popup-Tutorials – möglichst zu vermeiden, um die Immersion zu erhalten.

Ein weiterer entscheidender Aspekt der User Experience in 3D-Umgebungen ist die Usability des Interfaces. Laut Novak (Novak, 2012, S.259-261) steht dabei nicht

Ästhetik im Vordergrund, sondern Funktionalität. Ein Interface muss vor allem eines ermöglichen: das Spiel zu spielen. Eine Benutzeroberfläche gilt als schlecht, wenn sie kryptisch, überladen, inkonsistent oder ineffizient ist. Besonders in 3D-Welten, in denen Orientierung ohnehin eine Herausforderung darstellt, kann ein unklar gestaltetes Interface zu Frust und Verwirrung führen.

Ein gutes Beispiel für gelungene Integration von Benutzerführung in die Spielwelt ist *Call of Cthulhu: Dark Corners of the Earth*, das auf klassische HUD-Elemente weitgehend verzichtet und stattdessen auf Audio- und visuelle Signale im Spielgeschehen setzt, um etwa den Gesundheitszustand zu kommunizieren (MacLean, 2010, S. 263). HUD steht für Head-up-Display und ist eine Form der Informationsübermittlung für den Spieler in der Benutzeroberfläche. Durch den Verzicht bleibt die Immersion erhalten(Novak, 2012, S.248), während die notwendigen Informationen dennoch vermittelt werden. Im Gegensatz dazu steht ein Spiel wie *EVE Online*, das eine Vielzahl an Steuerungselementen und Informationsfenstern nutzt. Zwar ist die Usability für erfahrene Nutzer hoch, die Einstiegshürde aber ebenso.

Insgesamt zeigt sich: Gute Benutzerführung in 3D-Welten basiert auf einem durchdachten Zusammenspiel von Interface-Design, Raumgestaltung und adaptivem Schwierigkeitsgrad. Spieler sollen durch subtile Hinweise gelenkt werden, dabei aber das Gefühl von Kontrolle und Entdeckungsfreiheit behalten. Zentral ist dabei die Fähigkeit des Spiels, kontextbezogenes, prozedurales Wissen zu fördern – ohne den Spielfluss zu stören. Eine solche User Experience verlangt nach Design, das technisches Verständnis und spielerpsychologisches Feingefühl miteinander verbindet.

2.3.3 Bewegungsmöglichkeiten

Digitale Spiele bieten eine Vielzahl von Bewegungsformen, die je nach Spielwelt, Genre und Designziel variieren. Die Spielfigur kann sich zu Fuß fortbewegen, laufen, springen oder klettern. In bestimmten Szenarien sind auch Bewegungen mit Fahrzeugen möglich – etwa auf Schienen, auf der Straße oder im Wasser. Darüber hinaus ermöglichen viele Spiele auch dreidimensionale Fortbewegung durch Fliegen oder Schweben. Diese Vielfalt an Bewegungsoptionen eröffnet nicht nur spielerische

Abwechslung, sondern kann auch genutzt werden, um bestimmte Inhalte oder Herausforderungen gezielt zu gestalten (vgl. Novak, 2012).

Freie Bewegung - open Worlds

Spieler können den Raum ohne vorgegebene Routen in drei Dimensionen erkunden (häufig mit Springen, Fliegen oder Klettern).

- Open-World-Spiele (z. B. *The Legend of Zelda: Breath of the Wild*, *Minecraft*)

(vgl. buecherstadtmagazin.de)

Beschränkte Bewegung - flat or side view

Bewegung wird auf bestimmte Achsen, Pfade oder Rasterfelder begrenzt; erleichtert Level-Design, kann aber Exploration einschränken.

- Side-Scroller / 2,5-D-Plattformer (z. B. *Super Mario Bros.*, *Ori and the Blind Forest*) → Bewegung im Wesentlichen nur nach links/rechts und beschränkt vertikal (Novak 2012: 225)

2.3.4 Kamera perspektiven (PoV – Point of View)

Der Begriff „Point of View“ beschreibt kurz gesagt den Blickwinkel, aus dem man als Spieler die Welt im Spiel sieht. Dabei unterscheidet man hauptsächlich zwischen zwei Perspektiven: First Person und Third Person

Fist-Person

In der First-Person-Perspektive (First Person Point of View) sieht man die Welt aus der Sicht des spielbaren Charakters – so, als würde man direkt durch dessen Augen blicken. Die eigene Spielfigur ist dabei nicht sichtbar, höchstens Teile der Hände oder der Arme. Diese Sichtweise fördert die Immersion, da sich die Spielenden stärker mit dem Charakter identifizieren und „in ihn hineindenken“ können. Gleichzeitig erschwert sie jedoch die Wahrnehmung der Figur selbst, da keine direkte Sicht auf das Aussehen oder die Körperhaltung des Charakters besteht. (Novak 2012, S. 165)

Third-Person

Bei der Third-Person-Perspektive sieht man den spielbaren Charakter vollständig auf dem Bildschirm. Man beobachtet ihn sozusagen aus einem gewissen Abstand – meist von hinten und leicht oberhalb. Diese Sichtweise ermöglicht es, sich ein besseres Bild vom Charakter zu machen. Gleichzeitig fällt es oft schwerer, sich emotional in die Figur hineinzuversetzen. Deshalb ist es wichtig, dass sich der Charakter deutlich von der Außenwelt abhebt und visuell gut erkennbar ist.(Novak 2012, S. 165)

Steuerung

Die Art, wie sich Spielende in einer virtuellen Umgebung bewegen, hängt stark vom eingesetzten Eingabegerät ab. Je nach Plattform – PC, Konsole oder mobiles Gerät – kommen unterschiedliche Steuerungsmethoden wie Maus und Tastatur, Gamepads oder Touchscreens zum Einsatz. Diese Eingabeformen beeinflussen maßgeblich das Spielerlebnis und die Bewegungsfreiheit innerhalb des Spiels (vgl. Novak, 2012 S. 242).

2.4 Praxisbeispiele

2.4.1 Game Engins mit Geodaten

Science City Itzling

In diesem Projekt wurde ein virtueller, spielerisch erkundbarer Campus auf Basis realer Geodaten in einer 3D-Spielwelt (Unreal Engine) erstellt. Dabei kamen verschiedenste Datentypen zum Einsatz: 3D-Meshes von Drohnenbefliegungen, detaillierte Gebäudemodelle, einfache Blockmodelle sowie 360°-Panoramaaufnahmen. Ziel war es, einen teilautomatisierten, wiederverwendbaren Workflow zur Integration dieser Datenarten in die Spieleumgebung zu entwickeln.

Die Umsetzung zeigt, dass der technologische Fortschritt neue Möglichkeiten zur realitätsnahen Visualisierung eröffnet, gleichzeitig jedoch auch große Herausforderungen bei der Datenintegration mit sich bringt. Standardisierte Workflows oder klare Richtlinien bzw. Leitfäden fehlen häufig noch. Im Rahmen des Projekts wurden daher verschiedene Integrationsmethoden getestet, etwa für CAD-basierte Innenraummodelle oder Gebäudedaten aus BIM/GIS-Systemen. Das Ergebnis ist ein

Prototyp in Egoperspektive, der bereits zahlreiche Datenarten kombiniert und künftig noch erweitert werden soll – etwa um Livedaten oder Unterstützung für Mobilgeräte und VR-Brillen.

Fazit:
Das Projekt demonstriert anschaulich das Potenzial und die Herausforderungen der Geodatenintegration in Game Engines. Besonders deutlich wird: Unterschiedliche Datentypen erfordern flexible, teilweise individuelle Lösungen. Gleichzeitig zeigt der entwickelte Workflow eine wichtige Perspektive auf, wie reale Geodaten erfolgreich spielerisch erlebbar gemacht werden können (Caroline Atzl, 2023).

2.4.2 Serius Games mit Geodaten Bezug

An der TU Darmstadt wird im Rahmen der AG Serious Games ein digitaler Zwilling des Universitätscampus entwickelt, der auf Geodaten basiert und vielfältige Einsatzmöglichkeiten bietet – von der Lehre über die Forschung bis hin zum Marketing. Das Projekt wird kontinuierlich im Rahmen von Lehrveranstaltungen und Abschlussarbeiten erweitert und ist somit ein wachsendes Beispiel für den Einsatz von Geodaten in Serious Games.

Ein zentraler Aspekt des Projekts ist die Integration verschiedener Geodatenquellen (u. a. amtliche Daten des HLBG, OpenStreetMap und Daten aus dem TU Campus-Navigator) mithilfe von GIS-Software wie ArcGIS und CityEngine in die Unity-Engine. So entstand eine realitätsnahe, interaktive Umgebung. Ergänzend wurden 360-Grad-Bilder an Schlüsselstellen des Campus aufgenommen und eingebettet, um die Immersion zu erhöhen und die Verbindung zur realen Umgebung zu stärken.

Ein weiterer Projektstrang beschäftigte sich mit der Erstellung hochdetaillierter 3D-Modelle einzelner Gebäude durch UAV-basierte Photogrammetrie. Dabei wurde z. B. das Gebäude L5|01 mit 16K-Texturen und hoher Genauigkeit digitalisiert. Diese Daten wurden in einer spielbaren Demo in Unreal Engine 5 umgesetzt, die einen ersten Eindruck des digitalen Zwillings in interaktiver Form ermöglicht.

Fazit:
Das Projekt der TU Darmstadt zeigt beispielhaft, wie Geodaten vielfältig in Spielumgebungen eingebunden werden können. Es kombiniert unterschiedliche

Datenarten, Game Engines (Unity und Unreal) sowie Visualisierungsansätze (360°-Bilder, Photogrammetrie), um ein immersives, interaktives und stetig wachsendes Lern- und Erkundungserlebnis zu schaffen.

(vgl. etit.tu-darmstadt.de)

3 Methodik & technische Umsetzung

3.1 Grundlagen des Gebiets

Das im Spiel abgebildete Gebiet liegt in Niederbayern, südwestlich von Regensburg, mit dem geographischen Schwerpunkt auf der Stadt Kelheim. Kelheim ist eine mittelgroße Stadt mit rund 16.842 Einwohnern (Stand: 31.12.2019) und erstreckt sich über eine Fläche von etwa 76,76 km². Die Stadt liegt an einer markanten Flussgabelung, an der sich die Donau mit dem kleineren Fluss Altmühl vereinigt – letzterer ist durch den Main-Donau-Kanal schiffbar gemacht und stellt eine bedeutende Wasserstraße dar. Aufgrund dieser Lage spielt der Hafen Kelheim eine wichtige wirtschaftliche Rolle, besonders für den Güterumschlag und die Binnenschifffahrt.(vgl. kelheim.de)

Aus historischer Sicht besitzt die Region eine lange Besiedlungstradition, die bis in die Kelten- und Römerzeit zurückreicht. Kelheim war aufgrund seiner strategischen Lage an der Donau und am Eingang zum Altmühltal schon früh von Bedeutung.

Eine besondere landschaftliche und kulturelle Dominante bildet die Befreiungshalle, ein nationales Denkmal des 19. Jahrhunderts. Sie thront auf dem Michelsberg oberhalb der Stadt und markiert den Punkt, an dem die Altmühl in die Donau mündet. Ihre exponierte Lage macht sie nicht nur zu einem historischen Wahrzeichen, sondern auch zu einem idealen Orientierungspunkt im Spiel.(vgl. befreiungshalle.org; schloesser.bayern.de)

Ein weiteres markantes Merkmal in der Umgebung ist der Kalksteinbruch süd-östlich Kehlheims. Er ist geologisch und landschaftlich prägend und diente früher als Materialquelle für bedeutende Bauwerke der Region, unter anderem auch für die Befreiungshalle selbst. Der Steinbruch veranschaulicht die enge Verbindung zwischen natürlicher Ressource und kultureller Nutzung (vgl. fels.de).

Geomorphologisch ist das Gebiet durch das Donautal und das Altmühltal geprägt. Diese Täler wurden im Laufe der Erdgeschichte durch Flusserosion und tektonische Vorgänge geformt. Die steilen Hänge, markanten Felsformationen und das Relief der Mittelgebirgslandschaft machen das Gebiet besonders charakteristisch und visuell reizvoll für die Darstellung in einem 3D-Spiel (vgl. hafen-kelheim.de).

3.2 Daten

Für das Projekt wurden verschiedene Geodaten vom Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) bereitgestellt. Die Daten liegen in unterschiedlichen Formaten und Auflösungen vor und bilden die Grundlage für die visuelle Darstellung sowie die spätere Einbindung in die Spielumgebung. Alle Daten verwenden das Koordinatensystem UTM32 (EPSG:25832).

Folgende Datensätze wurden zur Verfügung gestellt:

- Digitales Geländemodell (DGM)
  Auflösung: 10 m, Stand: 2022, Format: ASC
- 3D-Gebäudemodelle
  - LoD1: Gebäudeumrisse aus dem Jahr 1817, Format: SHP
  - LoD2: Gebäude aus dem Jahr 2022, Format: SHP
- Historische Karten im Format GeoTIFF
  - Uraufnahme von 1817
  - Positionskarte 1870: Bodenauflösung ca. 2 m
  - TK25: Topographische Karte 1:25.000 aus dem Jahr 2024
- Orthophotos (DOPs) im Format GeoTIFF
  - DOP 1945: Bodenauflösung 40cm
  - DOP 1982 Bodenauflösung 40 cm
  - DOP 2024: Bodenauflösung 40 cm

Die konkrete Bearbeitung und Einbindung der Daten erfolgt in den folgenden Kapiteln.

3.3 Software-Tools & Entwicklungsumgebung

3.3.1 Adobe Photoschop

Adobe Photoshop ist eine professionelle Bildbearbeitungssoftware, die vielfältige Möglichkeiten zur Bearbeitung und Gestaltung von Bildmaterial bietet. Im Rahmen dieser Arbeit wurde Photoshop primär für zwei Anwendungsbereiche genutzt: zur Aufbereitung von Geodaten sowie zur Erstellung visueller Elemente für das Spiel.

(vgl. adobe.com)

Zum Einsatz kam dabei insbesondere das Plug-in Geographic Imager, das die Bearbeitung georeferenzierter Daten direkt in Photoshop ermöglicht. Damit lassen sich geodätische Informationen analog zu einem GIS-System verarbeiten, während gleichzeitig die typischen Vorteile von Photoshop – insbesondere hinsichtlich der Bildverbesserung – genutzt werden können.

Die Anwendungsschwerpunkte lagen auf der Zusammenführung unterschiedlicher Geodatensätze, der Anpassung spezifischer Kartenausschnitte sowie der radiometrischen Korrektur. Darüber hinaus ermöglicht das Plug-in auch die Umrechnung von Auflösungen und den Wechsel zwischen verschiedenen Koordinatensystemen. Gängige Formate wie TIFF, GeoTIFF für Rasterdaten und ASC für 3D-Daten konnten problemlos importiert und exportiert werden.

Ergänzend kamen auch klassische Photoshop-Funktionen zum Einsatz, beispielsweise zur Freistellung von Bildelementen oder zur Erstellung eigener Grafiken mithilfe der integrierten Generierungsfunktionen.
Eine detaillierte Beschreibung der Arbeitsschritte erfolgt im Abschnitt zum Workflow.

Abbildung 2: In Photoshop generiertes Bild

3.3.2 Infraworks

Autodesk InfraWorks® ist eine CAD-Software für den konzeptionellen Entwurf und die kontextbezogene 3D-Geodaten-Modellierung von Infrastrukturprojekten. Sie dient in erster Linie der Integration und Visualisierung großer Geodatenmengen, aus denen realitätsnahe 3D-Kontextmodelle erstellt werden können. Typische Einsatzgebiete sind die Planung und Darstellung von Straßen-, Schienen- und Wasserbauprojekten sowie städtebaulichen Vorhaben (vgl. autodesk.com).

Ein weiteres zentrales Anwendungsfeld ist die Durchführung von Verkehrsanalysen und Sichtbarkeitsstudien, um die Auswirkungen geplanter baulicher Veränderungen besser abschätzen und visualisieren zu können.

Im Rahmen dieser Arbeit wurde InfraWorks als vorbereitendes Werkzeug verwendet, um sämtliche Rohdaten zusammenzuführen, diese auf einen definierten Kartenausschnitt zu begrenzen und anschließend kompakt zu exportieren. Der Import der Geodaten erfolgte dabei in Formaten wie ASC, SHP oder GeoTIFF, während der Export in das weiterverarbeitbare FBX-Format durchgeführt wurde. Detaillierte Informationen zu den einzelnen Verarbeitungsschritten finden sich im Abschnitt zum Workflow

3.3.3 Blender

Blender ist eine frei verfügbare Open-Source-Software zur Erstellung und Bearbeitung von 3D-Grafiken. Die Software wurde 1994 vom niederländischen Softwareentwickler Ton Roosendaal ins Leben gerufen. Zu den grundlegenden Funktionen von Blender zählen unter anderem die Modellierung, das Rigging, die Animation, Simulation, das Rendering, Compositing sowie die Erstellung von Spiele-Assets. (vgl. blender.org)

Im Rahmen dieser Arbeit wurde Blender hauptsächlich zur Weiterverarbeitung und Optimierung von 3D-Modellen sowie zur Erstellung spielrelevanter Charakterdesigns und Animationen verwendet. Der Daten-Import von Daten aus Infraworks erfolgte über das FBX-Format in Blender, während der Export in das für die Game Engine Godot geeignete Format GLTF/GLB 2.0 durchgeführt wurde. Weitere Details zu diesem Prozess sind im Abschnitt Workflow dokumentiert.

In Blender wurden alle zentralen Spielcharaktere modelliert, darunter das UFO, die fünf kleinen Aliens sowie der Traktorstrahl (Beam). Die entsprechenden Animationen – sowohl Standard- als auch individuelle – wurden ebenfalls in Blender erstellt. Jeder Alien-Charakter erhielt eine einheitliche Standardanimation („Default“-Animation), in diesem Fall eine kontinuierliche Laufbewegung. Darüber hinaus wurden für jedes Alien individuelle Einsauganimationen entworfen, um den Einsammelprozess visuell abwechslungsreich zu gestalten.

Auch alle notwendigen Collision Shapes (Kollisionskörper) zur physikalischen Interaktion im Spiel wurden in Blender modelliert und vorbereitet. Eine detaillierte Beschreibung dieses Vorgangs findet sich ebenfalls im Abschnitt zum Workflow.

3.3.4 Godot

Godot ist eine vergleichsweise junge Game Engine, die am 23. August 2022 in den Niederlanden veröffentlicht wurde. Insbesondere durch erfolgreiche Veröffentlichungen im Bereich 2D-Spiele gewann die Engine zunehmend an Bekanntheit. Einen deutlichen Zuwachs an Aufmerksamkeit erfuhr Godot, als der Mitbewerber Unity Änderungen am Preismodell einführte, wodurch viele Entwickler:innen nach alternativen Lösungen suchten.

Godot ist vollständig Open Source und wird durch eine stetig wachsende Community weiterentwickelt. Die Engine bietet eine integrierte Asset Library und erhält regelmäßige Updates, die den Funktionsumfang kontinuierlich erweitern. Ziel des Projekts ist es – ähnlich wie bei Blender – eine frei zugängliche, leistungsfähige und stetig verbesserbare Game Engine bereitzustellen.

Obwohl Godot in bestimmten Bereichen, wie z. B. beim Shading oder Rendering, derzeit noch nicht mit etablierten Engines wie Unity oder Unreal Engine mithalten kann, bietet sie bereits ein breites Spektrum an Funktionen. Besonders im 3D-Bereich sind in den letzten Jahren erhebliche Fortschritte zu verzeichnen. Auch Virtual Reality (VR) wird mittlerweile unterstützt, wodurch sich Godot zunehmend auch für komplexere Projekte eignet.

Ein großer Vorteil von Godot liegt in der benutzerfreundlichen Oberfläche und der einfachen Bedienung. Diese machen die Engine besonders attraktiv für Einsteiger und Umsteiger. Zusätzlich existiert eine umfangreiche Online-Dokumentation sowie eine Vielzahl an Tutorials und Forenbeiträgen, die den Einstieg deutlich erleichtern.

In dieser Arbeit übernimmt Godot die zentrale Rolle der Spielentwicklung. Als technisches Herzstück fungiert die Engine als Plattform, in der alle zuvor erstellten Elemente – darunter die aufbereiteten Geodaten, 3D-Modelle, Animationen und Interaktionslogiken – zusammengeführt werden, um das finale Serious Game zu realisieren.

(vgl. godot.foundation; https://godotengine.org/)

3.3.5 Adobe Character Animator

Adobe Character Animator ist eine Software, die es ermöglicht, fiktiven Figuren Leben einzuhauchen, indem sie mit Stimmen, Mimik und Gestik animiert werden. In diesem Projekt wurde die Software genutzt, um dem Spieler Informationen auf eine ansprechende und interaktive Weise zu vermitteln.(vgl. adobe.com) Zum einen wurde sie im Tutorial verwendet, um den Ablauf und die Steuerung des Spiels zu erklären, und zum anderen, um Informationen über die jeweiligen Einsammelorte oder Informationen der Karten der Aliens zu geben.

Um eine Animation in Character Animator zu erstellen, benötigt man mehrere Bausteine. Zunächst ist ein Hintergrund erforderlich, da die Figuren nicht ohne weiteres exportiert werden kann. Für dieses Projekt wurde eine vorgefertigte Figur aus der Adobe- Figur -Sammlung verwendet. Der nächste Schritt ist besteht darin, eine Audiodatei im MP3-Format zu importieren, um der Figur Wort in den Mund zu legen. Mit der Funktion der Lippensynchronisation in Character Animator wird der Figur die Mundbewegungen synchron zum Audio angepass. Anschließend können vorgefertigte Gesten hinzugefügt werden, um der Figur mehr Ausdruck zu verleihen und sie lebendiger wirken zu lassen.

Für das Tutorial wurde zusätzlich eine kleine Animation erstellt, um die Steuerung des Ufos zu veranschaulichen. Diese Animation besteht aus PNG-Dateien, die importiert und dann in Character Animator bearbeitet wurden. Nach der Fertigstellung wurde das Video im MP4-Format exportiert. Da Godot jedoch OGG als bevorzugtes Format für Videoimporte benötigt, wurden die MP4-Dateien anschließend in das OGG-Format konvertiert.

Die Einbindung dieser Videos in Godot ist relativ einfach. Hierfür gibt es einen speziellen Node namens „VideoStreamPlayer“. In diesem Node kann jedes beliebige Video eingefügt werden. Per Code wird festgelegt, wann das Video gestartet werden soll.

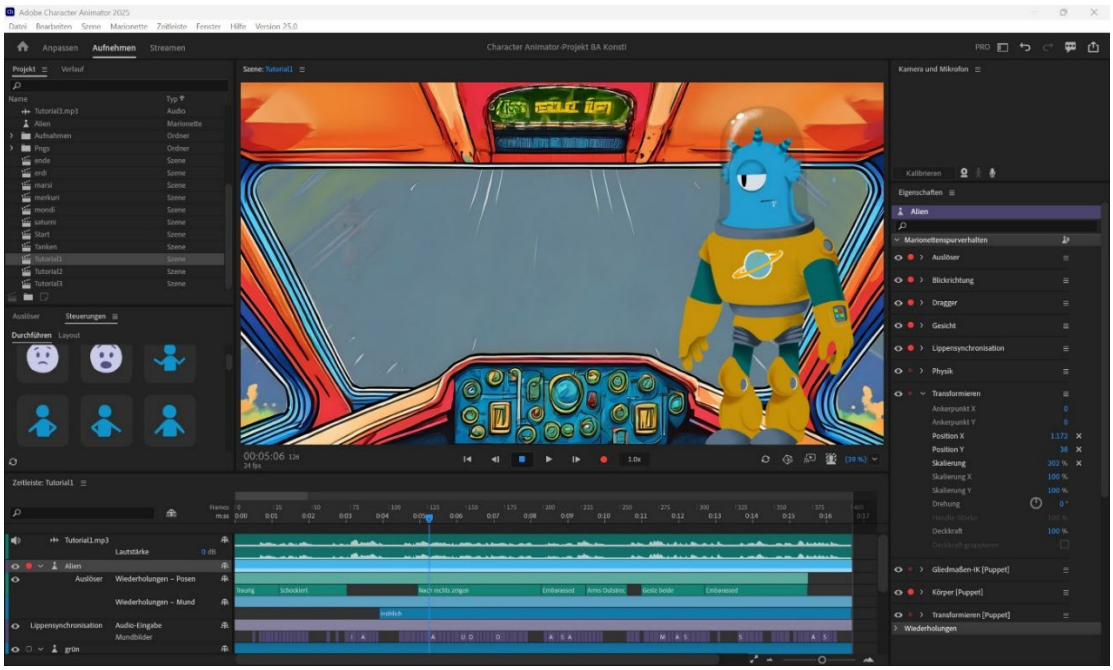


Abbildung 3: Figur in Adobe Character Animator

3.4 Workflow der Entwicklung

Bestellung der Daten

Die Geodaten werden über ein LDBV-internes Programm mit dem Namen „Geodaten-Bestellung“ ausgespielt; externe Nutzerinnen und Nutzer haben keinen Zugriff darauf. Über diese Software ist es möglich, gezielt verschiedene Geodaten von Raster- bis hin zu Vektordaten zu bestellen. Im Programm kann unteranderem ein rechteckiger Bereich auf der Karte aufgezogen werden; alle innerhalb dieses Bereichs liegenden Daten werden für die Bestellung ausgewählt . Auf diese Weise lassen sich auch größere Datenmengen effizient und gebündelt anfordern. Zudem bietet das Programm die Möglichkeit, Rasterdaten zu einem Mosaik zusammenzufassen und die Bodenpixelgröße auszuwählen, die letztendlich auch die Datenmenge beeinflusst. Das reduziert einerseits die Anzahl der Geodaten insgesamt und erleichtert den Import der Geodaten in eine Software.

Die verfügbaren Formate variieren je nach Datentyp: Für das Digitale Geländemodell (DGM) steht das Format ASCII Grid (ASC), für die Digitalen Orthophotos (DOP) das GeoTIFF-Dateien zur Verfügung, während die 3D-Gebäudemodelle der Stufe LOD2 im Shapefile-Format (SHP) bereitgestellt werden.

Photoshop

Nach der Bereitstellung der Geodaten beginnt die erste Bearbeitungsphase in Adobe Photoshop. Der Import der DOPs (Digitale Orthophotos) erfolgt im TIFF-Format, das DGM (Digitale Geländemodell) wird im ASC-Format eingebunden. Da die Rohdaten in der Regel aus mehreren Kacheln bestehen, müssen diese zunächst mit dem Plug-in Geographic Imager zusammengefügt werden. Dafür wird die Funktion *Mosaic* verwendet, die es ermöglicht, mehrere Kacheln zu einer durchgängigen Datei zu kombinieren.

Anschließend wird der gewünschte Kartenausschnitt herausgeschnitten. Dazu wird über die große, noch nicht zugeschnittene Kachel eine kleinere Referenzkachel gelegt, die den finalen Ausschnitt vorgibt. Mit dem Zauberstab-Werkzeug werden dann alle Bildbereiche markiert, die außerhalb dieses Ausschnitts liegen, und entfernt. Diese Schritte werden für alle Kartenarten durchgeführt – je nach Datensatz sind es mal mehr, mal weniger Kacheln.

Es folgt eine radiometrische Verbesserung. Hierbei werden Farbwerte, Helligkeit und Kontraste angepasst, um eine möglichst stimmige und visuell ansprechende Darstellung zu erreichen. Bei historischen Karten – etwa der Uraufnahme oder beim Positionsblatt – erfolgt zuerst die Farbkorrektur und anschließend das Zusammenfügen der Einzelkacheln. Der Grund dafür ist, dass diese alten Karten oft wie ein Flickenteppich wirken – durch die farbliche Angleichung wirken sie harmonischer zueinander.

In einigen Fällen muss auch das Koordinatensystem angepasst werden, was sich ebenfalls mit dem Geographic Imager unkompliziert umsetzen lässt. Die Bodenauflösung des DGM wurde in dieser Arbeit auf 10 Meter festgelegt, um eine gute Balance zwischen Detailgenauigkeit und Performance in der Game Engine zu gewährleisten. Das verwendete DGM stammt aus dem Jahr 2022.

Abschließend wurden die bearbeiteten DOPs mit neuem Namen im GeoTIFF— und das DGM im ASC-Format exportiert.

Aufbereitung in InfraWorks

Nach der Bearbeitung in Photoshop erfolgt die Weiterverarbeitung in Autodesk InfraWorks und wechseln somit in ein CAD-System, um Geodaten dreidimensional zusammenfügen. Zunächst wird ein neues Projekt angelegt, das auf dem zuvor exportierten Digitalen Geländemodell (DGM) basiert. Auf dieses Modell werden die verschiedenen Kartenmaterialien wie das aktuelle DOP, die historische Uhraufnahme, die Topographische Karte (TK25) sowie der Planungskarte aufgesetzt.

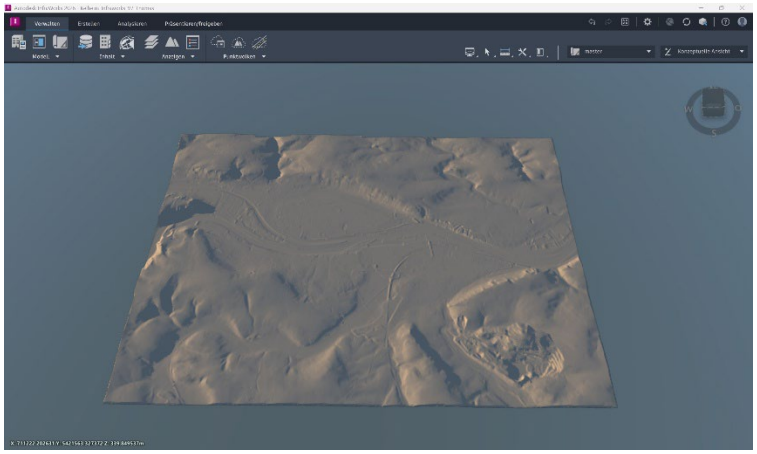


Abbildung 4: Digitales Geländemodell

In den Haupteinstellungen wird ein einheitlicher Gebietseingrenzung definiert. Dieser wurde so gewählt, dass die gesamte Stadt Kelheim sowie umliegende Besonderheiten der Region vollständig abgedeckt sind. Der Vorteil dieser Begrenzung liegt in der einheitlichen Größe aller erzeugten Modelle, was die spätere Verarbeitung erheblich vereinfacht.

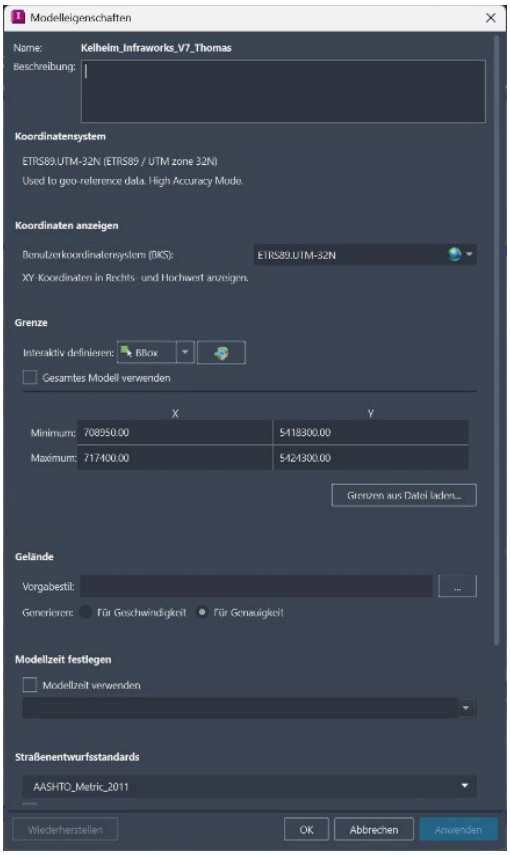


Abbildung 5: Projekteigenschaften in InfraWorks

Für das aktuelle DOP werden zusätzlich automatisch erkannte Objekte wie Gewässer, Straßen und Bahnlinien hinzugefügt. Bei den älteren Kartenaufnahmen wird bewusst auf diese automatisierten Ergänzungen verzichtet, um die historische Authentizität zu wahren.

Ein weiterer Schritt ist die Integration von 3D-Gebäudemodellen (LOD).

- Dem aktuellen DOP und der aktuellen TK25 werden LOD2-Gebäude hinzugefügt, die detaillierte und standardisierte Dachformen enthalten und die 3D-Welt ideal ergänzen bzw. erweitern.
- Das DOP von 1945 und eine weitere historische Aufnahme erhalten keine LOD2-Gebäude, da hierfür keine Datenlage vorhanden ist.

- Die Uraufnahme und das Positionsblatt werden mit LOD1-Gebäuden versehen – einfachen, quaderförmigen Baukörpern ohne Dachstruktur. Die Gebäudeumrisse wurde auf der Grundlage der Uraufnahme digitalisiert und in Infraworks mit einer Standardhöhe von 10m eingelesen.

Im Anschluss erfolgt der Export jedes Kartenmodells einzeln im FBX-Format. In den Exporteinstellungen wird darauf geachtet, dass verschiedene Bestandteile wie das DGM, die Texturen (z. B. DOP), die LOD-Modelle und die Gewässer separat gespeichert werden.

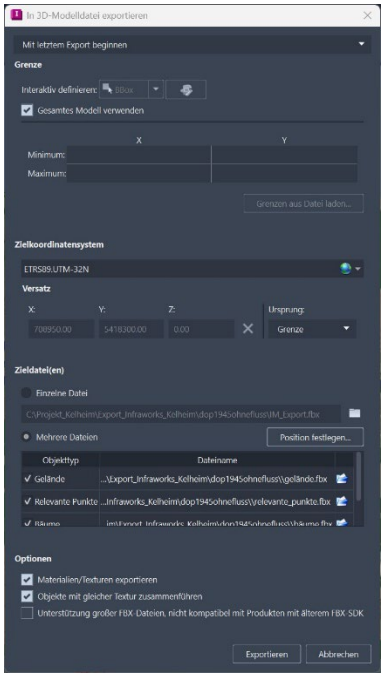


Abbildung 6: Exportfenster in InfaWorks

Das bedeutet:

- Eine FBX-Datei enthält das DGM mit der dazugehörigen Textur (z. B. das DOP),
- Eine zweite Datei enthält die Gebäudemodelle - entweder LOD1- oder LOD2-Gebäude,
- und eine dritte Datei beinhaltet ausschließlich die Gewässerdaten.

Dieses strukturierte Vorgehen bringt den Vorteil, dass bestimmte Elemente – etwa die LODs oder die Gewässer – nur einmal exportiert und dann mehrfach wiederverwendet werden können. Dies spart erheblich Zeit bei der späteren Weiterverarbeitung in Blender. Dort müssen Gewässer und LODs lediglich einmalig angepasst werden und können anschließend flexibel in jedes Modell eingefügt werden.

Erweiterung zum Abschnitt „Aufbereitung in InfraWorks“:

Das bedeutet, dass die LODs und das DGM zusammen mit dem jeweiligen Orthophoto als separate Bestandteile exportiert werden. Insgesamt wurden sechs unterschiedliche Kartenausschnitte als FBX-Dateien ausgegeben jeweils Luftbildkarten von 1945, 1982 und 2024, eine Topografische Karte im Maßstab 1:25000, eine Uraufnahme vom Jahre 1817 und eine Positionskarte von 1870. Theoretisch könnten diese Daten direkt in Godot importiert werden. Das Format GLTF bzw. GLB ist für Godot besser geeignet, da der Import über das Format optimiert wurde und Meta-Informationen über Texturen, Materialien und Strukturen automatisch in Godot importeiert werden. Aus diesem Grund werden die Daten nicht direkt in die Game Engine übernommen, sondern zuvor noch einmal in Blender eingelesen und exportiert . (vgl. docs.godotengine.org)

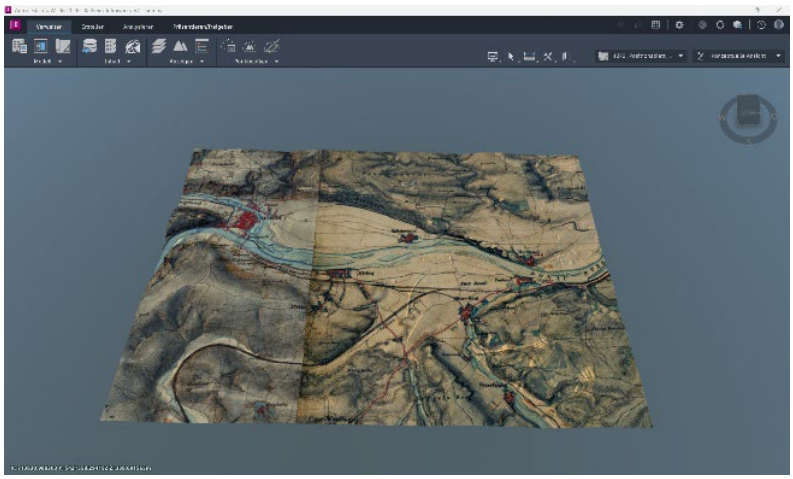


Abbildung 7: Positionskarte in InfaWorks

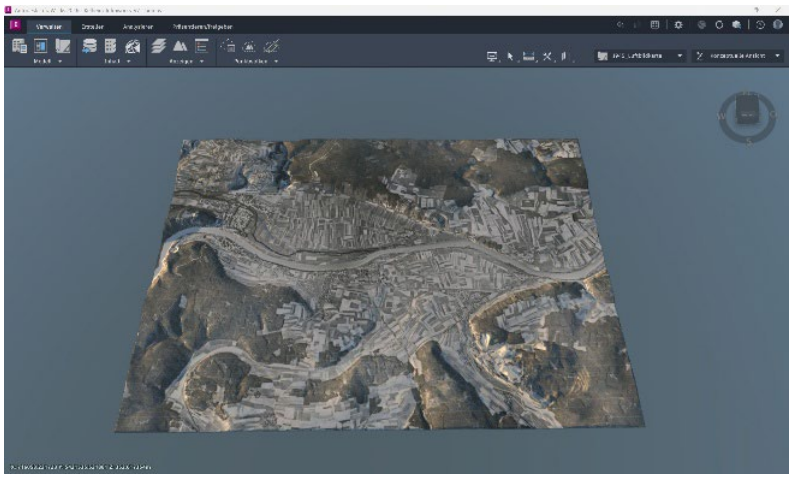


Abbildung 8: Luftbildkarte von 1945 in InfaWorks

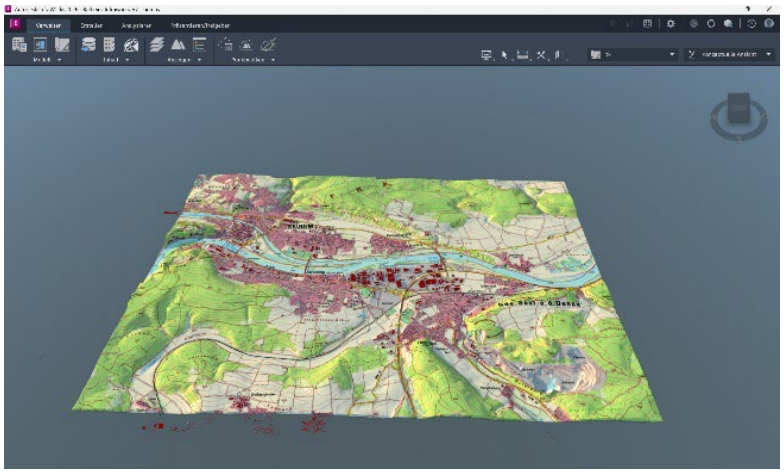


Abbildung 9: Topografische Karte 1:25000 in InfaWorks

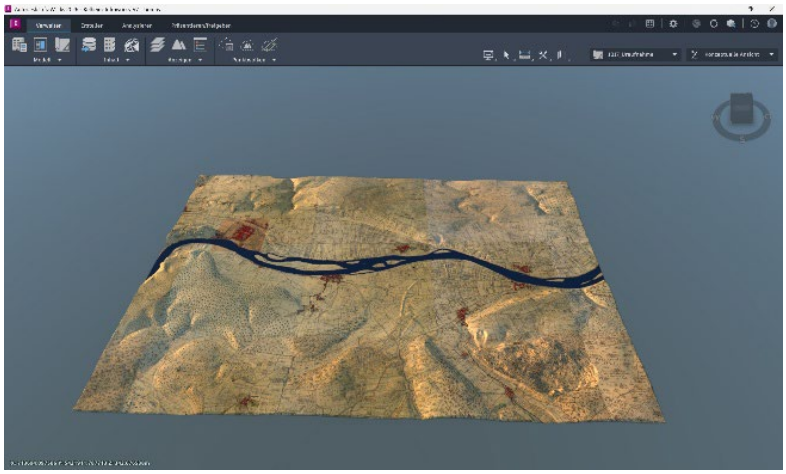


Abbildung 10: Uraufnahme von 1817 in InfaWorks

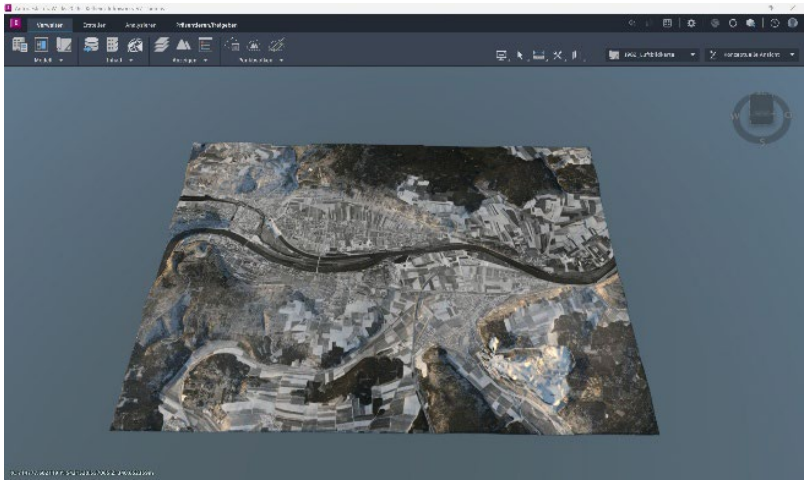


Abbildung 11: Luftbildkarte von 1982 in InfaWorks

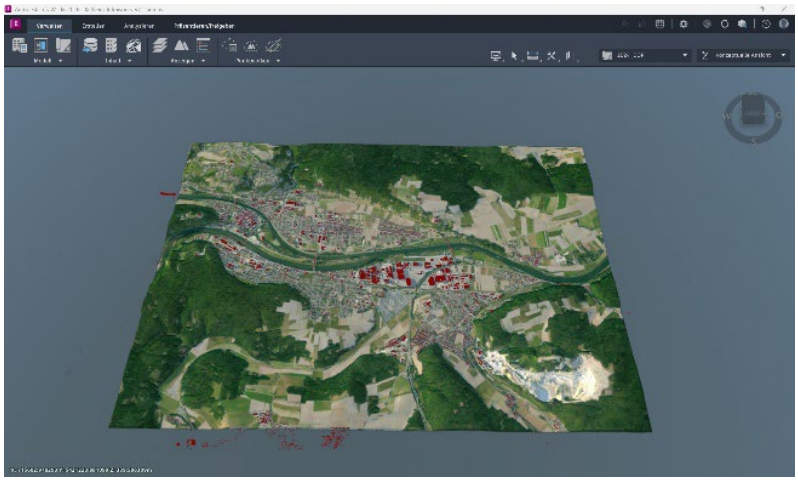


Abbildung 12: Aktuelle Luftbildkarte von 2024 in InfaWorks

Weiterverarbeitung in Blender

Im Gegensatz zu InfraWorks ist die Weiterverarbeitung der Daten in Blender etwas zeitaufwändiger, da beim Export aus InfraWorks die Geodaten wieder in Kacheln aufgeteilt werden. Zwar werden die Rohdaten bereits beim Import in InfraWorks automatisch zusammengefügt, beim Export erfolgt jedoch erneut eine Aufteilung. Das bedeutet, dass sowohl das DGM (Digitale Geländemodell) als auch das DOP (Digitales Orthophoto) in Blender wieder als einzelne Kachel-Objekte vorliegen und entsprechend einzeln bearbeitet werden müssen. Die Anzahl der Kacheln ist dabei von Karte zu Karte unterschiedlich.

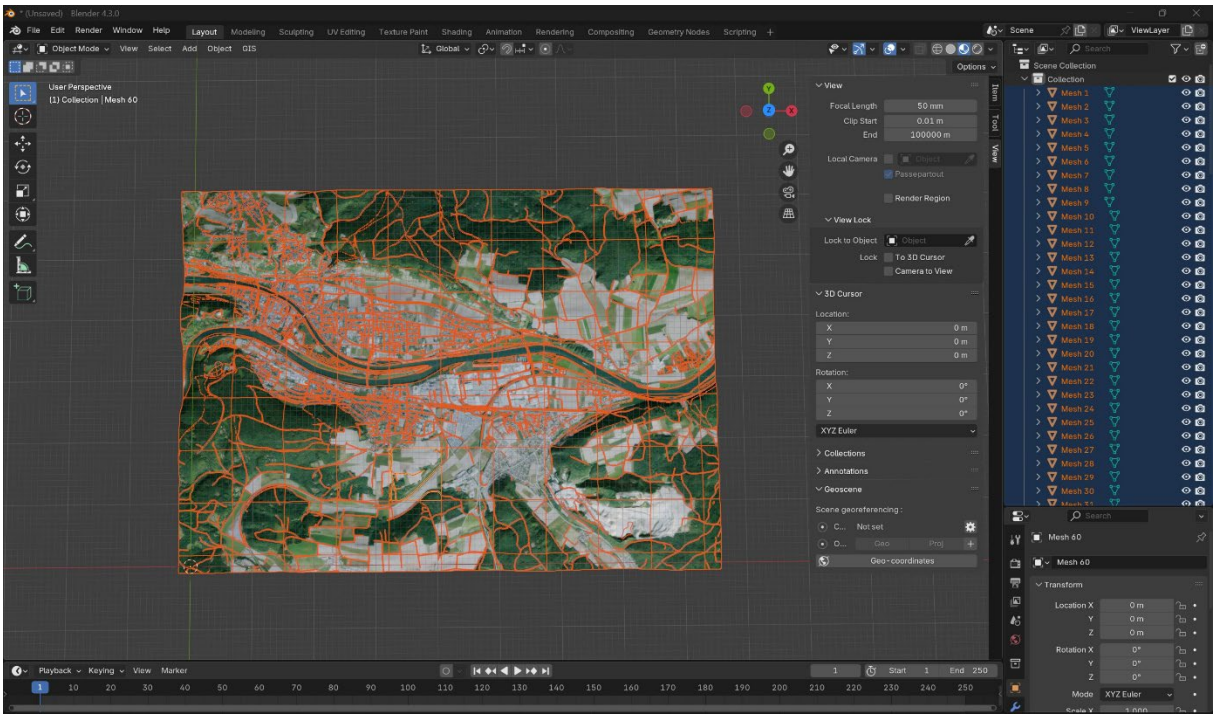


Abbildung 13: Aktelle Luftbildkarte in Blender

Bearbeitung der DOPs

Nach dem Import weisen die Luftbilddaten in Blender eine ungewollt glänzende Oberfläche auf . Dieses Problem lässt sich beheben, indem bei der Materialeinstellungder Wert „Roughness“ erhöht wird. Zusätzlich kann auch der „Metallic“-Wert reduziert werden, was das Erscheinungsbild weiter verbessert – allerdings kann dies die Farbdarstellung verfälschen. Deshalb wurde in den meisten Fällen nur der Roughness-Regler angepasst. In Einzelfällen war es jedoch notwendig, zusätzlich mit dem Metallic-Wert zu arbeiten, um ein harmonischeres Gesamtbild zu erreichen. Da diese Anpassungen manuell für jede Kachel durchgeführt werden müssen, ist dieser Schritt besonders zeitintensiv.

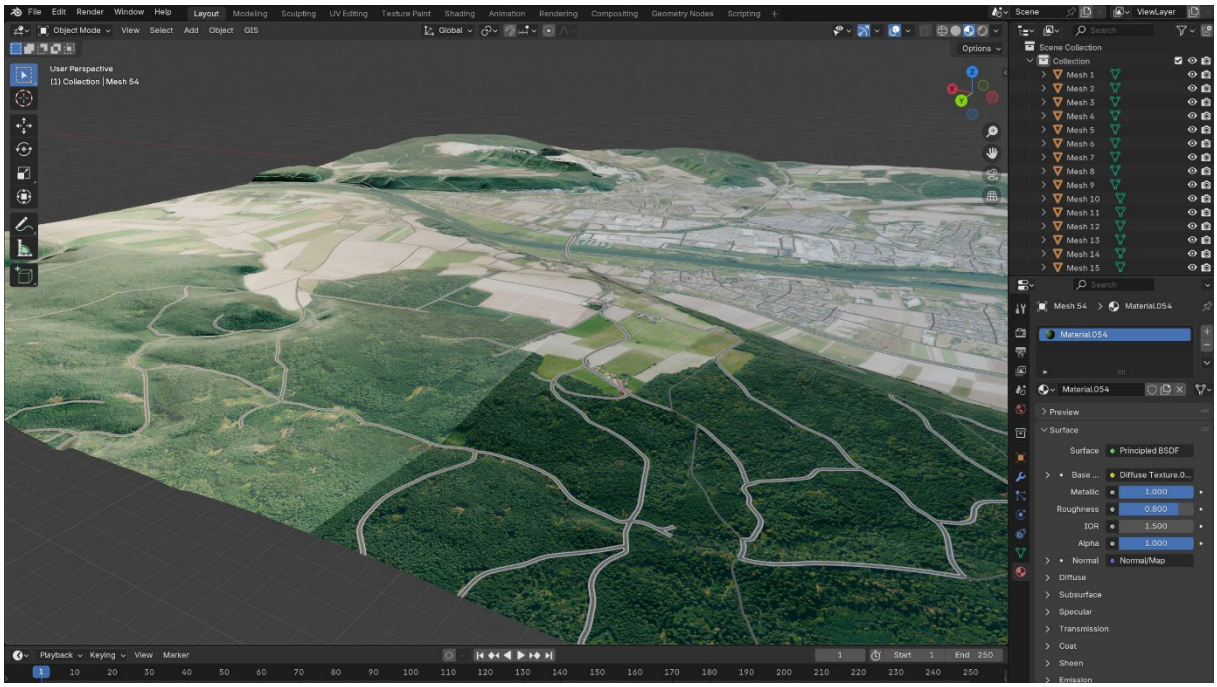


Abbildung 14: Unterschied zwischen Roughness hoch und niedriger

Bearbeitung der DGMs

Im nächsten Schritt werden die DGMs bearbeitet. Zunächst werden alle einzelnen Kacheln zu einer großen Fläche zusammengefügt. Dazu markiert man alle DGM-Objekte in Blender und nutzt die Tastenkombination Strg + J, um sie zu einem einzigen Objekt zu verbinden. Die DGM-Daten sind im Grunde reine Punktinformationen mit Lage und Höhe, die zu Dreiecksverbindungen - sogenannten Meshes - zusammengesetzt werden

Diese Meshes setzen sich aus Millionen l von Polygonen zusammen, die die Performance des Spiels direkt beeinflussen. Grundsätzlich gilt: Je weniger Polygone ein Objekt hat, desto performanter läuft das Spiel. Die Anzahl der Polygone hängt hauptsächlich von zwei Faktoren ab – der Gesamtgröße der Karte und der Bodenauflösung des Digitalen Geländemodells (DGM).
Je größer der Kartenausschnitt und je kleiner die Gitterweite, desto detaillierter, aber auch rechenintensiver wird das Mesh. In der Praxis muss hier ein geeigneter Kompromiss zwischen Detailgrad und Performance gefunden werden, indem beide Parameter entsprechend angepasst werden.

Unter Blender erfolgt deshalb eine Polygonredzierung, um die Leistungsfähigkeit des Spiels in der Game Engine zu verbessern. Da Godot alles in Echtzeit rendert, ist eine

niedrigere Polygonanzahl entscheidend. Dafür wird der „Decimate Modifier“ verwendet, mit dem die Anzahl der Polygone reduziert werden kann. Wichtig ist dabei, ein ausgewogenes Verhältnis zwischen Detailgenauigkeit und Performance zu finden. Ein guter Mittelwert lag bei den meisten Darstellungen bei 0.5, was einer Halbierung der Polygonanzahl entspricht. Bei besonders detailreichen Karten – etwa der topografischen Karte – konnte der Wert nur auf ca. 0.75 reduziert werden, da sich ansonsten Höhenlinien und Beschriftungen zu stark verzerrten.

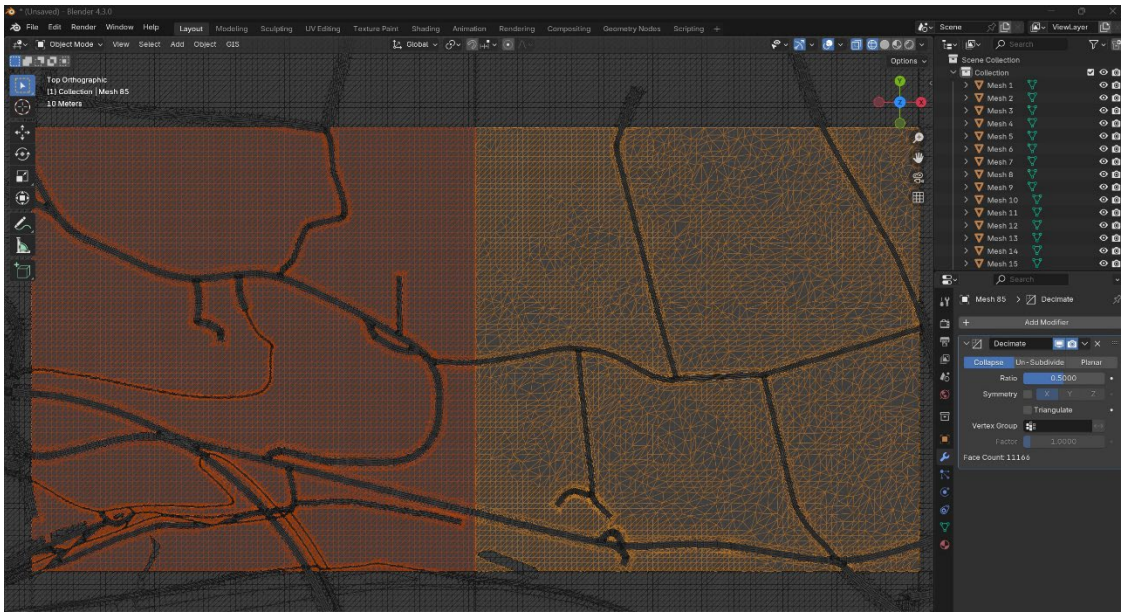


Abbildung 15: Polygonvergleich zwischen dem modifier decimate

Bearbeitung der LODs

Die LODs (Level of Detail) wurden ebenfalls einzeln bearbeitet. Der erste Schritt bestand darin, die Dachflächen farblich hervorzuheben – in diesem Projekt wurden sie der Einfachheit halber rot eingefärbt. Dazu wurden im Objektmodus alle Dachflächen markiert und eine entsprechende Materialfarbe zugewiesen.

Auch hier erfolgte eine Reduktion der Polygone über den „Decimate“-Modifier. Als geeigneter Faktor stellte sich 0.4 heraus. Wichtig war, dass trotz der Reduktion keine Lücken in den Gebäuden entstanden und die Form erhalten blieb.

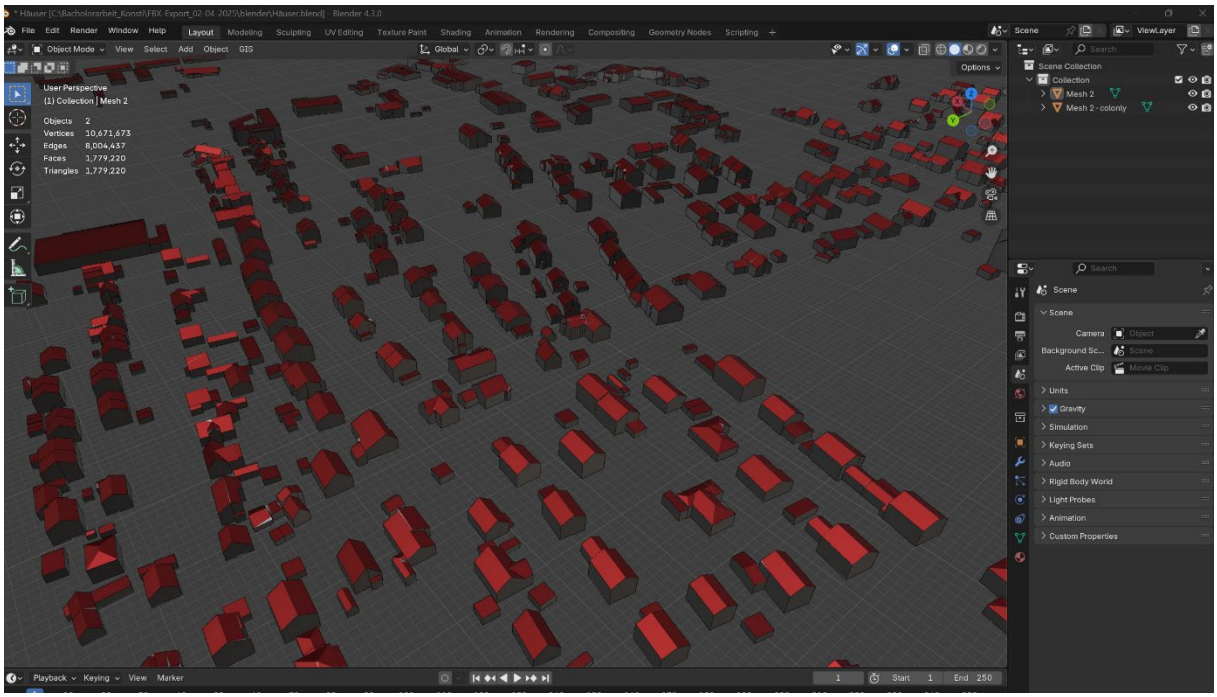


Abbildung 16: LOD2 in Blender

Bearbeitung der Gewässer

Auch die Gewässer wurden in Blender gesondert importiert und bearbeitet. Sie mussten nicht polygonreduziert werden und standen für den Export nach Godot direkt zur Verfügung.

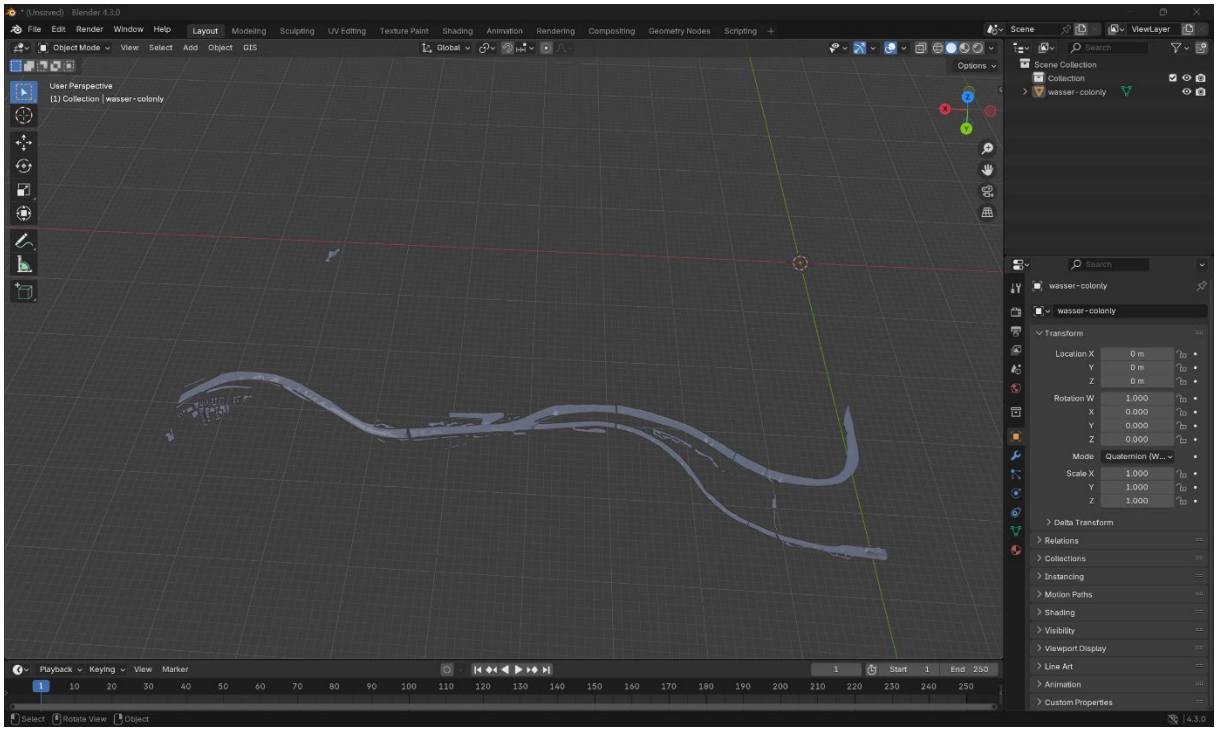


Abbildung 17: Gewässer in Blender

Export aus Blender

Nach der Bearbeitung wurden alle Elemente ins Format GLTF (GLB 2.0) exportiert – dies ist das optimal unterstützte Format für die Game Engine Godot.

Ein letzter wichtiger Schritt war die Erstellung von Collision Shapes. Eine Collision verhindert das Abtauchen in das Erdinnere – somit bleibt der Spieler immer auf der Erdoberfläche und fliegt nicht durch Objekte.. Die Erstellung erfolgt in Blender sehr einfach: Die Objekte (z. B. Gebäude, DGM, Gewässer) werden separat exportiert und im Dateinamen mit der Endung **„-**colonly**“** versehen. Beim Import in Godot erkennt die Engine diese automatisch als Kollisionselemente und ordnet sie korrekt zu.

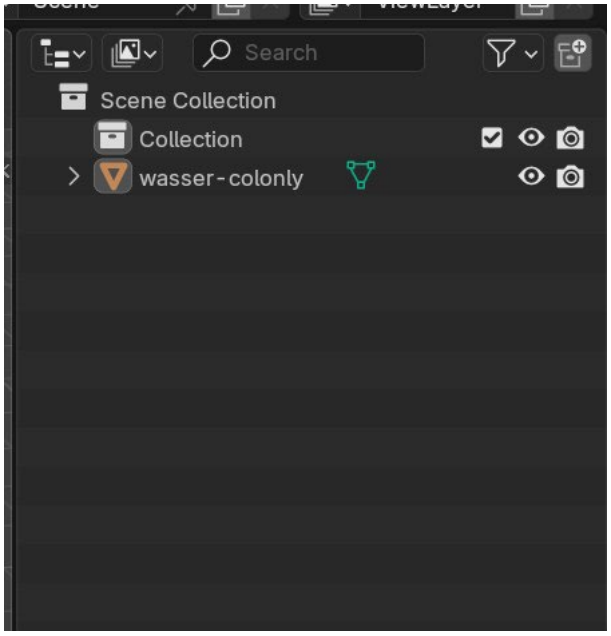


Abbildung 18: Gewässer mit dem Anhängsel -colonly

Umsetzung in Godot

Wie bereits in vorherigen Abschnitten beschrieben, ist das GLTF-Format (GLB2) besonders gut geeignet, um 3D-Modelle in Godot zu importieren. Der Importvorgang ist entsprechend unkompliziert: Unten links in Godot befindet sich die Ordnerstruktur. Um hier die Übersicht zu bewahren, empfiehlt es sich, einen eigenen Ordner für alle Importe anzulegen. In diesen kann man die GLB-Dateien ganz einfach per Drag-and-Drop einfügen. Der Import dauert in der Regel nur wenige Sekunden.

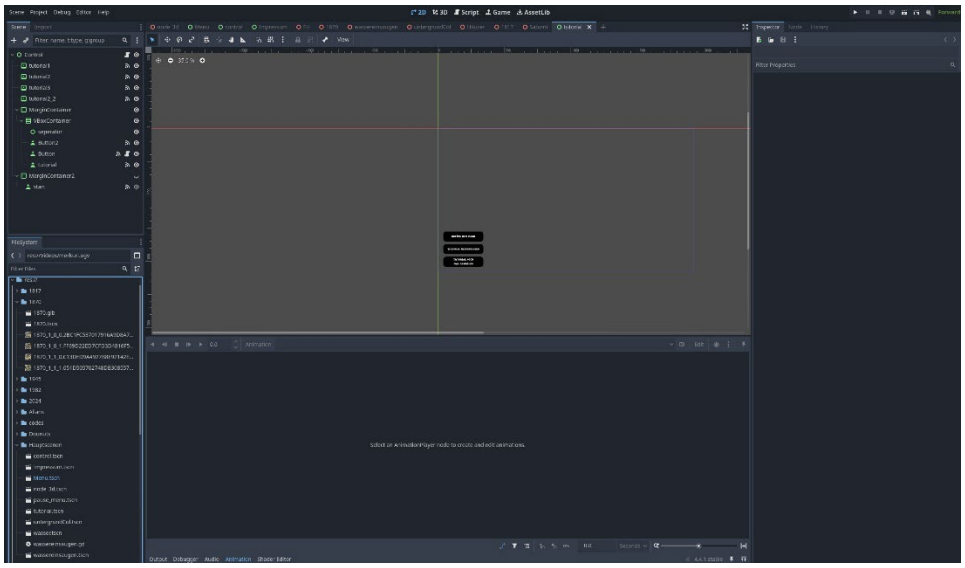


Abbildung 19: Lings untern die Ordner Struktur

Wichtig ist, dass jede importierte GLB-Datei zweimal geöffnet werden muss. Beim ersten Öffnen wird die Datei als nicht bearbeitbare Scene geladen. Um das Modell bearbeiten zu können, muss man es ein zweites Mal öffnen und anschließend als eigene, neue Scene abspeichern. Diesen Schritt wiederholt man für jedes einzelne Modell.

Liegen alle Modelle als eigenständige Scenes vor, lassen sich diese beliebig kombinieren und modular einsetzen. Das brigt einen großen Vorteil: Einzelne Scenes bleiben übersichtlich und können gezielt bearbeitet werden e. Gerade bei komplexen Projekten mit vielen Objekten – etwa einer umfangreichen Karte – kann es schnell unübersichtlich werden, wenn alle Modelle in einer einzigen Haupt-Scene liegen. Werden z. B. an einem Kartenmodell nur kleine Änderung vorgenommen, kann das in der Haupt-Scene unpraktisch sein. Stattdessen werden einfach nur die entsprechende Karten-Scene geöffnet und führt dort die Änderungen durch. Das macht die Arbeit nicht nur übersichtlicher, sondern auch deutlich effizienter. Weitere Bearbeitungsschritte finden sich in den Kapiteln zu den einzelnen Gameplay-Mechaniken.

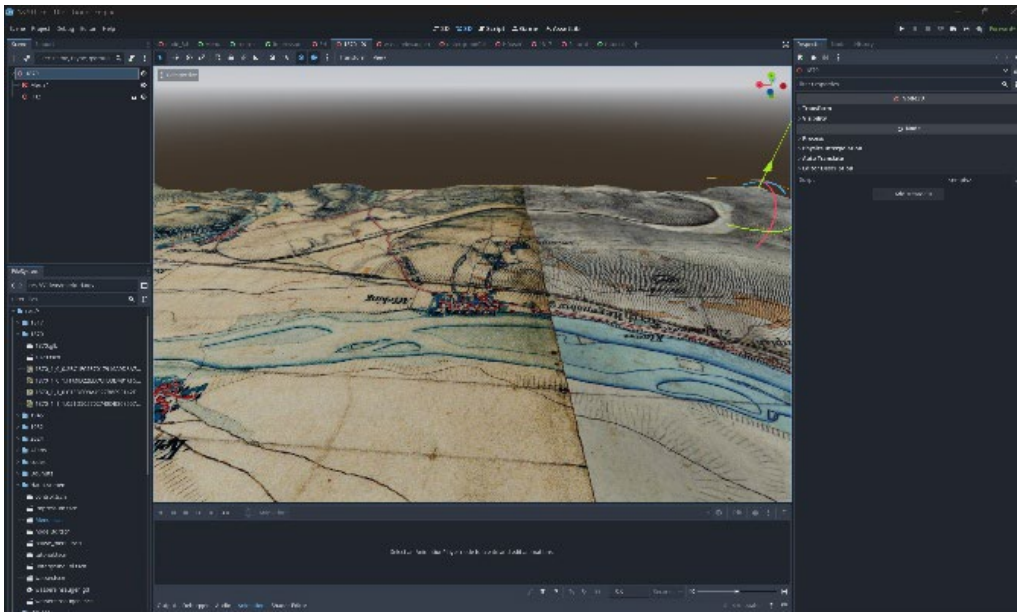


Abbildung 20: Positionskarte in Godot

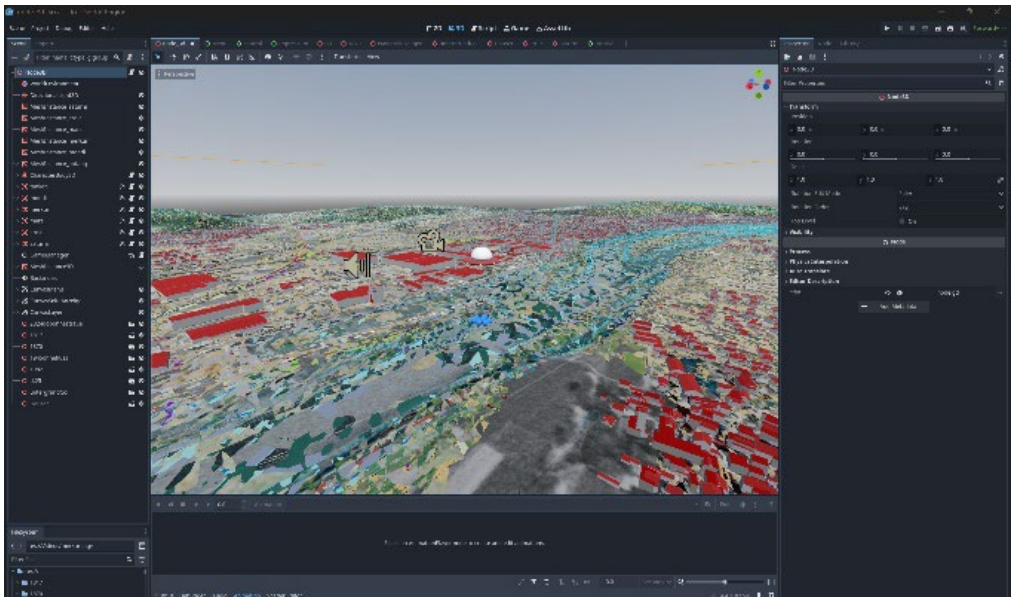


Abbildung 21: Alle Karten Übereinander in Godot

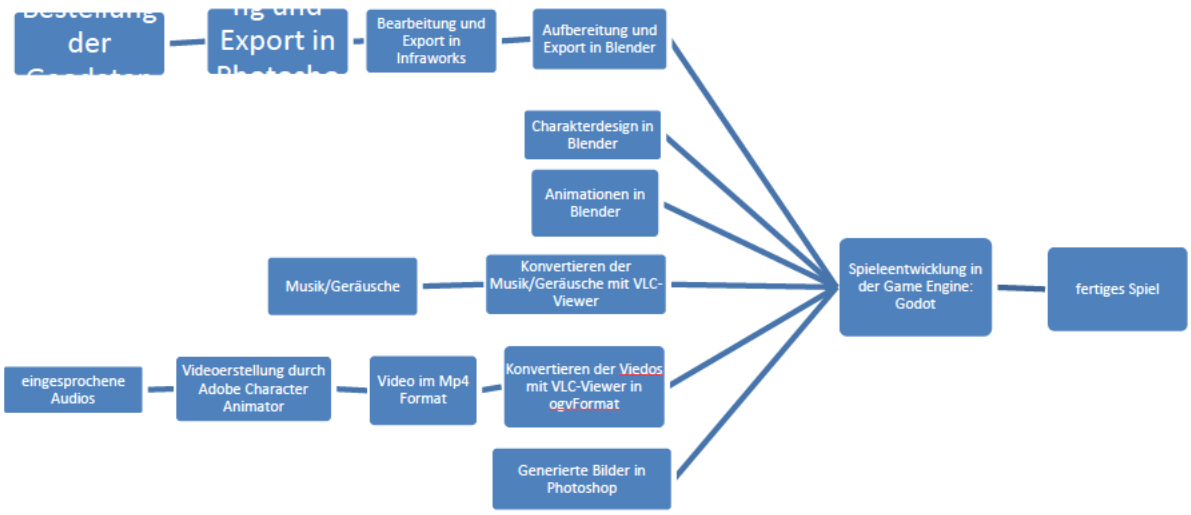


Abbildung 22: Workflow des ganzen Projektes

3.5 Entwicklung des Spiels

3.5.1 Aufbau/Story des Spiels

Im Game übernimmt der Spieler die Kontrolle über ein Alien, das seine Kinder auf der Erde verloren hat. Das Ziel besteht darin, die kleinen Aliens mit einem UFO einzusammeln.

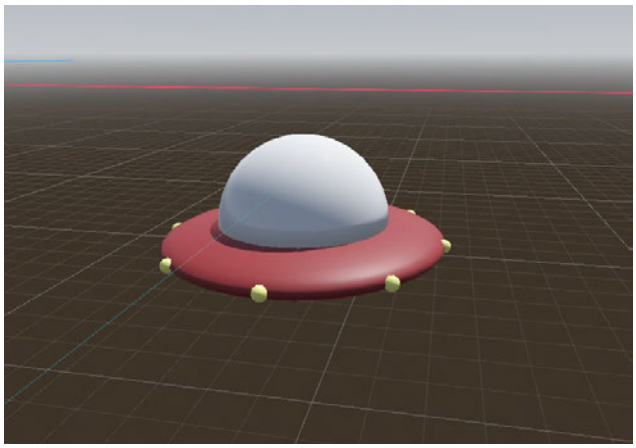


Abbildung 23: Ufo Modell

Beim Start des Spiels Öffnet sich zunächst das Hauptmenü. Dort kann der Spieler zwischen drei Optionen wählen: das Spiel starten, das Spiel beenden oder das Impressum aufrufen (letzteres ist für Kinder eher uninteressant).

Entscheidet sich der Spieler für den Spielstart, wird er zunächst in eine Tutorial-Szene geleitet. Dort erklärt das Eltern-Alien, worum es geht, welches Ziel das Spiel verfolgt und wie das UFO gesteuert wird. Im Tutorial gibt es die Möglichkeit, entweder zum Hauptmenü zurückzukehren oder das Tutorial zu überspringen.

Nach dem Tutorial (oder wenn es übersprungen wurde) beginnt das eigentliche Spiel. Die Aufgabe des Spielers ist es, die fünf verlorenen Alien-Kinder namens Mondi, Saturni, Erdie, Marsi und Merkuri einzusammeln. Jedes Mal, wenn ein Alien gefunden und aufgenommen wird, verändert sich die Oberfläche der Karte:

- Mondi**:** Die Landschaft wechselt zu Uraufnahme von 1817
- Saturni**:** Die Landschaft wechselt zu Aufnahme von 1945
- Erdi**:** Die Landschaft wechselt zu Topografischen Karte 1:25 000
- Marsi**:** Die Landschaft wechselt zu planungskarte von 1870

- Merkuri**:** Die Landschaft wechselt zu zu Aufnahme von 1982

Abbildung 24: Die Alien Kinder: Erdi, Marsi, Saturni, Mondi und Merkuri

Nach jeder erfolgreichen Rettung erscheint das Eltern-Alien erneut und gibt Informationen über die jeweilige Umgebung oder zum Kartenmodell.

Sobald alle fünf Aliens eingesammelt sind, gibt es eine kleine unerwartete Wendung in dem Spiel: Das Eltern-Alien informiert den Spieler, dass der Treibstoff des UFOs aufgebraucht ist und es nun mit Wasser aufgetankt werden muss. Der Spieler muss daraufhin ein nahegelegenes Gewässer finden und das UFO dort aufzufüllen.

Nachdem das UFO erfolgreich betankt wurde, endet das Spiel. Der Spieler wird zur End Szene weitergeleitet und hat dort die Möglichkeit, das Spiel erneut zu starten oder ins Hauptmenü zurückzukehren.

3.5.2 Steuerung des Spiels

Das Spiel kann sowohl mit Maus und Tastatur als auch mit einem Controller gespielt werden. Es wurde jedoch primär für die Steuerung mit einem Controller optimiert. Dies betrifft sowohl die Navigation im Menü als auch das Steuern des UFOs im Hauptspiel.

Das Spiel wurde mit einem PlayStation- und einem Xbox-Controller getestet. Die Steuerung ist für beide Controller identisch. Godot ermöglicht es, Eingaben flexibel zu konfigurieren, sodass eine Funktion mit mehreren Eingabegeräten gleichzeitig genutzt werden kann

3.5.3 Steuerung des Ufos

- Vorwärts fliegen: Linker Joystick nach vorne (Tastatur: W)
- Rückwärts fliegen: Linker Joystick nach hinten (Tastatur: S)
- Seitwärts rechts fliegen: Linker Joystick nach rechts (Tastatur: D)
- Seitwärts links fliegen: Linker Joystick nach links (Tastatur: A)
- Nach oben fliegen: R2 gedrückt halten (Tastatur: Leertaste)
- Nach unten fliegen: L2 gedrückt halten (Tastatur: Shift)
- Beam aktivieren: L1 oder R1 gedrückt halten (Tastatur: Linke Maustaste)
- Geschwindigkeitsboost aktivieren: A gedrückt halten (Tastatur: K)
- Kameraposition anpassen: Rechter Joystick (Tastatur: Mausbewegung)
- Menü-Navigation: Linker Joystick zur Auswahl, A zur Bestätigung
- Spiel pausieren: Start-Taste auf dem Controller (Tastatur: ESC)

Abbildung 25: Controller Von der Draufsicht

Abbildung 26: Controller von der Vorderansicht

3.5.4 Gameplay-Mechaniken und deren Umsetzung in der Gameengen

Um die Spielmechaniken besser nachvollziehen zu können, folgt zunächst ein kurzer Abschnitt über die Grundlagen der Game Engine Godot.
In diesem Abschnitt werden die Grundlagen von Godot beschrieben, um einen groben Überblick über den Aufbau der Game Engine zu geben und spätere Zusammenhänge verständlicher zu machen.

Nach dem Öffnen von Godot , wird n – wie in den meisten Programmen – zunächst ein Projekt mit einem Namen erstellt. Dieses Projekt wird in einem separaten Ordner gespeichert, was es ermöglicht, es schnell auf verschiedenen Geräten zu sichern und weiterzuverwenden. Das bedeutet, dass ein Projekt einfach per Drag & Drop in Godot geöffnet werden kann, wenn es auf einem neuen Gerät noch nicht vorhanden ist.

Nach dem Start von Godot erscheint die Arbeitsoberfläche. Im Zentrum befindet sich der Viewport, in dem die aktuelle Szene visuell dargestellt und bearbeitet wird. Der Viewport ist das Hauptarbeitsfeld und ermöglicht die Anzeige sowie Bearbeitung von 2D- und 3D-Elementen in Echtzeit. Direkt darüber befindet sich die Toolbar, die verschiedene Werkzeuge wie Bewegen, Rotieren oder Skalieren von Objekten enthält.

Auf der rechten Seite der Benutzeroberfläche befindet sich der Inspektor. Hier können die Eigenschaften des aktuell ausgewählten Objekts angepasst werden. Unten links befindet sich die Dateistruktur, die für das Importieren von Spielelementen wichtig ist. Darüber liegt die Scene Dock List, mit ihr können die verschiedenen Szenen gewechselt werden.

Die Szenen sind in Godot die grundlegenden Bausteine eines Spiels. Eine Szene ist eine Sammlung von Nodes (Knoten), die zusammen eine bestimmte Funktion oder ein Objekt im Spiel darstellen. Es können drei Szenenarten gewählt werden:

- 3D-Szene: Für die Erstellung von Objekten und Umgebungen im dreidimensionalen Raum.
- 2D-Szene: Für die Entwicklung von 2D-Spielen.
- User Interface (UI): Speziell für Menüs und Benutzeroberflächen.

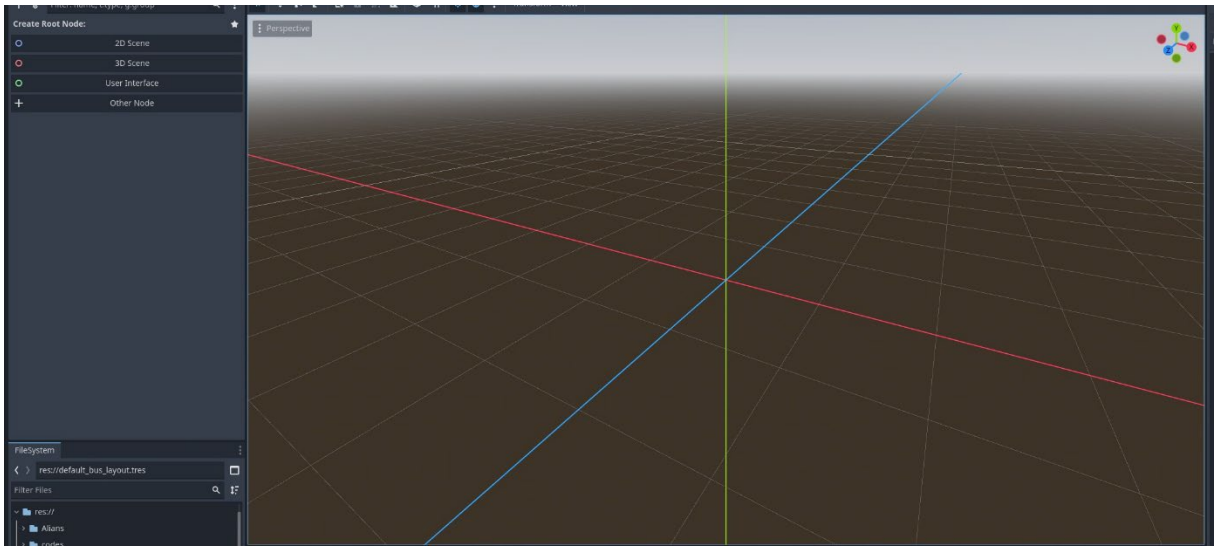


Abbildung 27: Godot UI

Szenen lassen sich hierarchisch organisieren, wiederverwenden und miteinander kombinieren, um komplexe Spielstrukturen zu erstellen.

Jede Szene besteht aus Nodes – den grundlegenden Bausteinen des Spiels. Nodes können verschiedene Funktionen haben, zum Beispiel:

- Spielercharaktere
- Kameras
- Meshes (3D-Modelle)
- Animations-Player
- Buttons

Nodes sind in drei Kategorien unterteilt, die farblich markiert sind:

- Rote Nodes – 3D-Nodes
- Blaue Nodes – 2D-Nodes
- Grüne Nodes – UI-Nodes

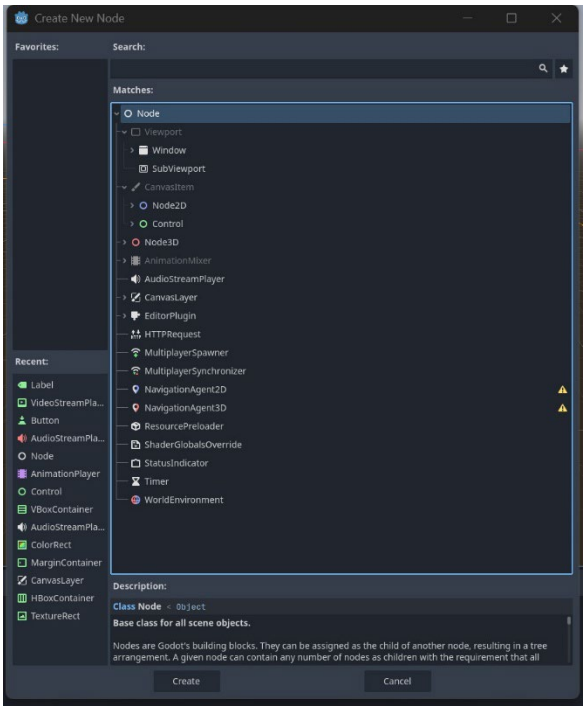


Abbildung 28: Verschiedenen Nodes

Zusätzlich gibt es graue Nodes, die keine direkten Auswirkungen auf das Spiel haben, sondern als Overlays oder Steuerungselemente dienen.

Nodes lassen sich hierarchisch anordnen: Ein übergeordnetes Haupt-Node kann untergeordnete Child-Nodes haben. Child-Nodes beziehen sich immer auf ihr übergeordnetes Node, was die Bearbeitung vereinfacht. Wenn das Haupt-Node bewegt wird, werden alle Child-Nodes mitbewegt. Es gibt auch Nodes, die nur als Child-Nodes existieren können.

Zu jedem Node kann ein Skript hinzugefügt werden. Skripte steuern das Verhalten von Nodes und sind in der Godot-eigenen Programmiersprache GDScript geschrieben, die stark an Python angelehnt ist.

Nodes können über Signale miteinander kommunizieren. Zum Beispiel, wenn ein Timer abläuft, kann er ein Signal an ein anderes Node senden. Besonders wichtig sind Signale bei der Interaktion von Objekten, wenn ein Spieler zum Beispiel mit einem Gegenstand kollidiert, wird durch die Funktion Collider-Nodes erkannt und ausgelöst.

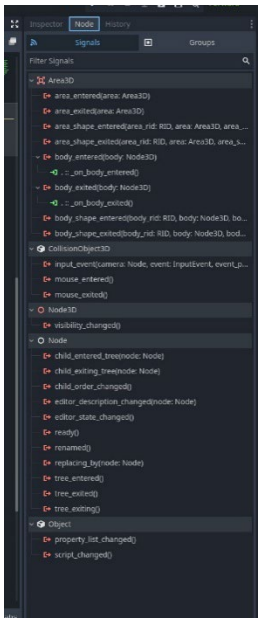


Abbildung 29: Verschiedene Signale

Collider sind unsichtbare Bereiche in einem Spiel, die für die physikalische Interaktion zwischen Objekten verantwortlich sind. Sie bestimmen, ob eine Spielfigur gegen eine Wand läuft, ein Objekt auf dem Boden landet oder mit einem Gegner kollidiert. In Godot gibt es dafür verschiedene Collider-Typen, wie zum Beispiel *CollisionShape2D* oder *CollisionPolygon2D* für 2D-Spiele. Diese Collider werden an sogenannte *PhysicsBody*-Knoten (z. B. KinematicBody2D, StaticBody2D oder RigidBody2D) angehängt, um festzulegen, wie sich das Objekt im Spiel verhalten soll. Dabei definiert der Collider die physikalische Form des Objekts – meist als Rechteck, Kreis oder benutzerdefinierte Form – die dann für Kollisionserkennung und Bewegungsbegrenzungen sorgt. Ohne Collider wüsste das Spiel nicht, wo Grenzen, Hindernisse oder Triggerpunkte sind. Sie sind also essenziell, damit das Spiel „weiß“, wann etwas gegen etwas anderes stößt oder mit etwas interagiert.

Spielmechaniken

Die Spielmechaniken beschreiben die wichtigsten Abläufe eines Spiels, die in der Game Engine umgesetzt wurden. Dazu gehört, welche Nodes verwendet wurden, welche Signale zum Einsatz kommen, welche Funktionen die einzelnen Spielobjekte haben und welche Code-Strukturen dahinterstehen. Ziel ist es, einen Einblick in die Funktionsweise des Spiels zu geben.

Beim Design des Spiels wurde bewusst darauf geachtet, dass es nicht zu viele Gameplay-Mechaniken gibt und dass diese möglichst simpel gehalten sind. Das

bedeutet, dass nur wenige Eingaben erforderlich sind, um das Spiel zu steuern, sodass es intuitiv und leicht zugänglich bleibt.

Im Spiel bewegt sich der Spieler frei in einer dreidimensionalen Open-World. Der Begriff *Open-World* beschreibt eine Spielumgebung, in der sich der Spieler ohne festgelegte Pfade oder Einschränkungen bewegen und die Welt eigenständig erkunden kann. Auch wenn die in dieser Arbeit verwendete Spielwelt nur einen ausgewählten geografischen Ausschnitt darstellt, besteht innerhalb dieses Bereichs vollständige Bewegungsfreiheit. Der Spieler kann beliebig die Höhe variieren, die Flugroute selbst wählen und sich frei in der Umgebung orientieren.

Aus Sicht der Game Engine besteht die Spielwelt im Wesentlichen aus einem *Mesh* mit einem zugewiesenen Material. Die Meshes bestehen aus mehreren Oberflächen, die sich aus den in Kapitel 3.1.3 beschriebenen Kacheln zusammensetzen. Beim Import in Godot werden diese als einfache 3D-Nodes eingebunden und mit den jeweiligen Materialien versehen.
Zur Begrenzung der Spielwelt wurden in Godot statische Kollisionen eingebaut. Der Boden und die Gebäude bestehen aus sogenannten *StaticBody*-Elementen mit einer entsprechenden *CollisionShape*. Diese verhindern, dass sich andere physikalische Objekte (wie der Spielercharakter) durch sie hindurch bewegen können. Die *StaticBody* wirkt somit wie ein unsichtbares Netz, das die Form des DGMs und der Gebäude (*LODs*) abbildet. Zusätzlich wurden an den Seiten der Spielwelt unsichtbare Mauern implementiert, um den Spielbereich klar einzugrenzen.

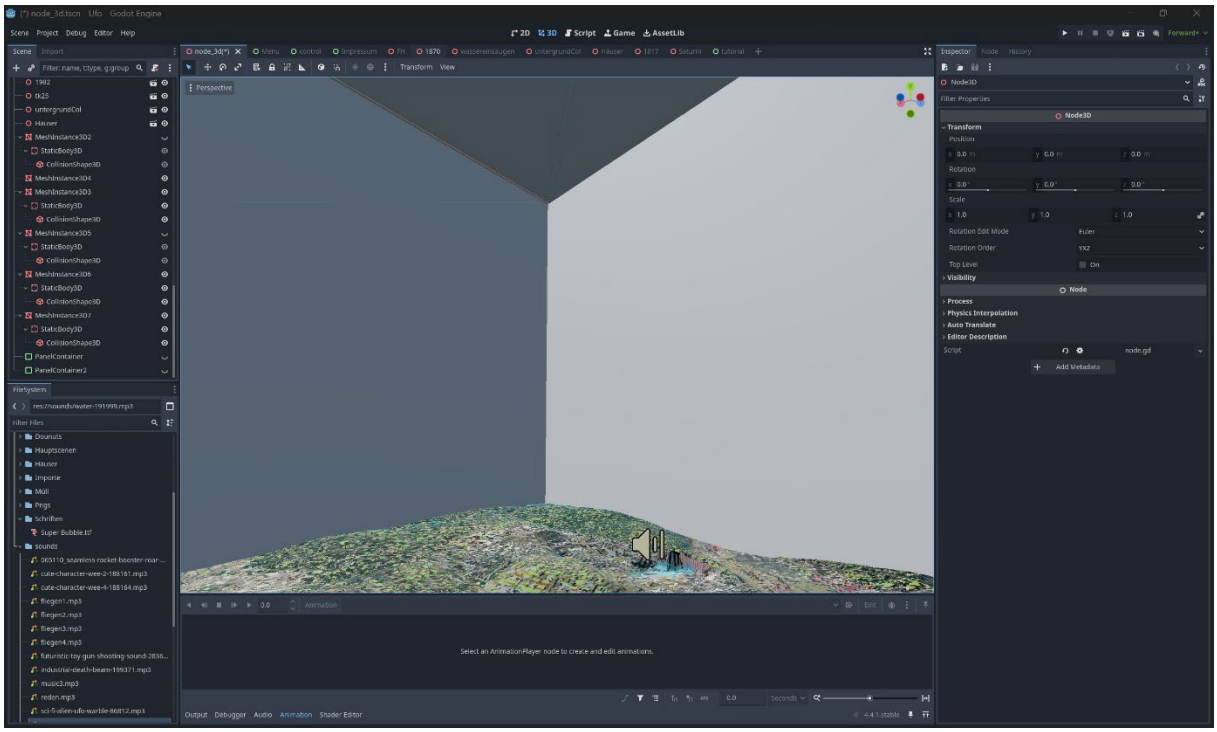


Abbildung 30: Spielbegrenzungen in Godot

Wie bereits im Abschnitt Steuerung beschrieben, kann sich das UFO in alle Richtungen frei bewegen. Wenn keine Eingabe erfolgt, bleibt es auf der Stelle stehen. Das UFO verfügt über keine Beschleunigung, sondern bewegt sich stets mit einer konstanten Geschwindigkeit. Diese Entscheidung erleichtert das Manövrieren für den Spieler erheblich.

Um zusätzliches Feedback zu bieten, werden zwei verschiedene Soundeffekte verwendet:

Im Stand: Ein sanftes, schwebendes Geräusch vermittelt das Gefühl von Leichtigkeit.

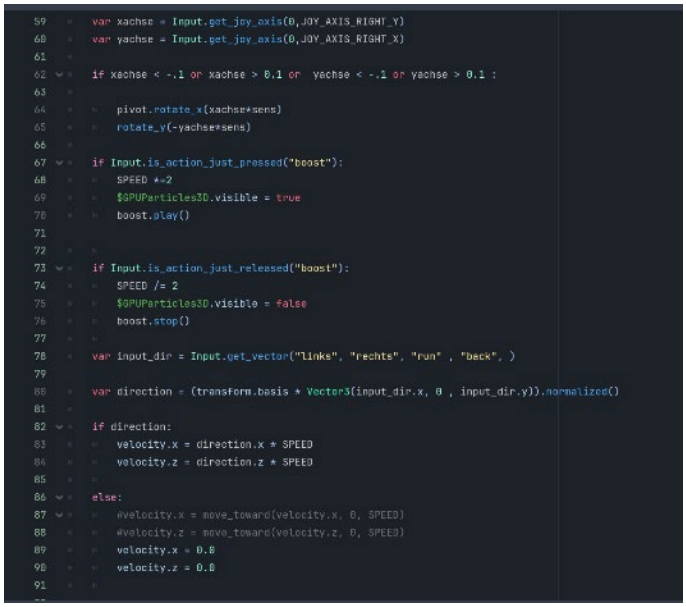
In Bewegung: Ein leicht verstärkter Flug-Sound signalisiert, dass sich das UFO aktiv durch die Spielwelt bewegt.

Diese akustische Unterscheidung unterstützt den Spieler dabei, intuitiv zu erfassen, ob das UFO gerade gesteuert wird oder nicht.

Aus technischer Sicht ist die Bewegung in Godot relativ einfach umzusetzen. Das UFO ist in der Engine als Spieler-Node angelegt, an dem ein Skript zur Steuerung der Bewegungen und Interaktionen im Spiel implementiert ist, um ein dynamisches und immersives Spielerlebnis zu ermöglichen. In diesem Skript ist die Variable speed definiert, welche die feste Bewegungsgeschwindigkeit des UFOs bestimmt – eine Beschleunigung ist also nicht vorgesehen.

Im Code gibt es zwei Hauptblöcke:

Bewegung in X- und Z-Richtung (links, rechts, vorwärts, rückwärts)
Bewegung in Y-Richtung (hoch und runter)
Die Eingaben erfolgen über den linken Joystick für die horizontale Bewegung und über die Schultertasten für das Auf- und Absteigen. Sobald eine Eingabe erfolgt, wird die jeweilige Richtung mit dem speed-Wert multipliziert, und das UFO bewegt sich entsprechend. Wird die Eingabe beendet, wird die Geschwindigkeit wieder auf null gesetzt, und das UFO bleibt stehen.
Soundeffekte



```
var xachse = Input.get_joy_axis(0,JOY_AXIS_RIGHT_Y)
var yachse = Input.get_joy_axis(0,JOY_AXIS_RIGHT_X)

if xachse < -.1 or xachse > 0.1 or  yachse < -.1 or yachse > 0.1 :

    pivot.rotate_x(xachse*sens)
    rotate_y(-yachse*sens)

if Input.is_action_just_pressed("boost"):
    SPEED *=2
    $GPUParticles3D.visible = true
    boost.play()


if Input.is_action_just_released("boost"):
    SPEED /= 2
    $GPUParticles3D.visible = false
    boost.stop()

var input_dir = Input.get_vector("links", "rechts", "run" , "back", )

var direction = (transform.basis * Vector3(input_dir.x, 0 , input_dir.y)).normalized()

if direction:
    velocity.x = direction.x * SPEED
    velocity.z = direction.z * SPEED

else:
    #velocity.x = move_toward(velocity.x, 0, SPEED)
    #velocity.z = move_toward(velocity.z, 0, SPEED)
    velocity.x = 0.8
    velocity.z = 0.8
```

Abbildung 31: Skriptbeispiel der Steuerung des Ufos

```
if Input.is_action_just_pressed("up"):
    movement= Vector3(0,SPEED,0)
if Input.is_action_just_pressed("down"):
    movement = Vector3(0,-SPEED,0)


elif Input.is_action_just_released("up")|| Input.is_action_just_released("down"):
    movement = Vector3.ZERO


if input_dir != Vector2.ZERO || movement != Vector3.ZERO:
    if !fliegen.playing:
        schweben.stop()
        fliegen.play()
else:
    if !schweben.playing:
        fliegen.stop()
        schweben.play()

move_and_collide(movement*delta)
```

Abbildung 32: Skriptbeispiel der Steuerung des Ufos

Die Umsetzung der Soundeffekte erfolgt mit zwei AudioStreamPlayer-Nodes:

- Beim Stillstand des Ufos wird der Schwebe-Sound eingespielt
- Bei der Bewegung wird der Flug-Sound abgespielt

Im Skript wird geprüft, ob eine Eingabe zur Bewegung erfolgt. Bei einer Bewegung des Ufos wird der Schwebe-Sound gestoppt und der Flug-Sound abgespielt.
Wenn nein, passiert das Gegenteil: Der Flug-Sound wird gestoppt, und der Schwebe-Sound startet.
Diese Sounds tragen maßgeblich zur Lebendigkeit der Spielwelt bei und sorgen dafür, dass sich das Spielerlebnis dynamischer anfühlt – auch wenn sich das UFO im Leerlauf befindet.

Im Spiel ist die Kamera mit der Funktion Third-Person-Controller ausgestattet, der es dem Spieler ermöglicht, sich frei umzusehen. Diese Perspektive ist besonders wichtig, da der Spieler Objekte suchen muss und durch die Kameraansicht die Steuerung des UFOs präzise und intuitiv erfolgt. Die Kamerastruktur besteht aus insgesamt drei Nodes: Zunächst gibt es einen normalen 3D-Node, der als Mittelpunkt dient. An diesem Node ist ein Springarm befestigt, an dem wiederum die eigentliche 3D-Kamera angebracht ist. Der Springarm fungiert als Platzhalter zwischen dem Ursprungspunkt und der Kamera, sodass sich die Kamera in einem festen Radius um den 3D-Node bewegen kann.

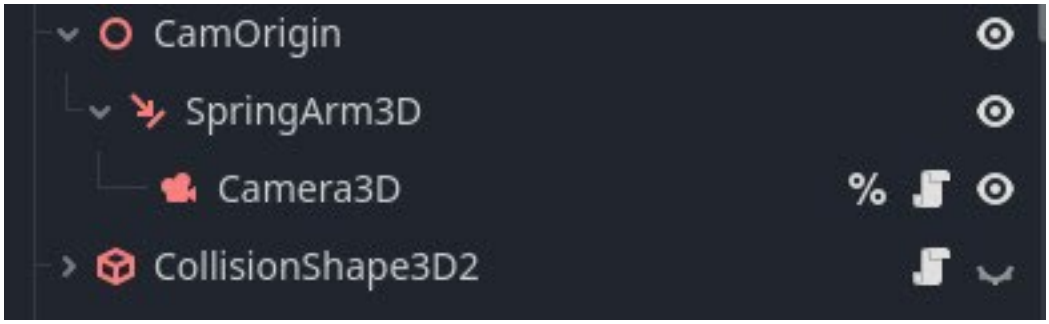


Abbildung 33: Aufbau der Kamera mit Nodes

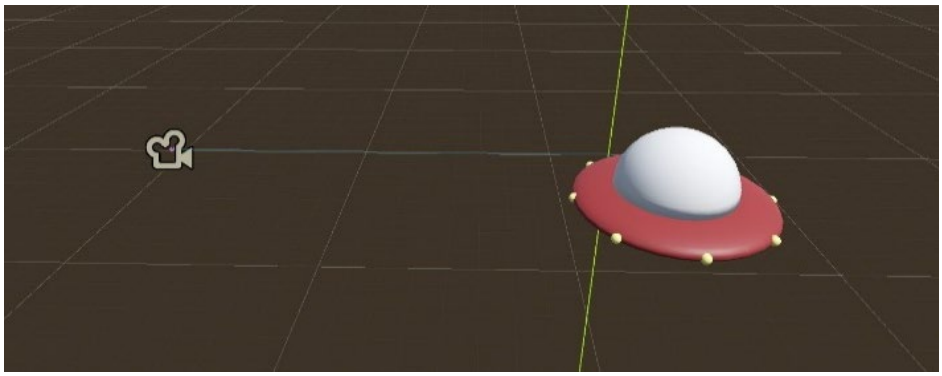


Abbildung 34: Kamera und Ufo

Das Sichtfeld der Kamera in Godot ist wie eine Pyramide aufgebaut, das bedeutet, dass die Kamera nur nach vorne schaut und einen begrenzten Winkel hat. Dabei ist die Kamera stets auf den Ausgangspunkt (also das UFO) ausgerichtet. Zu jedem Zeitpunkt sieht der Spieler das UFO, was eine konstante Orientierung gewährleistet.
Die Bewegung der Kamera wird durch die Eingaben des Joysticks gesteuert. Diese Eingaben werden im Code verarbeitet und mit einer Sensibilität verrechnet, die zu Beginn des Spiels definiert wird. Die Sensibilität hilft dabei, die Kamerabewegung feiner und präziser zu steuern. Wenn der Joystick bewegt wird, gibt er einen Wert aus, der je nach Entfernung der Joystickbewegung größer oder kleiner wird. Diese Werte sind jedoch relativ hoch, weshalb sie mit einem kleinen Sensibilitätswert multipliziert werden, um die Geschwindigkeit der Kamerabewegung zu reduzieren und eine sanftere Steuerung zu ermöglichen. Zudem wurde im Code festgelegt, dass die Kamerabewegung erst ab einer bestimmten Schwelle ausgelöst wird, um zu verhindern, dass schon die kleinste Joystickabweichung die Kamera bewegt.
Die resultierende Bewegung beeinflusst die Rotation der Kamera in der X- und Y-Achse. Dabei ist zu beachten, dass die Kamera hauptsächlich entlang der Y-Achse (also in vertikaler Richtung) frei bewegt werden kann, während die Rotation entlang der X-Achse das gesamte UFO dreht. Diese Konfiguration wurde absichtlich so gewählt, damit die Blickrichtung des Spielers immer mit der Flugrichtung des UFOs übereinstimmt.
Ein spezieller Code-Pivot sorgt dafür, dass die Kamera nur um den Ursprungspunkt rotiert. Hätte auch die X-Achse eine Pivot-Rotation, würde sich die Blickrichtung bei einer Drehung um 90 Grad nach rechts verschieben und nicht mehr mit der Flugrichtung übereinstimmen. Dies würde zu einer verwirrenden Steuerung führen,

weshalb bei der X-Achse das ganze UFO mitbewegt wird, um die Sichtachse stets mit der Bewegungsrichtung des UFOs abzustimmen.

```
58
59      var xachse = Input.get_joy_axis(0,JOY_AXIS_RIGHT_Y)
60      var yachse = Input.get_joy_axis(0,JOY_AXIS_RIGHT_X)
61
62      if xachse < -.1 or xachse > 0.1 or  yachse < -.1 or yachse > 0.1 :
63
64          pivot.rotate_x(xachse*sens)
65          rotate_y(-yachse*sens)
```

Abbildung 35: Skriptbeispiel von dem Joystick eingaben

Alle Aliens im Spiel teilen sich dieselbe Standard-Laufanimation. Das bedeutet, dass sie sich, solange der Spieler nicht mit ihnen interagiert, mit kleinen Sprüngen in einem Kreis um ihren festgelegten Platz bewegen, bis sie eingesammelt werden. Jedes Alien verfügt jedoch über eine individuelle Einsammel-Animation, die sich in der Art und Weise unterscheidet, wie es sich in den Himmel bewegt. Einige Aliens schweben in einer geraden Linie nach oben, während andere sich in einer Korkenzieherbewegung aufwärts bewegen oder sich um ihre eigene Achse drehen. Sobald ein Alien erfolgreich eingesaugt wurde, verschwindet es aus der Spielwelt.

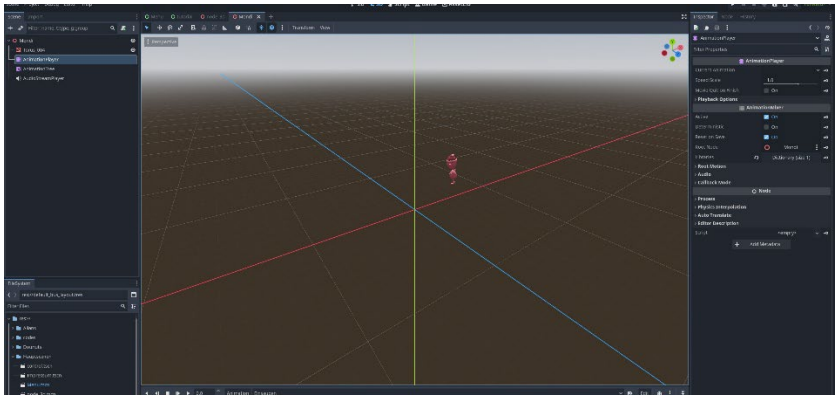


Abbildung 36: Alien Animation im Kreis laufen

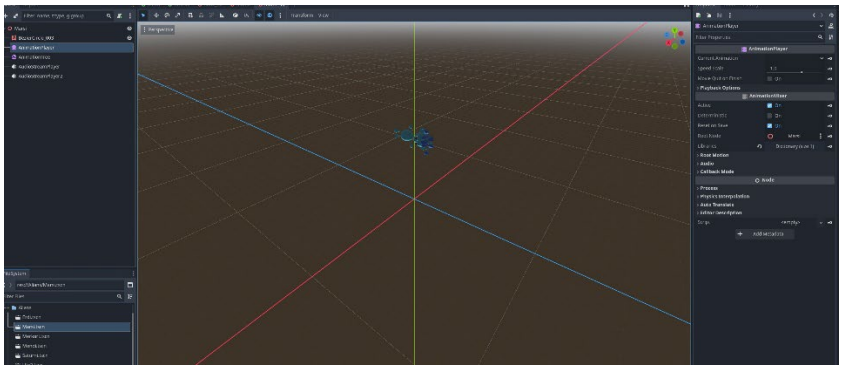


Abbildung 37: Alien Animation hochbeamen

Die Animationen für die Bewegung und das Einsammeln der Aliens werden in Blender erstellt. Von dort aus können sie exportiert und in Godot importiert werden. Beim Import bestehen die Aliens aus einem Haupt-Node, einem Mesh-Node, der den Körper des Aliens darstellt, und einem AnimationsPlayer-Node. Der AnimationsPlayer speichert alle Animationen, die für das Alien notwendig sind.
Um diese Animationen in Godot zu verwenden, wird ein AnimationTree-Node benötigt. Der AnimationTree erlaubt eine hierarchische Struktur, in der Animationen als Knotenpunkte platziert und miteinander verbunden werden können. Im Fall der Aliens wird die Laufanimation zunächst mit der Einsaug-Animation verbunden. Diese Verbindung wird durch Bedingungen gesteuert: Wenn der Beam in Kontakt mit dem Alien ist, wird die Beam-Animation aktiviert. Wenn der Kontakt unterbrochen wird, kehrt die Animation zur Laufanimation zurück.

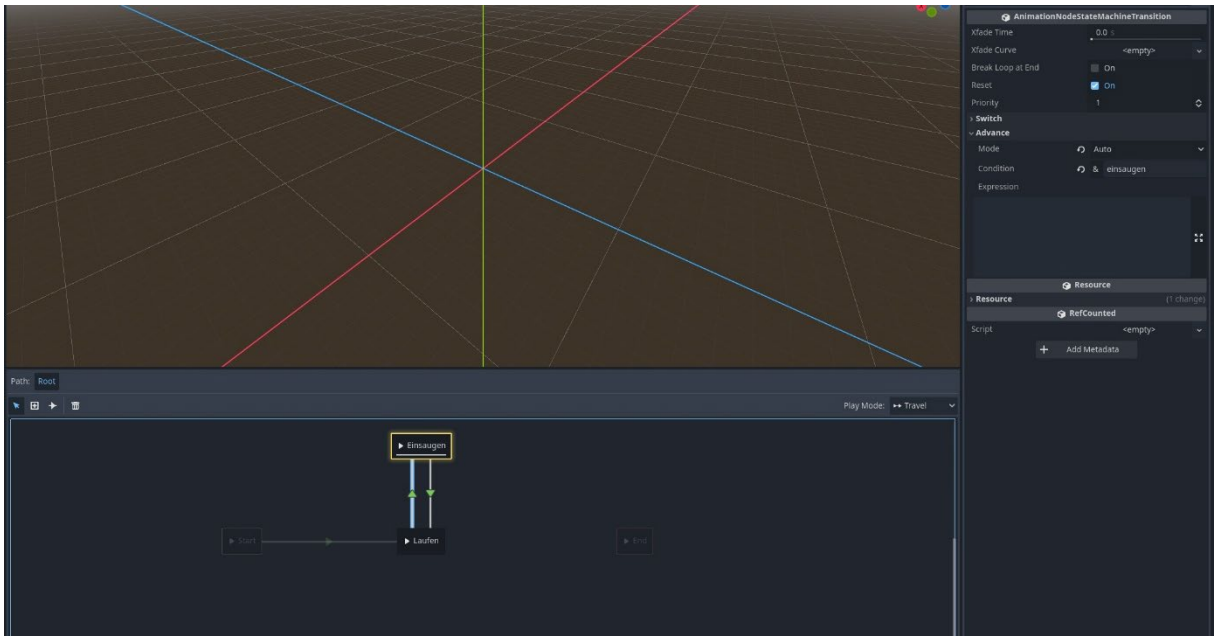


Abbildung 38: Animationtree in Godot

Diese Bedingungen werden im Code überprüft und gesteuert. Beispielsweise wird im Code festgelegt: Wenn der Beam in Kontakt mit dem Alien ist, wird zur Beam-Animation gewechselt. Wenn der Kontakt endet, wird die Laufanimation wieder aktiviert. Die Laufanimation ist dabei so eingestellt, dass sie in einer Endlosschleife läuft – wenn sie endet, beginnt sie automatisch wieder von vorne.
Die Einsammel-Animation (die Einsaug-Animation) dauert exakt 2 Sekunden, was die vorgegebene Zeit ist, die benötigt wird, um ein Alien einzusaugen. Diese präzise Zeitsteuerung stellt sicher, dass das Einsaugen des Aliens immer gleich abläuft und das Spielerlebnis konsistent bleibt.

Durch das Drücken einer bestimmten Taste(L1 oder R1 wird) ein Beam aktiviert, der die Fähigkeit hat, Aliens einzusaugen. Um den Effekt visuell ansprechender zu gestalten, wurde eine Animation integriert: Der Beam besteht aus fünf kleinen Ringen, die sich nach oben bewegen und dabei ihre Größe verändern.

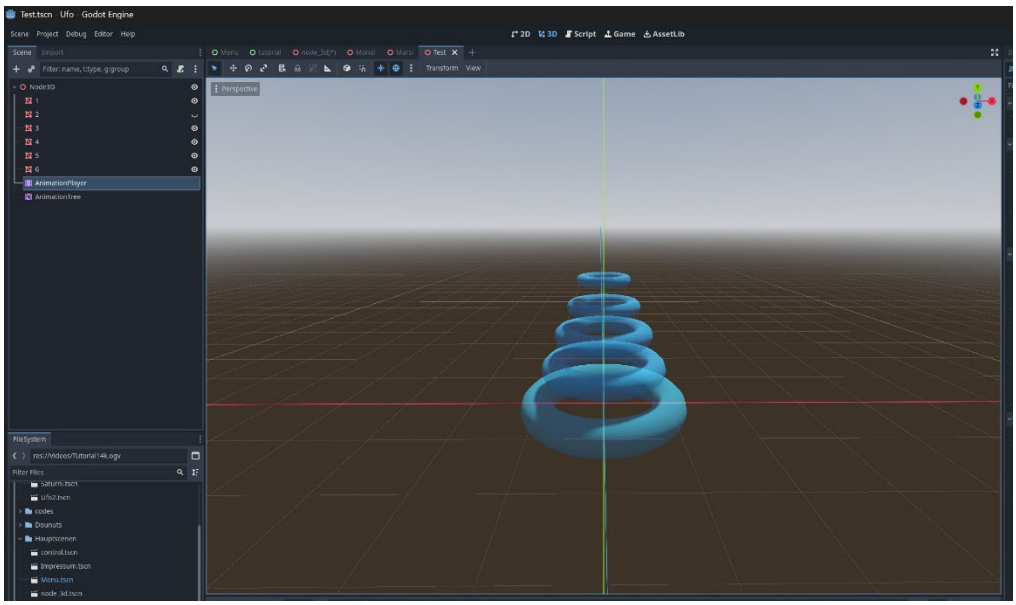


Abbildung 39: Beam in Godot

Zusätzlich wird ein haptisches Signal: Sobald der Beam aktiviert wird, vibriert der Controller leicht. Wenn ein Alien erfolgreich eingesaugt wird, verstärkt sich die Vibration, um dem Spieler ein direktes Feedback zu geben und das Einsammeln zu signalisieren.

Der gesamte Einsammelmechanismus wird vollständig vom Code gesteuert. Dieser Mechanismus ist das Herzstück des Spiels, da er der Auslöser viele verschiedene wichtige Signale ist. Um das Einsaugen korrekt durchzuführen, werden zwei Skripte benötigt: Eines befindet sich am Spieler, das andere an den Aliens. Warum zwei Skripte? Dafür ist eine kurze Erklärung nötig:

Am Spieler-Node befindet sich der Beam, zusammen mit einem Mesh und einer Kollisions-Shape. Damit zwei Objekte miteinander interagieren können, müssen sie über Kollisions-Shape-Komponenten verfügen. Der Collider des Beams interagiert mit einem zweiten Collider, der an den Aliens angebracht ist. Die Aliens umkreisen einen festen Punkt, an dem ihr Collider positioniert ist. Diese beiden Collider sollen jedoch nur miteinander interagieren, wenn der Beam tatsächlich aktiv ist.

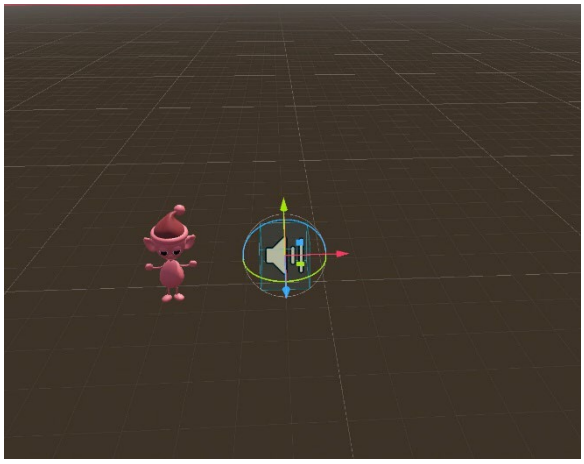


Abbildung 40: Alien und deren Collisionsshape

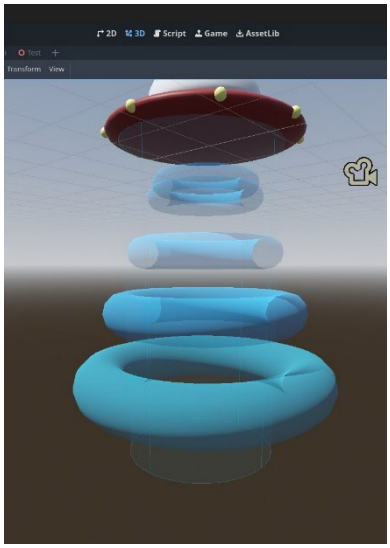
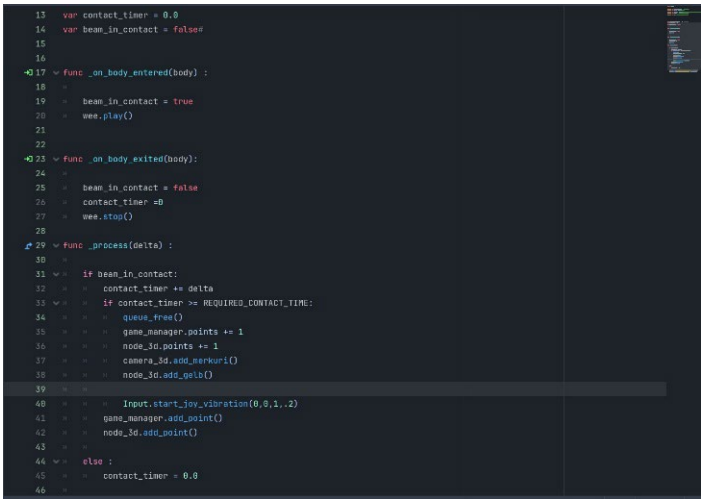


Abbildung 41: Beam und deren Collisionsshape

Dieses Problem wird durch die Verwendung von Kollisionsmasken gelöst, die eine Art Ebenensystem darstellen. Das bedeutet, dass zwei Collider nur dann miteinander

interagieren können, wenn sie sich auf der gleichen Ebene befinden. Der Alien-Collider bleibt immer auf der Ebene 1, während der Beam-Collider standardmäßig auf Ebene 2 positioniert ist. Wenn der Beam nicht aktiviert ist und der Spieler einfach über das Alien hinwegfliegt, passiert nichts. Wird jedoch der Beam aktiviert, wird der Collider des Beams in das Skript des Spielers geändert, sodass dieser auf Ebene 1 wechselt. Jetzt können die beiden Collider miteinander interagieren.

Nun zum Skript der Aliens: Die Skripte der Aliens sind grundsätzlich gleich, mit wenigen Unterschieden. Im Skript gibt es zwei Variablen und eine Konstante. Die Konstante stellt die Zeit dar, die für das Einsaugen des Aliens erforderlich ist (in diesem Fall 2 Sekunden). Die erste Variable ist die Zeit (initial auf null gesetzt), die die Dauer des Kontakts misst. Die zweite Variable ist beam_in_contact, die standardmäßig auf false gesetzt ist und anzeigt, dass der Beam zunächst nicht mit dem Alien in Kontakt steht.



```
13  var contact_timer = 0.0
14  var beam_in_contact = false#
15
16
17  func _on_body_entered(body) :
18      
19      beam_in_contact = true
20      wee.play()
21
22
23  func _on_body_exited(body):
24      
25      beam_in_contact = false
26      contact_timer =0
27      wee.stop()
28
29  func _process(delta) :
30      
31      if beam_in_contact:
32          contact_timer += delta
33          if contact_timer >= REQUIRED_CONTACT_TIME:
34              queue_free()
35              game_manager.points += 1
36              node_3d.points += 1
37              camera_3d.add_merkuri()
38              node_3d.add_gelb()
39              
40              Input.start_joy_vibration(0,0,1,.2)
41          game_manager.add_point()
42          node_3d.add_point()
43          
44      else :
45          contact_timer = 0.0
46
```

Abbildung 42: Skriptbeispiel des Einsaugens

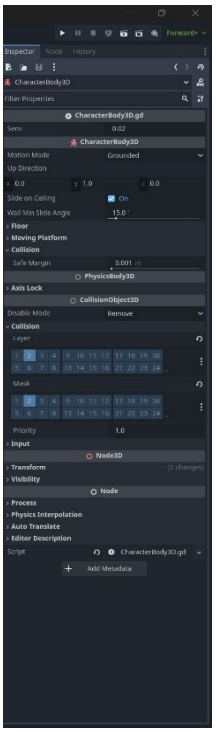


Abbildung 43: Collisionlayer

Wenn der Beam aktiviert ist und mit dem Collider des Aliens in Kontakt kommt, wird ein Signal an das Skript gesendet, dass etwas den Collider betreten hat. Sobald das passiert, wird die Variable beam_in_contact auf true gesetzt und startet ein Timer, der die Zeit zählt. Sollte der Beam dann für 2 Sekunden oder länger in Kontakt mit dem Alien bleiben, verschwindet das Alien.
Falls der Beam den Kontakt verliert – etwa indem er den Collider des Aliens verlässt oder nicht mehr aktiviert ist – wird ein Signal vom Alien-Collider gesendet, dass das Objekt den Kollisionsbereich verlassen hat. In diesem Fall wird die Variable beam_in_contact wieder auf false gesetzt, und der Timer wird zurückgesetzt und der, Einsaugprozess wird gestoppt

Nach dem Einsammeln eines Aliens verändert sich die angezeigte 3D-Welt. Zur Auswahl stehen dabei verschiedene Varianten: eine historische Karte von 1817, eine aktuelle Luftbildkarte von 2024, eine Karte von 1945 und 1982 (DOP), sowie eine topografische Karte (TK25).
Die technische Umsetzung dahinter ist vergleichsweise simpel: Alle Kartenmodelle werden in die Hauptszene geladen und exakt übereinander positioniert. Dabei ist entscheidend, dass alle Modelle die gleiche Größe und den gleichen Ursprungspunkt haben. So liegen alle sechs Karten exakt übereinander im Raum.

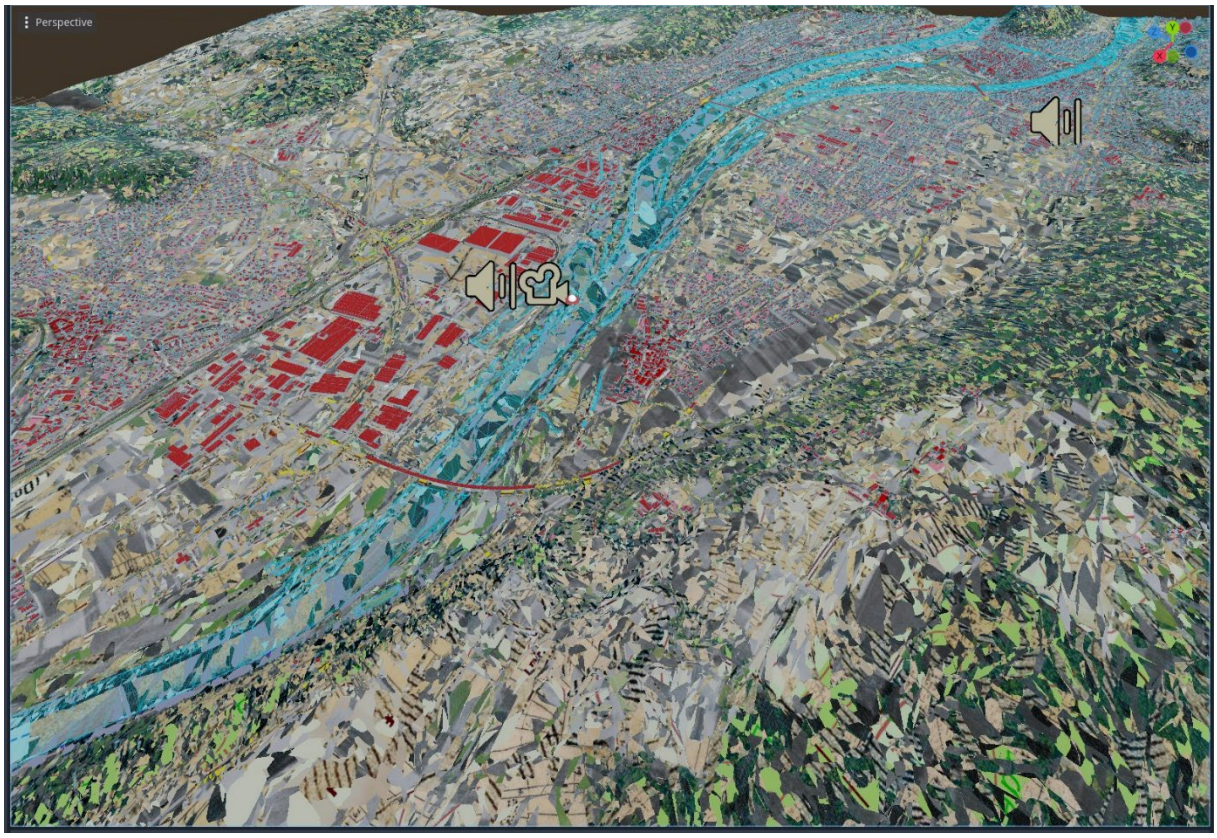


Abbildung 44: Alle Karten übereinander in Godot

Eine zentrale Rolle spielt die Kamera beim Hauptcharakter. Godot arbeitet mit sogenannten Sichtmasken (Visibility Layers). Diese Sichtmasken sind Ebenen, auf denen Objekte angezeigt werden. Jede Kamera in Godot zeigt nur die Objekte an, die sich auf einer Sichtmaske befinden, die für diese Kamera aktiviert wurde.
Die Kartenmodelle wurden so vorbereitet, dass jedes Modell auf einer eigenen Sichtmaske liegt. Das bedeutet: Pro eingesammeltem Alien wird ein Signal ausgesendet, welches die Kamera veranlasst, auf eine andere Sichtmaske zu wechseln. Somit kann die Darstellung der aktiven Karte gezielt verändert werden.
Wichtig dabei: Jedes Alien schickt beim Einsammeln ein individuelles Signal. Dadurch ist die Reihenfolge der Kartenveränderung beliebig – der Spieler kann die Welt also in einer nicht-linearen Reihenfolge aufdecken. Zu Beginn zeigt die Kamera standardmäßig die aktuelle DOP-Karte von 2024 an.

Die Aliens geben akustische Hinweise auf ihre Position. Je näher man ihnen kommt, desto lauter werden die Geräusche – so erhält der Spieler eine zusätzliche Orientierungshilfe. Gleichzeitig ertönt beim erfolgreichen Einsammeln ein markanter

Einsaug-Sound, der dem Spieler akustisch signalisiert, dass ein Alien aufgenommen wurde.

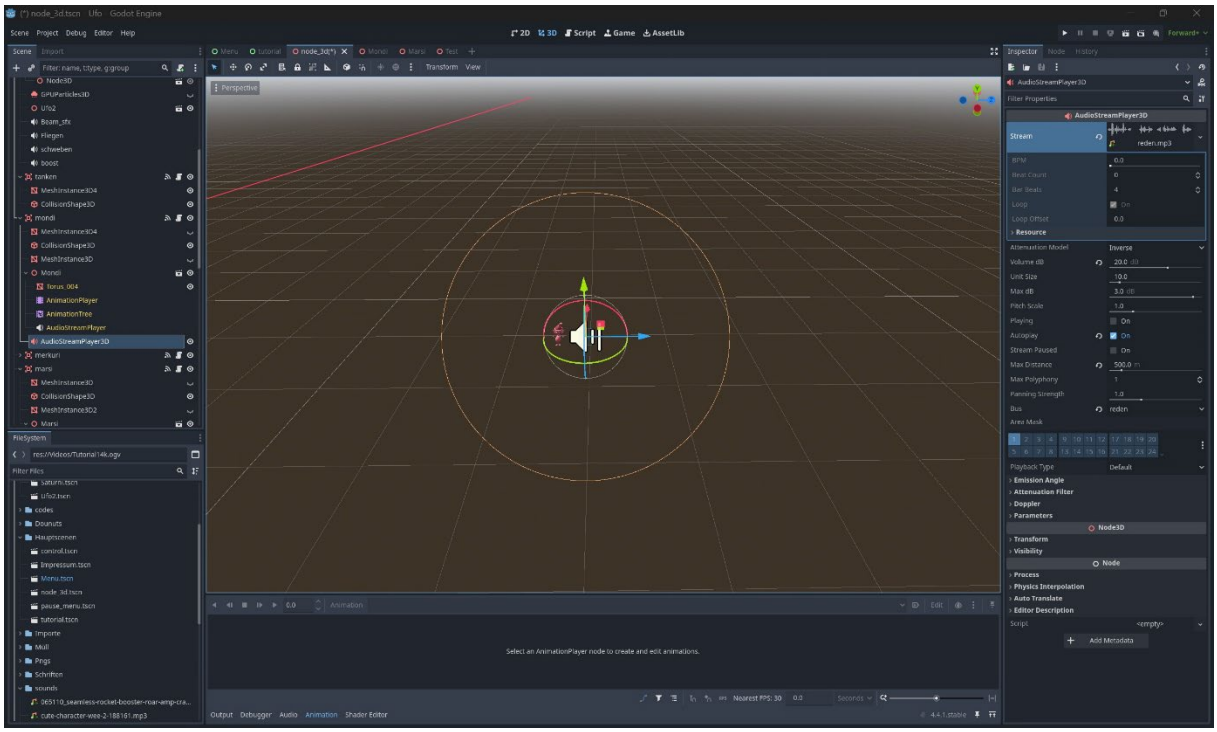


Abbildung 45: Soundsphäre

Technisch umgesetzt wird dies mit Hilfe von Sound-Nodes und ein wenig Skript. Für die Umgebungsgeräusche der Aliens wird der Node AudioStreamPlayer3D verwendet. Das „3D“ steht hierbei für räumlichen Klang: Wenn sich ein Alien beispielsweise links vom Spieler befindet, hört man den Ton ausschließlich auf der linken Seite – bei Nutzung von Kopfhörern sorgt das für ein immersives Erlebnis.

Dem AudioStreamPlayer3D wird eine Audiodatei im MP3-Format zugewiesen – in diesem Fall ein leises Murmeln oder ein rufähnliches Geräusch. Zusätzlich wird ein Radius definiert, in dem der Ton hörbar sein soll. Der Mittelpunkt dieses Bereichs ist stets das jeweilige Alien. Das Audiostream wird geloopt, das heißt: Der Sound wiederholt sich dauerhaft.

Damit die Umgebungsgeräusche nicht während des Einsaugens stören, wird der Player im Code während des Einsaugens stummgeschaltet. Sobald der Beam das Alien berührt, wird der AudioStreamPlayer3D gestoppt. Wenn der Kontakt wieder abbricht – etwa weil der Beam deaktiviert wird – startet das Geräusch erneut.

Zusätzlich gibt das Alien beim Einsaugen einen kurzen Ruf von sich. Dafür wird ein gewöhnlicher AudioStreamPlayer (ohne 3D) verwendet. Auch hier wird eine MP3-Datei zugewiesen. Im Code wird festgelegt, dass dieser Sound exakt entgegengesetzt

zur Umgebungsgeräusch-Logik abgespielt wird: Sobald der Beam in Kontakt mit dem Alien ist, startet der Schrei – bricht der Kontakt ab, stoppt auch dieser Sound.

Mit der Boost-Taste ermöglicht es dem Spieler doppelt so schnell fortzubewegen, wie mit der normalen Geschwindigkeit. Zusätzlich erscheinen kleine Partikel am Ufo, die den Effekt visuell stärker zu verdeutlichen. Der Boost hat außerdem einen eigenen Soundeffekt, um dem Spieler direkt ein akustisches Feedback zu geben, dass die Geschwindigkeit erhöht wurde.

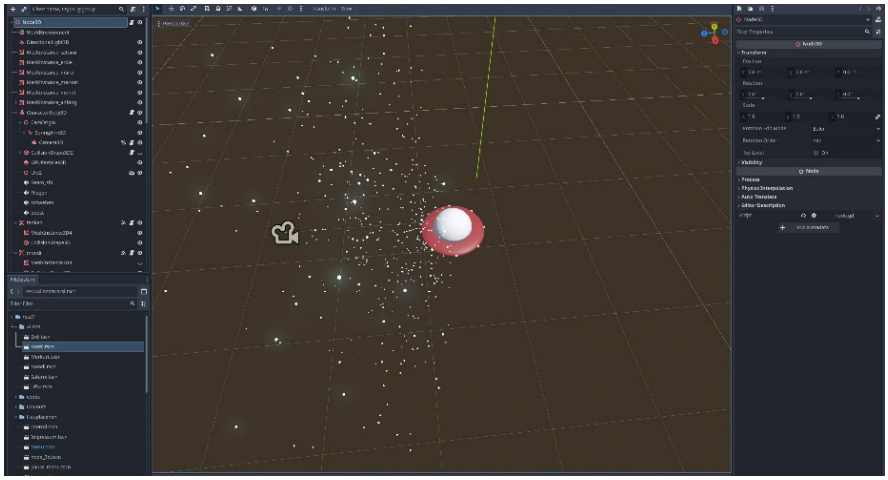


Abbildung 46: Boostpartikel

Der Boost wird im Skript ganz einfach getriggert. Sobald die Boost-Taste drückt wird, wird im Code die variable Geschwindigkeit verdoppelt bzw. beim Loslassen der Boost-Taste reduziert sich die Bewegung auf die die Ausgangsgeschwindigkeit.

Auf der rechten Seite des Bildschirms befindet sich eine vertikale Leiste, auf der alle Aliens als ausgegraute Icons dargestellt sind. Sobald ein Alien eingesammelt wurde, wird das entsprechende Icon farbig. Diese Leiste dient dem Spieler als visuelle Checkliste, um jederzeit zu sehen, welche Aliens bereits eingesammelt wurden und welche noch fehlen.
Die Icons bestehen aus PNG-Dateien – jeweils einmal in Graustufen und einmal in Originalfarbe. In der Benutzeroberfläche werden sie übereinandergelegt, wobei zu Beginn nur das graue Icon sichtbar ist. Das farbige Icon ist zunächst ausgeblendet.
Die farbige Wechsel erfolgt über ein Signal, das beim Einsammeln eines Aliens ausgesendet wird. Dieses Signal wird an die Einsammelleiste weitergeleitet. Dort wird es eine Funktion aufgerufen, die das graue Icon ausblendet und das farbige Icon

sichtbar macht. So entsteht der Eindruck, dass das eingesammelte Alien „aktiviert“ wurde.

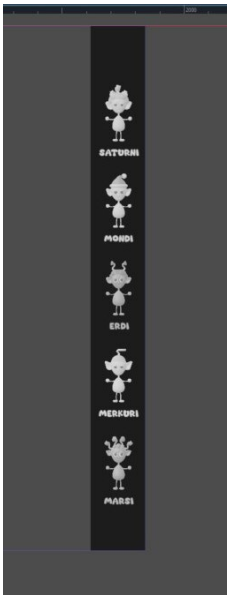


Abbildung 47: Uneingesammelte Aliens

Abbildung 48: Eingesammelte Aliens

Das Spiel wird von einer atmosphärischen Hintergrundmusik begleitet, die zur Stimmung beiträgt, ohne dabei aufdringlich zu wirken. Sie ist bewusst leiser abgemischt, damit wichtige Spielgeräusche – wie etwa Hinweise von den Aliens oder der Beam-Sound – gut hörbar bleiben.

Technisch wird die Musik über einen einzelnen AudioStreamPlayer-Node realisiert. Die gewünschte Musikdatei wird im MP3-Format in diesen Node eingefügt und auf Loop gesetzt, sodass sie durchgehend im Hintergrund abgespielt wird.
Der Start der Musik erfolgt direkt zu Beginn des Spiels, in der ready()-Funktion. Dort wird einfach der Befehl zum Abspielen der Hintergrundmusik aufgerufen.

3.5.5 Darstellung der Geodaten & Interaktion

Im Spiel werden insgesamt sechs verschiedene Kartendarstellungen präsentiert, die gezielt ausgewählt wurden, um eine möglichst große Bandbreite an Geodatenformaten des LDBV abzubilden. Ziel ist es, den Spielenden die Vielfalt und den Wandel kartografischer Darstellungen näherzubringen – sowohl in Bezug auf technische Präzision als auch auf visuelle Ausdrucksformen.
Die Karten reichen dabei von aktuellen digitalen Orthophotos über Geländemodelle bis hin zu historischen Kartenansichten. Durch diesen bewussten Kontrast zwischen modernen und älteren Darstellungen sollen insbesondere die Unterschiede in Genauigkeit, Informationsdichte und Darstellungsmethoden hervorgehoben werden. So wird zum Beispiel deutlich, wie sich die Vermessungstechniken, Maßstäbe und Kartografie im Laufe der Zeit verändert haben.
Die Interaktion mit den Karten erfolgt eingebettet in das Spielgeschehen: Die Spielenden bewegen sich aktiv durch die virtuellen Räume, erkunden die Landschaft aus der Third-Person-Perspektive und interagieren dabei mit den dargestellten Geodaten. Durch diese Bewegung entsteht ein räumliches Verständnis für Höhenunterschiede, Bebauung und Geländeformen – besonders anschaulich bei der Nutzung des digitalen Geländemodells (DGM) und der LoD2-Gebäudedaten.
Die Kombination aus spielerischer Endecken und immersiver Darstellung unterstützt nicht nur das Lernen, sondern stärkt auch die visuelle Unterscheidungskraft der Nutzerinnen und Nutzer. Sie lernen, Karten nicht nur zu betrachten, sondern auch zu interpretieren – und das eingebettet in eine motivierende, interaktive Umgebung.

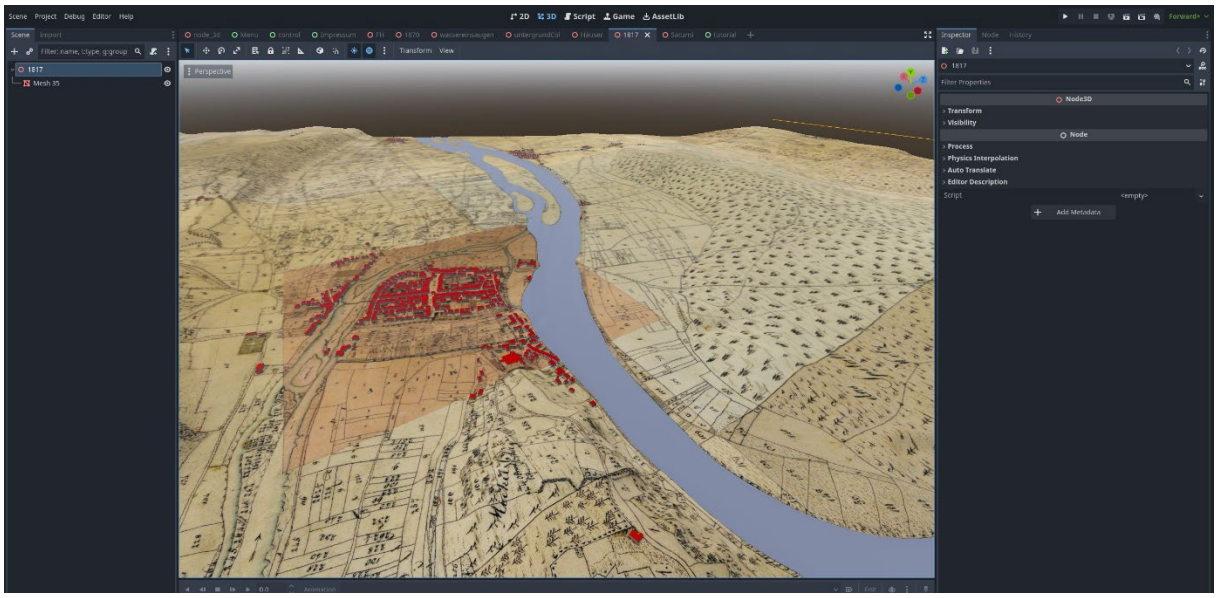


Abbildung 49: Uraufnahme von 1817 in Godot

3.5.6 Technische Herausforderungen & Lösungen

Erstes Nutzer Feedback

Im Rahmen eines ersten Nutzerfeedbacks wurde eine frühe Spielversion mit einer kleinen Gruppe von Kindern getestet. Die Teilnehmenden waren zwischen der ersten und vierten Klasse. Zu diesem Zeitpunkt befand sich das Spiel noch in einer sehr frühen Phase: Es wurde lediglich eine einzelne Karte verwendet, auf der digitale Orthophotos (DOPs), digitale Geländemodelle (DGMs) sowie 3D-Gebäudedaten (LoD2) integriert waren.

Ziel dieses ersten Tests war es vor allem, grundlegende Interaktionen zu beobachten – insbesondere, wie gut die Kinder mit der Steuerung zurechtkommen, ob sie den Sammelmechanismus verstehen und ob das Spielprinzip grundsätzlich Freude bereitet. Bereits während der Tests zeigten sich deutliche Unterschiede in der Bedienung: Kinder mit vorheriger Erfahrung in digitalen Spielen konnten sich deutlich schneller auf die Steuerung einstellen. Auch das Alter spielte eine wichtige Rolle: Während jüngere Kinder (z. B. aus der ersten Klasse) anfangs noch Schwierigkeiten hatten, lernten ältere Kinder meist rascher, mit der Steuerung umzugehen.

Zwei Erkenntnisse traten im Testverlauf besonders deutlich hervor. Zum einen stellte sich die gewählte Third-Person-Perspektive als potenzielle Herausforderung heraus. Zwar ermöglicht diese Perspektive eine gute Übersicht, gleichzeitig führte sie aber bei

vielen Kindern zu Problemen bei der räumlichen Orientierung – insbesondere bei der Einschätzung von Entfernungen im 3D-Raum. So kam es häufig vor, dass Spielerinnen und Spieler dachten, sie würden direkt über einem Alien schweben, obwohl sie in Wahrheit daran vorbeiflogen. Dieses Missverständnis lag unter anderem daran, dass viele Kinder die freie Kamerasteuerung nicht aktiv nutzten oder sogar vergaßen, dass sie diese Funktion überhaupt zur Verfügung hatten. Eine mögliche Lösung, die daraus abgeleitet wurde, ist die leicht versetzte Positionierung der Kamera weiter vom UFO weg, um die räumliche Lage des Spielobjekts klarer erkennbar zu machen.

Ein zweiter wichtiger Punkt betraf die audiovisuelle Rückmeldung beim Einsammeln der Aliens. Sobald ein Spieler mit dem Strahl („Beam“) in die Nähe eines Aliens gelangte, wurde eine Animation und ein Soundeffekt ausgelöst, die signalisierten, dass sich das Alien im Einfangbereich befindet. Schnell zeigte sich, dass die Kinder auf diese Hinweise reagierten, daraus lernten und gezielter versuchten, weitere Aliens zu lokalisieren. Diese Rückmeldemechanik erwies sich somit als wichtiges didaktisches Element zur Förderung der Exploration und Interaktion.

Ein weiteres Problem betraf den Sammelmechanismus selbst. Zu Beginn war das Einsammeln erschwert, da die sogenannte Collision Shape – also der Bereich, der physikalisch erkannt wird – bei den Aliens zu klein definiert war. Dieses Problem ließ sich jedoch relativ einfach beheben, indem die Form vergrößert wurde, wodurch die Spielmechanik deutlich intuitiver und zugänglicher wurde.

Insgesamt lieferte das erste Nutzerfeedback wertvolle Einblicke in das Spielverhalten der Zielgruppe. Es zeigte auf, wo noch Optimierungspotenzial besteht, aber auch, welche spielerischen Elemente bereits gut funktionieren und zur Motivation beitragen.

4 Einbindung von Serious-Gaming-Elementen

4.1 Pädagogische Prinzipien in der Spielmechanik

Die Spielmechaniken des entwickelten Serious Games orientieren sich gezielt an pädagogischen Prinzipien, um Lernprozesse spielerisch zu fördern. Dabei werden nicht nur Fakten vermittelt, sondern auch grundlegende Kompetenzen im Umgang mit Raum, Karten und digitalen Geodaten aufgebaut.

1. Räumliches Denken und Orientierung im dreidimensionalen Raum

Eine zentrale spielerische Herausforderung besteht darin, dass sich die Spielenden frei in einer 3D-Welt bewegen. Um Aufgaben zu lösen und Objekte zu finden, müssen sie lernen, sich in einem dreidimensionalen Raum zu orientieren, Entfernungen einzuschätzen und gezielt zu manövrieren. Dies schult nicht nur das visuell-räumliche Vorstellungsvermögen, sondern fördert auch die kognitive Fähigkeit zur Navigation in virtuellen Umgebungen – eine Kompetenz, die zunehmend auch in realweltlichen Kontexten (z. B. bei der Nutzung von digitalen Karten, Navigation oder in technischen Berufen) relevant ist.

2. Kartenverständnis und topografisches Lernen

Durch die Einbindung realer Geodaten (z. B. digitale Orthophotos, Höhenmodelle oder historische Karten) werden die Spielenden gezielt mit unterschiedlichen Kartenarten konfrontiert. Sie lernen Unterschiede zwischen alten und neuen Kartendarstellungen kennen, erhalten Einblick in die Entwicklung der Kartografie und erfahren, wie sich Landschaften über die Zeit verändert haben. Über die Spielmechanik des Vergleichs (z. B. durch Überlagerung oder Wechsel von Kartenlayern) wird das kritische Betrachten von Karteninhalten gefördert.

3. Exploratives Lernen und intrinsische Motivation

Das Spiel setzt auf Erkundung als zentrales Element. Spielerinnen und Spieler werden nicht durch lineare Aufgaben geführt, sondern sollen die Umgebung aktiv erforschen. Dies unterstützt ein exploratives, selbstgesteuertes Lernen, bei dem Neugier und Entdeckerdrang geweckt werden. Das Prinzip der „intrinsischen Integration“ kommt hier zum Tragen: Lerninhalte sind nicht losgelöst, sondern in die Spielmechaniken eingebettet.

4. Sammel- und Interaktionsmechaniken mit Feedback

Durch das Einsammeln bestimmter Objekte (z. B. „Aliens“, die bestimmte Daten symbolisieren) wird das Prinzip der Belohnung und Rückmeldung umgesetzt. Die Spielenden erhalten direktes audiovisuelles Feedback, wenn sie erfolgreich agieren.

Diese Mechanik fördert nicht nur die Motivation, sondern auch das Verständnis dafür, wann eine Aktion erfolgreich ist – ein grundlegendes Prinzip effektiven Lernens.

5. Lokales Wissen und Identifikation mit der Region

Durch die gezielte Auswahl bekannter Orte (wie der Befreiungshalle oder dem Kelheimer Hafen) wird eine emotionale Verbindung zur dargestellten Region hergestellt. Spielerinnen und Spieler, insbesondere aus der Umgebung, können so lokale Bezüge erkennen und sich mit dem Lernstoff stärker identifizieren. Das fördert die Kontextualisierung und macht das Gelernte greifbarer.

4.2 Motivation & Interaktivität für Kinder

Ein zentrales Ziel bei der Gestaltung des Spiels war es, eine Spielumgebung zu schaffen, die Kinder motiviert, neugierig macht und gleichzeitig Raum für eigene Entdeckungen bietet. Die Spielerinnen und Spieler können sich frei in einem dreidimensionalen Raum bewegen, ohne durch lineare Abläufe oder starre Regeln eingeschränkt zu sein. Diese Bewegungsfreiheit fördert nicht nur das individuelle Erkundungsverhalten, sondern vermittelt auch ein Gefühl von Selbstwirksamkeit: Die Kinder entscheiden selbst, in welcher Reihenfolge sie die Aufgaben – konkret: das Einsammeln der Aliens – lösen wollen.

Jedes der Aliens (im Spiel „Alinas“ genannt) ist bewusst unterschiedlich gestaltet, um den Sammelprozess abwechslungsreich zu halten und ein Gefühl von Neugierde und Überraschung zu erzeugen. Dadurch entsteht ein spielerisches Ziel, das über reines „Abarbeiten“ hinausgeht: Die Kinder sind motiviert, alle Alinas zu finden und zu entdecken, was sich hinter jeder Begegnung verbirgt. Das stärkt den Entdeckergeist und sorgt dafür, dass sie das Spiel als lohnend und spannend erleben.

Zusätzlich wird durch die klare Aufgabenstellung – alle Aliens einzusammeln – ein fester Rahmen gegeben, der den Kindern Orientierung bietet. Gleichzeitig bleibt genug Freiheit, um das Spiel individuell zu erleben. Dieses Wechselspiel aus Zielorientierung und Freiheit schafft eine motivierende Lernumgebung, die sich an den Interessen und dem natürlichen Spielverhalten von Kindern orientiert.

5 Diskussion und Ausblick

Im Rahmen des Projekts wurde erfolgreich ein kleines Spiel entwickelt, das Serious-Gaming-Elemente nutzt, um Wissen über Geodaten auf unterhaltsame Weise zu vermitteln. Ziel war es, ein Spiel zu gestalten, das nicht überfordert, leicht zugänglich ist und insbesondere durch eine intuitive Steuerung auch von jüngeren Nutzern problemlos bedient werden kann. Der Fokus lag dabei nicht auf der Komplexität des Gameplays, sondern auf einer Kombination aus Lerninhalten und Spielspaß. Zudem gab das Landesamt die Vorgabe, dass das Spiel nicht länger als 5 Minuten dauern soll, da es auf Messen und Ausstellungen für interessierte Kinder und Jugendliche präsentiert werden soll.

Die technische Umsetzung stellte eine große Herausforderung dar. Besonders der Einstieg in die Game Engine Godot war anspruchsvoll, da die Engine vergleichsweise jung ist und die Community entsprechend kleiner als bei etablierten Engines wie Unity oder Unreal. Viele spezifische Lösungen oder Tutorials sind entweder nicht vorhanden oder schwer auffindbar. Dadurch war es oft notwendig, eigene Ansätze zu entwickeln, Probleme individuell zu analysieren und kreative Lösungen zu finden. Diese Hürde war jedoch zugleich eine wertvolle Lernerfahrung, die das Verständnis für Game-Development insgesamt deutlich vertieft hat.

In Bezug auf die didaktische Wirksamkeit lässt sich sagen: Das Spiel wurde so konzipiert, dass es spielerisch geografisches Wissen vermittelt und Interesse an Geodaten weckt. Ob und wie nachhaltig diese Vermittlung gelingt, wird sich erst im Laufe der Zeit zeigen – etwa durch Rückmeldungen und Beobachtungen im praktischen Einsatz. Ziel ist es, dass Kinder entweder konkrete Informationen mitnehmen oder zumindest eine positive, neugierige Erinnerung an die Themen Geografie und Raumwahrnehmung entwickeln.

Für die Zukunft eröffnen sich vielfältige Möglichkeiten der Weiterentwicklung. Das Projekt demonstriert, welches Potenzial in der Verknüpfung von Geodaten und Game Design steckt – und kratzt dabei nur an der Oberfläche. Mit Game Engines lassen sich theoretisch unendlich viele Spielideen umsetzen. Denkbar wären neue Varianten des Spiels mit anderen Karten, neuen Sammelobjekten oder auch anderen geobezogenen Themen. Auch die aktuelle Version kann modular erweitert werden – etwa durch neue

Levels, zusätzliche Aliens oder alternative Umgebungen. So könnte das Projekt langfristig wachsen oder als Vorlage für andere Bildungsanwendungen dienen.

Insgesamt zeigt dieses Projekt eindrucksvoll, wie durch den kreativen Einsatz von Geodaten in Kombination mit spielerischen Elementen ein Lernangebot entstehen kann, das nicht nur Wissen vermittelt, sondern auch Neugier, Entdeckerfreude und einen persönlichen Bezug zur realen Umgebung fördert.

6 Literaturverzeichnis

- Arthur, P. & Passini, R. (1992). Wayfinding: People, signs, and architecture. New York: McGraw-Hill.
- Atzl, C., Andorfer, M. & Mittlböck, M. (2023). Geodatenintegration in 3D-Spielewelten: Gamified Campus Science City Itzling. *AGIT – Journal für Angewandte Geoinformatik*, 2023(9), S. 2-11.
- Emmerich, K., & Bockholt, M. (2016). Serious games evaluation: Processes, models, and concepts. In R. Dörner, S. Göbel, M. Kickmeier-Rust et al. (Eds.), Entertainment Computing and Serious Games (S. 265-283). Cham: Springer.
- Fritsch, D. (2003). 3D building visualisation – outdoor and indoor applications. *Photogrammetric Week,* 03, S. 281-290.
- Girard, C., Ecalle, J. & Magnan, A. (2013). Serious games as new educational tools: How effective are they? A meta-analysis of recent studies. Journal of *Computer Assisted Learning*, 29(3), S. 207–219.
- Kerres, M., Bormann, M. & Vervenne, M. (2009). Didaktische Konzeption von Serious Games: Zur Verknüpfung von Spiel- und Lernangeboten. *Zeitschrift für Theorie und Praxis der Medienbildung,* 2009, S. 1-16.
- Laamarti, F., Eid, M. & El Saddik, A. (2014). An overview of serious games. *International Journal of Computer Games Technology*, 2014(3), 1-15.
- Lewis, M. & Jacobson, J. (2002). Game engines in scientific research. *Communications of the ACM*, 45(1), S. 27-31.
- Loh, C. S., Sheng, Y. & Ifenthaler, D. (2015). Serious games analytics: Theoretical framework. In C. S. Loh, Y. Sheng & D. Ifenthaler (Eds.), Serious games analytics (S. 3-29). Cham: Springer.
- Novak, J. (2012). Game development essentials: An introduction (3rd ed.). Delmar: Cengage Learning.
- Salen, K. & Zimmerman, E. (2004). Rules of play: Game design fundamentals. Cambridge: MIT Press.

Internetquellen

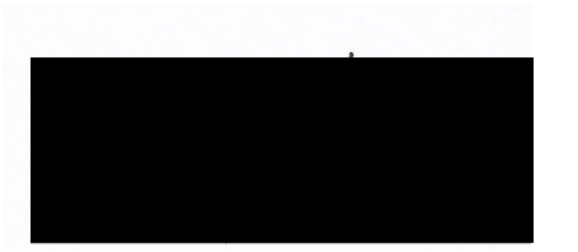
- Adobe (2025). Adobe Character Animator – Übersicht. https://helpx.adobe.com/de/adobe-character-animator/using/overview.html (Abgerufen am 07.05.2025).
- Adobe (2025). Adobe Photoshop. https://www.adobe.com/de/products/photoshop.html (Abgerufen am 07.05.2025).
- Animations and More (2025). Lexikon – Game Engine. https://www.animations-and-more.com/lexikon-game-engine/ (Abgerufen am 07.05.2025).
- Autodesk (2025). InfraWorks. https://www.autodesk.com/de/products/infraworks/overview (Abgerufen am 07.05.2025).
- Bayerische Verwaltung der staatlichen Schlösser, Gärten und Seen. (2024). Befreiungshalle Kelheim. https://www.befreiungshalle.org/ (Abgerufen am 07.05.2025).
- Blender Foundation (o.D.). Blender – Open Source 3D Creation. https://www.blender.org/ (Abgerufen am 07.05.2025).
- Fels-Werke GmbH (o.D.). FELS – Rohstoffe für Generationen. https://www.fels.de/de/ (Abgerufen am 07.05.2025).
- Peter-Jan, Giant Sloth Games (19.04.2023). So you want to make a Game Engine!? (WATCH THIS before you start). https://www.youtube.com/watch?v=3rcka6P2cVI&t=417s (Abgerufen am 07.05.2025).
- Godot Engine (2025). Godot Engine. https://godotengine.org/ (Abgerufen am 07.05.2025).
- Godot Engine Documentation (2014). Importing Scenes – Assets Pipeline. https://docs.godotengine.org/en/3.5/tutorials/assets_pipeline/importing_scenes.html (Abgerufen am 07.05.2025).
- Godot Foundation (2025). Godot Foundation – Förderung der Engine-Entwicklung. https://godot.foundation/ (Abgerufen am 07.05.2025).
- Hafen Kelheim GmbH. (2025). Hafen Kelheim – Bayernhafen. https://www.hafen-kelheim.de/ (Abgerufen am 07.05.2025).

- Landkreis Kelheim (2019). Wirtschaftsstandort Kelheim – Fakten und Daten. https://wirtschaft.kelheim.de/wirtschaftsstandort-fakten/ (Abgerufen am 07.05.2025).
- Satzhüterin, P. (17.02.2020). Was sind Open World Games? Bücherstadt Magazin. https://buecherstadtmagazin.de/was-sind-open-world-games/ (Abgerufen am 07.05.2025).
- Schlossverwaltung Bayern (o.D.). Schlösser in Kelheim – Bayerische Verwaltung. https://www.schloesser.bayern.de/deutsch/schloss/objekte/kelheim.htm (Abgerufen am 07.05.2025).
- Technische Universität Darmstadt (o.D.). 3D-Campus-Projekt – Serious Games. https://www.etit.tu-darmstadt.de/serious-games/forschung_und_projekte_sg/3d_campus_sg/index.de.jsp (Abgerufen am 07.05.2025).
- Unity Technologies (2025). Unity – Echtzeitentwicklungsplattform. https://unity.com (Abgerufen am 07.05.2025).
- Unreal Engine (2025). Unreal Engine – Offizielle Seite. https://www.unrealengine.com/de (Abgerufen am 07.05.2025).

ERKLÄRUNG

Ich versichere, dass ich diese Arbeit selbstständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Poing , den 12.05.2025



Konstantin Zeller