

Studienprojekt
**Entwicklung eines Farming Simulators auf
einer zeitgemäßen Game Engine**

im Studiengang Softwaretechnik und Medieninformatik
der Fakultät Informationstechnik
Wintersemester 2023/24

Pavithra Sureshkumar

Zeitraum: 15.10.2023 - 15.02.2024
Prüfer: Prof. Dr.-Ing. Harald Melcher
Zweitprüfer: Prof. Dr.-Ing. Andreas Rößler

Inhaltsverzeichnis

1	Einleitung	1
1.1	Kurze-Zusammenfassung	1
1.2	Motivation/Einblick	1
2	Grundlagen der Godot Game Engine	2
2.1	Einführung in Godot	2
2.1.1	Vorteile und Besonderheiten	2
2.1.2	Erste Schritte	3
2.2	Die Oberfläche von Godot	5
2.2.1	Der Projekt Manager	5
2.2.2	Die Panelübersicht	6
2.3	Szenen und Nodes	8
2.3.1	Szenen	8
2.3.2	Nodes	8
2.3.3	Hierarchie von Nodes	9
2.4	GDScript	11
2.4.1	Was ist GDScript?	11
3	Realisierung des Farming Simulators	13
3.1	Konzeptualisierung des Farming Simulators	13
3.1.1	Spielidee und Motivation	14
3.1.2	Gestaltung des Hauptcharakters	16
3.2	Praktische Umsetzung des Farming Simulators	18
3.2.1	Nomadsculpt	18
3.2.2	Mixamo von Autodesk und Blender	18
3.2.3	Blender und Godot	18
3.2.4	GDScript: Umsetzung Charakter Idle und Walk	18
4	Schluss	19
A	Kapitel im Anhang	20
	Literatur	21

Abbildungsverzeichnis

2.1	Godot Logo	2
2.2	Godotengine macOS download Seite	3
2.3	Godot Projekt Manager	5
2.4	Godot Panelübersicht	6
2.5	Godot Scene	8
2.6	Godot Nodes Hierarchie	9
2.7	GDScript Code Beispiel Sprite2D	11
3.1	Mindmap: Gedanken zum Spiel	14
3.2	Animal Crossing Body	16
3.3	Charakter Skizze	17
3.4	Erster Prototyp Charakter mit NomadSculpt	18

Tabellenverzeichnis

1

Kapitel 1

Einleitung

Die vorliegende Arbeit befasst sich mit der Entwicklung eines 3D Farming Simulators. Dieser Simulator wird unter Verwendung der zeitgemäßen Game Engine Godot erstellt, während die 3D-Assets mit NomadSculpt, Blender und Mixamo erstellt werden.

1.1 Kurze-Zusammenfassung

Das Ziel dieses Projektes ist die Entwicklung eines Farming Games unter Verwendung der Game Engine Godot 4.1.3.

Die vorliegende Arbeit dokumentiert den gesamten Prozess, einschließlich der Planung, Erstellung und Bewältigung von Herausforderungen im Zusammenhang mit dem Projekt. Dabei stehen die zentralen Aspekte im Fokus: die Planung, die Umsetzung mit der Game Engine Godot und 3D Blender sowie die Schaffung des Spiels.

1.2 Motivation/Einblick

Das Projekt wird durchgeführt, um die heute zeitgemäße Game Engine Godot kennenzulernen. Das Farming Game soll zum Entspannen dienen und einem das Gefühl geben, in einer anderen Welt einzutauchen.

Kapitel 2

Grundlagen der Godot Game Engine

Um ein besseres Verständnis über der Godot Game Engine zu erlangen, muss man ein paar Grundlagen wissen. Dieses Kapitel gibt eine kleine Einführung zu Godot und erklärt grob wie die Game Engine zu bedienen ist und was man alles mit ihr erreichen kann.

2.1 Einführung in Godot

Godot ist eine freie open-source Game Engine und wurde von Juan Lienentsky und Ariel Manzur in 2007 entwickelt. Im Jahr 2014 wurde die Game Engine unter der MIT-Lizenz veröffentlicht und kann von der öffentlichen Seite von Godot heruntergeladen werden ([1](#), S.1).



Abb. 2.1: Godot Logo ([2](#))

2.1.1 Vorteile und Besonderheiten

Die Godot Game Engine ist eine Open-Source-Plattform für die Entwicklung von Computerspielen. Sie bietet viele Vorteile, wie die Erstellung von 2D-Plattformspielen und 3D-Plattformspielen. Außerdem ist die Benutzeroberfläche der Game-Engine sehr benutzerfreundlich gestaltet.

Alles, was mit Godot erschaffen wird, gehört zu 100 Prozent den Entwicklern, im Gegensatz zu anderen kommerziellen Game-Engines. Bei anderen Game Engines ist in

der Regel eine vertragliche Zusammenarbeit erforderlich. Der Entwickler kann selbst entscheiden, wie und wo das Spiel vertrieben wird. Viele kommerzielle Game Engines haben strikte Voraussetzungen für das Veröffentlichen von aufwendigen Spielen und erfordern den Kauf einer Lizenz (3, S.4).

Godot ist auch sehr transparent, denn Entwickler können selbst entscheiden, ob sie eine bestimmte Eigenschaft der Engine modifizieren oder neue Eigenschaften implementieren möchten, ohne eine spezielle Genehmigung einholen zu müssen. Diese Eigenschaft ist besonders hilfreich bei der Entwicklung größerer Projekte, da man vollen Zugriff auf die internen Funktionen der Engine hat (3, S.5).

Für viele Entwickler ist Godot daher die bessere Lösung für die Spieleentwicklung. Godot wird nicht von einem Unternehmen entwickelt, sondern ist den Entwicklern gewidmet, die ihre Zeit und Erfahrung investieren, um die Engine zu erstellen, zu testen und Fehler zu beheben. Außerdem führen sie ausführliche Dokumentationen über die Game Engine. Viele Entwickler erstellen auch einige Prototypen von Starter-Gameszenen, um mit Godot den ersten Schritt zu machen (3, S.5).

2.1.2 Erste Schritte

Um mit Godot durchzustarten, muss man die neuste Version von der Godot Seite herunterladen <https://godotengine.org/download/macOS/>. Die jetzige Version sollte im Bereich von Godot 4.x.x sein. Auf der Seite wird auch .NET angeboten, diese wird aber nur gebraucht, wenn man mit C# arbeiten möchte. Alternativ kann man Godot auch per Steam, itch.io oder mit dem Paketmanager (Homebrew, Scoop, Snap) der jeweiligen OS installieren (3, S.6).



Abb. 2.2: Godotengine macOS download Seite (2)

Nach dem klassischen download wird dann ein zip Ordner bereitgestellt, diese wird dann entpackt. Alternativ kann man auch kann man diese auch zu dem Programm

Ordner oder dem Applikationen Ordner rüberziehen. Nach dem öffnen der Applikation sieht man den Godot Projekt Manager Fenster (3, S.6).

Die folgenden Kapitel handeln von der Godot Engine, wie sie zu bedienen ist, die einzelnen Kernmerkmale der Godot Engine und die verschiedenen Elemente, um einen groben Überblick zu geben. was die Game Engine zu bieten hat. Die Kapiteln dienen als Basis um mit dem Projekt Farming Simulator durchzustarten zu können.

2.2 Die Oberfläche von Godot

Dieses Kapitel beschreibt den Aufbau der Oberfläche von Godot.

2.2.1 Der Projekt Manager

Nach dem Öffnen von Godot erscheint zunächst der Projekt Manager. Hier kann man entscheiden, ob man ein neues Projekt erstellen möchte oder in der öffentlichen Asset-Bibliothek nach Projekten suchen möchte (3, S.7-8).

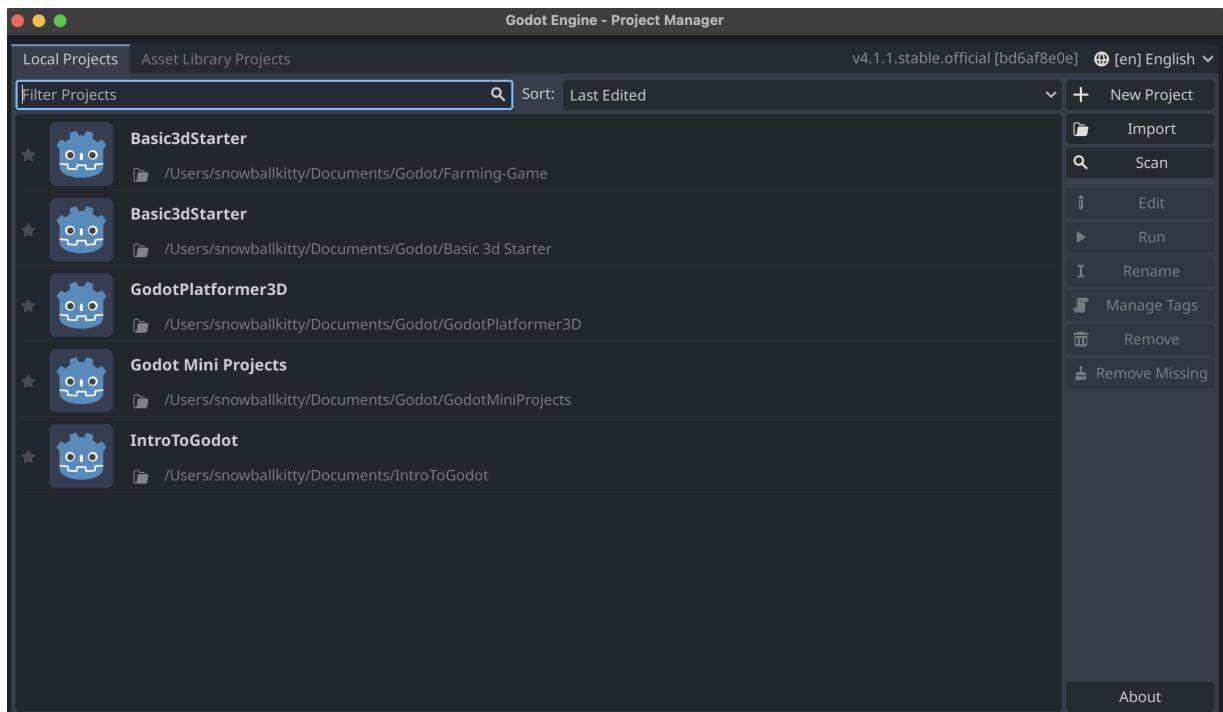


Abb. 2.3: Godot Projekt Manager

2.2.2 Die Panelübersicht

Die Godot-Oberfläche besteht aus mehreren Panels. Die einzelnen Abschnitte wie das Szenenbaum, der Fielssystem-Abschnitt, die Arbeitsfläche und der Inspector haben ihre eigene Funktion (3, S.9-11).

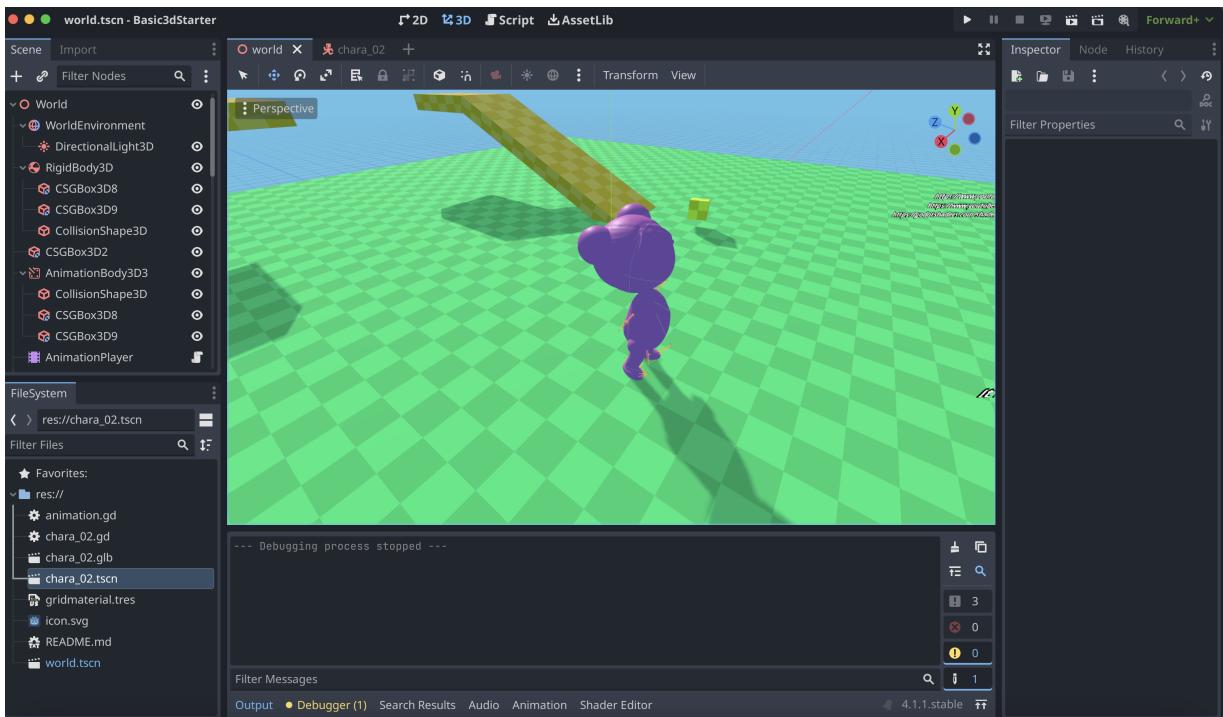


Abb. 2.4: Godot Panelübersicht

In Abbildung 2.4 ist zu erkennen, dass der größte Teil des Editor-Panels das Viewport ist, welches sich in der Mitte befindet. Der Viewport dient der Übersicht über die einzelnen Elemente, die ein Entwickler bearbeitet (3, S.9).

Im oberen Abschnitt der Oberfläche befindet sich die Liste der Arbeitsplätze. Hier kann zwischen 2D-, 3D- und Script-Modus gewechselt werden. Im Script-Modus wird der Spielcode bearbeitet. Der Tab AssetLib ist für das Herunterladen von Add-ons und Beispielprojekten zuständig, die von der Godot-Community bereitgestellt werden (3, S.10).

Über dem Viewport ist die Toolbar sichtbar. Hier können verschiedene Tools verwendet werden, um mit dem Viewport und den einzelnen Elementen zu interagieren (3, S.10).

In der oberen rechten Ecke befindet sich der Player, um das Spiel zu testen. Beim Testen des Spiels erscheint ein neues Fenster, das das Spiel debuggt (3, S.10).

In der linken unteren Ecke des Panels findet man das Dateisystem. Hier werden alle Dateien des Projektordners angezeigt. Alle Ressourcen werden im relativen Pfad res:// lokalisiert, was dem Stammverzeichnis des Projekts entspricht (3, S.10).

In der oberen linken Hälfte ist der Szenenbaum zu sehen, in Abbildung 2.4 auch als Scene bezeichnet. Dieser zeigt die aktuelle Szene an, die im Viewport bearbeitet wird (3, S.11).

Auf der rechten Seite befindet sich der Inspector, in dem die Eigenschaften der Spielobjekte angepasst werden können (3, S.11).

Mit diesen Informationen ist man nun bereit, mit der Godot Engine zu arbeiten. Die Funktionsweise der einzelnen Elemente in Godot lässt sich jedoch nur durch das Bearbeiten von Projekten erfahren. Die folgenden Kapitel geben einen noch tieferen Einblick in die Elemente von Godot, um das Projekt "Farming Simulator" zu starten.

2.3 Szenen und Nodes

Im Kapitel 2.2.2 wurden die einzelnen Elemente der Godot-Oberfläche grob aufgezeigt. Eines der wichtigen Elemente ist der Szenenbaum, welches in diesem Kapitel näher behandelt wird.

2.3.1 Szenen

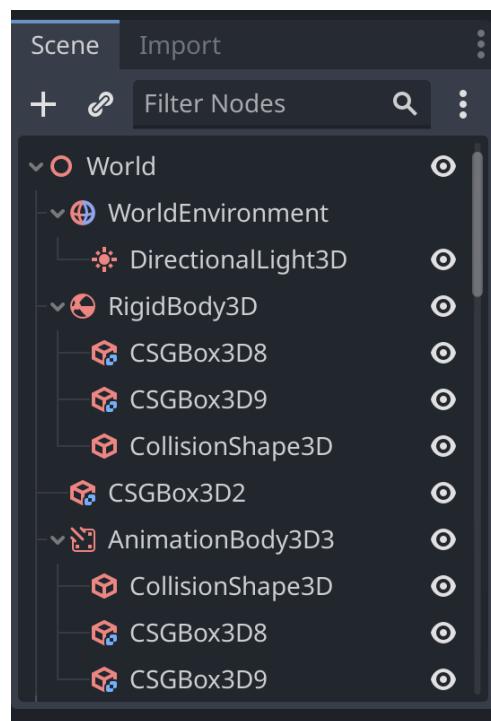


Abb. 2.5: Godot Scene

Eine Szene 2.5 wird in Godot verwendet, um die verschiedenen Spielobjekte im Projekt zu erstellen und zu ordnen. Zum Beispiel kann eine Szene nur für den Spieler erstellt werden, mit den dazugehörigen Nodes und Skripten, um den Spieler zu steuern. Eine andere Szene könnte für die Spielwelt erstellt werden, in der der Charakter mit den erstellten Objekten und Hindernissen in der Welt interagiert. Am Ende werden die Szenen kombiniert, um ein finales Spiel zu erstellen (3, S.12).

2.3.2 Nodes

Nodes sind die Grundbausteine von Godot, um Spiele zu erstellen. Eine Node ist ein Objekt, das verschiedene spezialisierte Spielfunktionen haben kann. Zum Beispiel kann eine Node ein Typ von einer Bildanzeige sein, um eine Animation abzuspielen, oder

ein 3D-Modell darstellen. Eine Node kann eine Sammlung von Eigenschaften besitzen, die dann verwendet werden können, um das Verhalten der Node anzupassen. Welche Nodes ein Entwickler verwendet, ist ihm überlassen, je nach der gewünschten Funktionalität. Es handelt sich um ein modulares System, das dem Entwickler genügend Freiraum gibt und flexibel ist, um die Spielobjekte zu erstellen (3, S.11).

Nodes bringen hauptsächlich ihre eigenen Eigenschaften und Funktionen mit. In Godot kann der Entwickler jedoch das Verhalten oder die Funktionalität der Nodes erweitern, indem er Skripte zu den jeweiligen Nodes hinzufügt. Diese Möglichkeit ermöglicht es dem Entwickler, mehr aus dem Standard-Node herauszuholen. Zum Beispiel kann man zum Sprite2D-Node eine Eigenschaft hinzufügen, die ein Bild anzeigen soll. Möchte man jedoch, dass das Bild beim Klicken verschwindet und beim erneuten Klicken wieder erscheint, muss man ein Skript einfügen, um dieses Verhalten zu erzeugen (3, S.12).

2.3.3 Hierarchie von Nodes

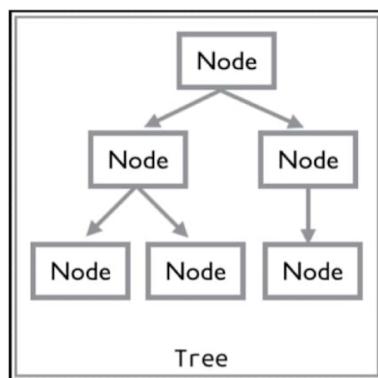


Figure 1.10: Nodes arranged in a tree

Abb. 2.6: Godot Nodes Hierarchie
(3, S.11)

In Abbildung 2.6 werden die Nodes als eine Baumstruktur dargestellt. In einem Baum werden die Nodes als Kinder von anderen Nodes hinzugefügt. Eine bestimmte Node kann viele Kinder haben, aber nur eine Eltern-Node. Wenn eine Gruppe von Nodes zu einem Baum zusammengefügt wird, nennt man dies eine Szene (3, S.11). Wird eine Szene abgespeichert, kann diese als eine neue Node zu einem anderen Node als Kind hinzugefügt werden (4).

Nodes sind ein mächtiges Werkzeug. Das Verständnis dieses Tools ist eine Voraussetzung, um die Objekte in Godot zu begreifen. Doch ohne eine richtige Spiellogik sind Nodes nicht viel wert. Eine Spiellogik legt Regeln fest, die die Objekte befolgen sollen

(3, S.12). Im nächsten Kapitel wird die Skriptsprache GDScript behandelt, die eine der Sprachen ist, die man bei Godot verwenden kann, um eine Spiellogik aufzubauen.

2.4 GDScript

Dieses Kapitel behandelt die verfügbaren Programmiersprachen in Godot, wobei hauptsächlich auf die Skriptsprache GDScript eingegangen wird.

In Godot kann man zwischen zwei Programmiersprachen wählen, um die Nodes zu scripten: GDScript und C#. GDScript ist die dedizierte, integrierte Sprache mit der engsten Integration zur Engine und ist die unkomplizierte Sprache, um Godot zu nutzen. Alternativ kann man auch mit C# programmieren, indem man die entsprechende Version von Godot herunterlädt, die diese Sprache unterstützt. Godot selbst ist in C++ geschrieben, und man kann durch die Verwendung von C++ mehr Leistung und Kontrolle über die Funktionen der Engine direkt erhalten (3, S.12).

Die beste Wahl zum Scripten mit Godot ist GDScript, da es eng mit der Godot - Programmierschnittstelle integriert ist und speziell für schnelle Entwicklung entwickelt wurde (3, S.5).

2.4.1 Was ist GDScript?

Die Syntax der Skriptsprache GDScript ist stark an die Python-Sprache angelehnt. Personen, die bereits Erfahrung mit anderen dynamischen Programmiersprachen wie JavaScript haben, sollten es relativ einfach finden, GDScript zu erlernen. Ähnlich wie Python ist GDScript eine benutzerfreundliche Programmiersprache, besonders geeignet für Anfänger.

```
extends Sprite2D
var speed = 200

func _ready():
    position = Vector2(100, 100)

func _process(delta):
    position.x += speed * delta
```

Abb. 2.7: GDScript Code Beispiel Sprite2D (3, S.13)

GDScript ist eine dynamisch typisierte Skriptsprache, das bedeutet, dass man keine Variablenarten deklarieren muss, wenn man eine Variable erstellt. Stattdessen verwendet GDScript Leerzeichen (Einrückungen), um Codeblöcke zu kennzeichnen. Der größte Vorteil von GDScript liegt in der engen Integration mit der Game Engine, was zu einer schnellen Entwicklung führt und somit wenig Code erforderlich macht.

Die Abbildung 2.7 zeigt grob, wie GDScript aussieht. Der Code repräsentiert ein Sprite2D, das von links nach rechts über den Bildschirm bewegt wird.

Falls weitere Fragen zur GDScript-Sprache auftreten, sollten entsprechende Informationen in der Godot - Dokumentation zu finden sein: <https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/index.html>.

Im weiteren Verlauf des Projekts werden weitere Code-Snippets in GDScript eingeführt.

Kapitel 3

Realisierung des Farming Simulators

Dieses Kapitel befasst sich mit der Umsetzung des Farming Simulators mithilfe der Godot Engine. Dabei werden nicht nur die technischen Aspekte erörtert, sondern auch die Gestaltung des Spiels. Es werden Fragen behandelt, wie beispielsweise: "Warum sollte überhaupt ein Farming Simulator-Spiel entwickelt werden?"

3.1 Konzeptualisierung des Farming Simulators

In diesem Unterkapitel werden hauptsächlich die Spielkonzepte diskutiert. Die technischen Aspekte und die praktische Umsetzung werden im nächsten Kapitel behandelt.

3.1.1 Spielidee und Motivation

Zunächst werden die Gedanken zum Spiel sortiert, dies wurde mithilfe einer Mind Map realisiert 3.1.

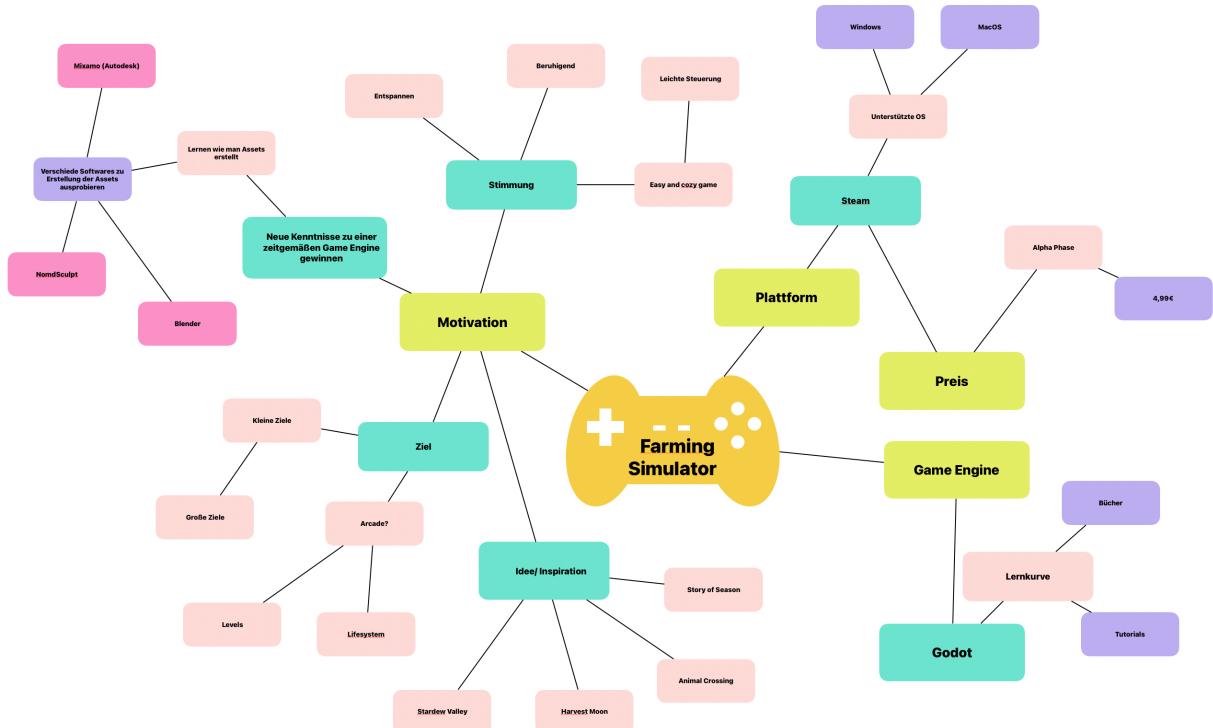


Abb. 3.1: Mindmap: Gedanken zum Spiel

Als erstes stellt sich die Frage: Warum überhaupt ein Farming Simulator Spiel?

Die Motivation hinter diesem Spielkonzept besteht darin, eine bestimmte Stimmung zu erzeugen. Die Idee wurde von bekannten Farming-Simulatoren wie Story of Seasons, Animal Crossing, Stardew Valley und Harvest Moon inspiriert. Bei all diesen Spielen liegt der Fokus auf "Easy Gaming" und "Cozy Gaming". Das Ziel ist es, dem Spieler eine beruhigende Atmosphäre zu vermitteln und ihn auf eine besondere Weise zu entspannen.

Natürlich gibt es auch Entwicklungsziele, die als Motivation dienen. Dazu gehört das Erlernen zeitgemäßer Game Engines sowie das eigenständige Erstellen von Assets unter Verwendung neuer Softwaretools wie Mixamo, NomadSculpt oder Blender. Das übergeordnete Ziel bei der Spielentwicklung ist es, kleine Schritte zu unternehmen und diese schließlich zu großen Fortschritten zu führen.

Im Verlauf des Entwicklungsprozesses stellte sich auch die spannende Frage: "Sollte man ein Arcade-Element integrieren?" Bei diesem Überlegungsschritt wurden Gedanken darüber angestellt, welches Ziel der Spieler im Spiel verfolgt. Ein Spiel besteht oft

aus kleinen Zielen, die der Spieler erreichen möchte. Dazu gehören nicht nur Level oder ein Lebenssystem, sondern auch der Zyklus bestimmter Aufgaben. Im Farming Simulator wird dies beispielsweise durch das Einpflanzen von Samen, das Ernten der Pflanzen und das erneute Erlangen von Samen dargestellt.

Die Auswahl der richtigen Game Engine ist ein entscheidender Aspekt für Entwickler, insbesondere für diejenigen, die wenig Erfahrung haben. Godot wird als ausgezeichnete Wahl für Anfänger empfohlen, insbesondere für die Entwicklung kleiner Indie-Spiele. Dennoch erfordert das Erlernen der verschiedenen Elemente der Game Engine erheblichen Aufwand, und die Lernkurve ist eher träge und nicht stetig steigend. Es ist ratsam, die Godot-Dokumentation zu studieren, Tutorials durchzugehen und auch Bücher zu lesen, um sich mit der Plattform vertraut zu machen.

Wenn es darum geht, das Spiel auf einer Plattform zu veröffentlichen, stellt sich die Frage: "Wo soll es hochgeladen werden?" Eine Möglichkeit besteht darin, es auf Plattformen wie Steam zu veröffentlichen. Auch die Auswahl des Betriebssystems, auf dem das Spiel laufen soll, ist entscheidend. Die Frage nach dem Verkaufspreis ist ebenfalls wichtig. In der aktuellen Alpha-Phase könnte eine angemessene Preisgestaltung bei etwa 4,99 Euro liegen.

All diese Überlegungen dienen dazu, einen groben Überblick über das Spiel zu erhalten und das angestrebte Ziel zu erreichen.

3.1.2 Gestaltung des Hauptcharakters

In diesem Kapitel wird die Gestaltung des Charakters behandelt.

Wie bereits im vorherigen Kapitel 3.1 besprochen, wurde das Farmingspiel von Animal Crossing inspiriert.



Abb. 3.2: Animal Crossing Body (5)

Dabei wurde die Figur 3.2 von Animal Crossing genauer betrachtet. Auf den ersten Blick fällt auf, dass die Figur sehr einfach und dennoch einheitlich gestaltet ist. Der Kopf ist eine Kugel, der Körper ein Zylinder, der oben leicht eingezogen ist, und die Hände sind einteilig, das heißt, es gibt keine Finger, sondern nur eine Kugel als Hand.

Nach der Analyse wurde eine grundlegende Skizze des Körpers mit Procreate angefertigt.

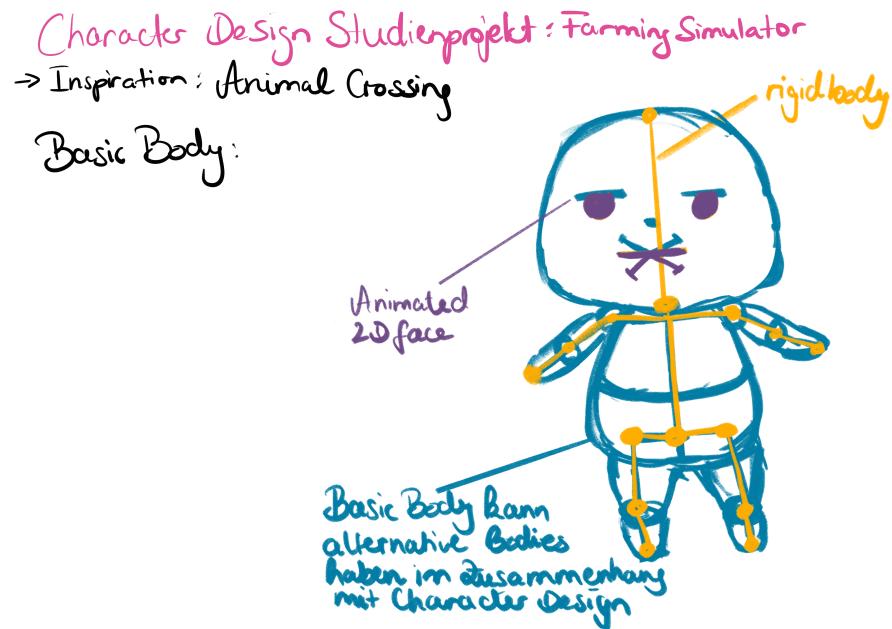


Abb. 3.3: Charakter Skizze

In Abbildung 3.3 wird der Charakter grob skizziert. Das grundlegende Körpermodell wird je nach Bedarf angepasst. Der Rigid Body wird später für Animationen verwendet. Das Gesicht kann entweder statisch oder in 2D animiert sein, abhängig von den Anforderungen.

3.2 Praktische Umsetzung des Farming Simulators

3.2.1 NomadSculpt

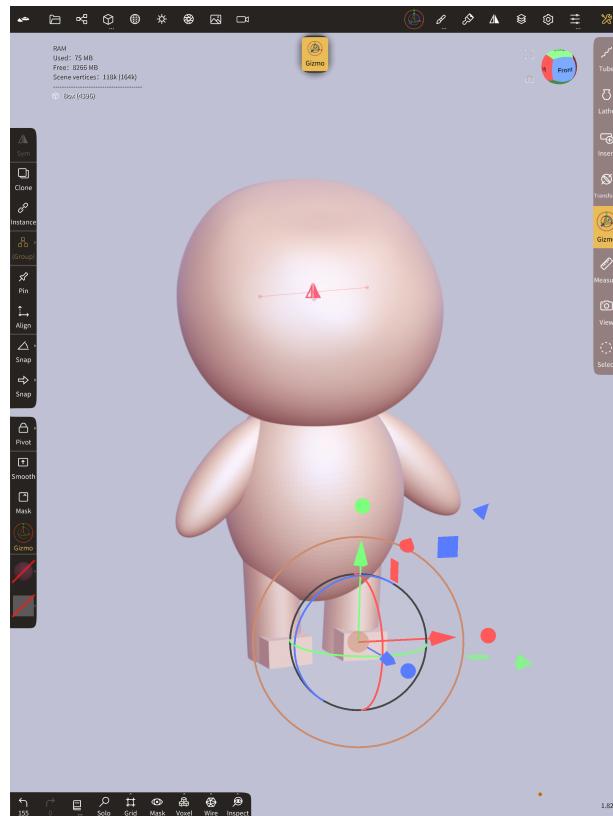


Abb. 3.4: Erster Prototyp Charakter mit NomadSculpt

3.2.2 Mixamo von Autodesk und Blender

3.2.3 Blender und Godot

3.2.4 GDScript: Umsetzung Charakter Idle und Walk

Kapitel 4

Schluss

Ergebnis-Bewertung, Zusammenfassung und Ausblick

Anhang A

Kapitel im Anhang

Alles was den Hauptteil unnötig vergrößert hätte, z. B. HW-/SW-Dokumentationen, Bedienungsanleitungen, Code-Listings, Diagramme

Literatur

1. JOHNSON, J. *Godot 4 Game Development Cookbook: Over 50 solid recipes for building high-quality 2D and 3D games with improved performance*. Packt Publishing, 2023. ISBN 9781838827250. Auch verfügbar unter: <https://books.google.de/books?id=FNS-EAAAQBAJ>.
2. *Godotengine download macOs*. [o. D.]. Auch verfügbar unter: <https://godotengine.org/download/macos/>. Zugegriffen: 2023-25-12.
3. BRADFIELD, C. *Godot 4 Game Development Projects: Build five cross-platform 2D and 3D games using one of the most powerful open source game engines*. Packt Publishing, 2023. ISBN 9781804615621. Auch verfügbar unter: <https://books.google.de/books?id=GrfLEAAAQBAJ>.
4. *Godot Docs – 4.2 branch — docs.godotengine.org* [<https://docs.godotengine.org/en/stable/index.html>]. [o. D.]. [Accessed 30-12-2023].
5. *AnimalCrossing* [<https://i.pinimg.com/originals/07/f7/ec/07f7ec705585cd7a9c.jpg>]. [o. D.]. [Accessed 14-01-2024].