

# **Super Ninja**

# **2D**

**DIE FLUCHT AUS DER HÖHLE DES LICHTS**

**ITIG WS 24/25**

AMANULLAH NADERI, MOHAMMAD TAIBA UND EMAN AL-MESRI

# Contents

|   |    |
|---|----|
| Installationshinweise und Systemanforderungen ..... | 3  |
| 1. Systemanforderungen .....                        | 3  |
| 2. Installation .....                               | 3  |
| 3. Tastenbelegung .....                             | 3  |
| Spielkonzept .....                                  | 4  |
| Idee .....  | 4  |
| Genre .....   | 4  |
| Zielgruppe .....                                    | 4  |
| Inspirationsquellen .....                           | 5  |
| Spielmechanik .....                                 | 5  |
| Bewegung und Sprung: .....                          | 5  |
| Kampfmechanik: .....                                | 5  |
| Health-System: .....                                | 5  |
| Level-Fortschritt: .....                            | 6  |
| Gegner und Bosskämpfe: .....                        | 6  |
| Gameloop .....                                      | 6  |
| Levels, Design, Asthetic .....                      | 7  |
| Levels .....  | 7  |
| Allgemeiner Aufbau der Levels .....                 | 7  |
| Design und Herausforderungen .....                  | 10 |
| Interaktive Elemente .....                          | 12 |
| Grafik und Atmosphäre .....                         | 15 |
| Entwicklungsdetails .....                           | 16 |
| Assets .....  | 20 |
| fremde Assets .....                                 | 20 |
| Projektübersicht .....                              | 23 |
| Team .....  | 23 |

|  |    |
|--|----|
| Arbeitsweise.....                              | 23 |
| 1. Kommunikation und Organisation.....         |    |
| 23   |    |
| 2. Aufgabenverteilung.....                     |    |
| 24   |    |
| Offene ToDos, Ideen für Weiterentwicklung..... | 25 |
| Code-Ausschnitte.....                          | 26 |
| Anhang.....                                    |    |
| 30   |    |

# Installationshinweise und Systemanforderungen

## 1. Systemanforderungen

| Hardware        | Minimum              |
|-----------------|----------------------|
| Prozessor       | Intel Core i3 2100   |
| Arbeitsspeicher | 4 GB                 |
| Festplatte      | 1 GB freier Speicher |
| Grafikkarte     | Interne Grafik       |

Das Spiel auch im braucht sehr geringe Anforderungen. Mit der Low Poly Grafik und den Grafikeinstellungen ist das Spiel auf so gut wie jedem Rechner spielbar.

## 2. Installation

Die Super Ninja 2D Archivdatei muss zunächst entpackt werden. Anschließend kann die enthaltene "Super Ninja 2D.exe" gestartet werden. Nach ein paar Sekunden sollten Sie sich im Hauptmenü befinden und das Spiel beginnen können.

## 3. Tastenbelegung

| Taste   | Funktion                         |
|---|----------------------------------|
| - W, A, S, D<br><br>- Pfeiltasten: Arrow Up, Arrow Down,<br>Arrow Left, Arrow Right | Für die Bewegung des Charakters. |
| Mausklick links   | Der Charakter bringt um.         |
| ESC   | Aufruf Hauptmenü.                |
| Leerzeichen   | Der Charakter springt.           |

# Spielkonzept

## • Idee

Die Idee hinter unserem Spiel "**Super Ninja 2D**" ist es, den Spieler in eine spannende, illusionäre Höhlenwelt eintauchen zu lassen. Der Spieler übernimmt die Rolle eines mutigen Ninjas, der aus einer wunderschönen, aber gefährlichen Höhle entkommen muss. Dabei gilt es, Gegner zu besiegen, Hindernisse zu überwinden und Münzen zu sammeln, um das nächste Level zu erreichen. Die ultimative Herausforderung ist der Kampf gegen den Endgegner, einen mächtigen Höhlendrachen, der den Ausgang bewacht. Unser Ziel war es, eine Spielwelt zu schaffen, die die Spieler fesselt und gleichzeitig die Spannung zwischen der Schönheit der Umgebung und der Bedrohung durch die Gegner spürbar macht. Die Idee soll Abenteuerlust und Kampfgeist wecken und gleichzeitig die Spieler visuell und akustisch beeindrucken.

## • Genre

Das Spiel gehört zum Genre der **2D-Plattform-Action-Adventures**. Dabei stehen klassische Elemente wie Springen, Laufen und Hindernisse überwinden im Vordergrund, kombiniert mit actionreichen Nahkämpfen gegen Feinde. Das Spiel enthält außerdem strategische Aspekte, da der Spieler sich durch immer schwierigere Level kämpfen muss und dabei geschickt mit seinen Ressourcen wie Leben und Angriffsmöglichkeiten umgehen muss. Es ist ein Mix aus Retro-Charme, inspiriert von klassischen Plattformspielen wie "Super Mario", und moderner Ästhetik, mit Fokus auf immersivem Design und einer sich entwickelnden Schwierigkeit. Unser Ziel war es, ein herausforderndes, aber zugängliches Spiel zu entwickeln, das sowohl erfahrene Gamer als auch Gelegenheitsspieler anspricht.

## • Zielgruppe

Unsere Zielgruppe sind vor allem Jugendliche und junge Erwachsene, im Alter von etwa 12 bis 30 Jahren, die Spaß an klassischen Plattformspielen haben, aber auch an neuen und kreativen Spielideen interessiert sind. Das Spiel richtet sich an Menschen, die gerne Herausforderungen annehmen und Freude daran haben, eine lebendige und detailreiche Spielwelt zu erkunden. Besonders Spieler, die Spiele wie "Super Mario" lieben oder mit ähnlichen Plattformspielen aufgewachsen sind, dürften sich sofort wohlfühlen. Durch die

intuitive Steuerung und den schrittweisen Schwierigkeitsanstieg wollen wir jedoch auch Neulinge im Bereich der Action-Plattformspiele ansprechen. Unser Ziel war es, ein Spiel zu schaffen, das leicht zu verstehen ist, aber dennoch durch seine Tiefe und Atmosphäre überzeugt.

## • **Inspirationsquellen**

Unsere größte Inspirationsquelle war zweifellos das legendäre Spiel "**Super Mario**", das uns mit seiner Einfachheit, Zugänglichkeit und gleichzeitig großen Vielfalt an Ideen fasziniert hat. Wir haben uns von den Mechaniken und dem Leveldesign inspirieren lassen, aber gleichzeitig wollten wir etwas Eigenes schaffen, das durch seine Ästhetik und Atmosphäre auffällt. Zusätzlich haben wir Inhalte aus Vorlesungsbeispielen verwendet, um bestimmte Konzepte wie Bewegungs- und Kampfmechaniken umzusetzen. Die Idee einer illusionären Höhlenlandschaft mit spannenden Kämpfen und Licht- und Soundeffekten kam aus unserer Vorstellung, eine Welt zu schaffen, die sich realistisch und lebendig anfühlt, aber gleichzeitig eine mystische und gefährliche Atmosphäre ausstrahlt. Die Balance zwischen friedlicher Schönheit und intensiver Action war für uns ein zentraler Punkt bei der Entwicklung.

# **Spielmechanik**

## **Bewegung und Sprung:**

- o Der Spieler kann sich seitlich bewegen (rechts und links) und Hindernisse durch Sprünge überwinden. Es ist möglich, bis zu dreimal hintereinander zu springen, um höher gelegene Plattformen zu erreichen oder Angriffen auszuweichen.
- o Die Steuerung erfolgt intuitiv über die Pfeiltasten oder über „A,D,W,S“ für Bewegung und die Leertaste zum Springen.

## **Kampfmechanik:**

- o Der Spieler kann Feinde im Nahkampf angreifen, indem er mit einem Mausklick das Schwert schwingt. Dabei wird eine präzise Steuerung benötigt, um Feinde effektiv zu besiegen.
- o Jeder Angriff hat das Ziel, Illusionen zu durchbrechen und den Weg zum nächsten Level freizumachen.

## **Health-System:**

- o Der Spieler hat eine begrenzte Anzahl an Lebenspunkten. Wenn diese aufgebraucht sind, verliert der Spieler und muss das Level erneut beginnen.
- o Ein besonderes Feature ist, dass das Health-System während des Kampfes mit bestimmten Aktionen (z. B. durch Besiegen von Feinden oder Sammeln von Items) wieder aufgefüllt werden kann.

## **Level-Fortschritt:**

- o Jedes Level ist durch Hindernisse, Gegner und eine illusionäre Höhlenlandschaft gekennzeichnet. Der Spieler kann nur ins nächste Level aufsteigen, wenn alle Gegner im aktuellen Level besiegt wurden.
- o Münzen sammeln ist ein optionales Ziel, das die Spielerfahrung bereichert und den Spieler dazu motiviert, alle Ecken der Level zu erkunden.

## **Gegner und Bosskämpfe:**

- o Die Gegner variieren in ihrer Schwierigkeit und Mechanik je nach Level. Sie reichen von einfachen Wächtern bis hin zu mächtigen Bossgegnern, wie dem Höhlendrachen.
- o Besonders der Endboss-Kampf sollte so gestaltet, dass er sowohl strategisches Denken als auch schnelle Reaktionen erfordert.

## **Gameloop**

Der Gameloop von "Super Ninja 2D" ist auf eine intuitive, aber motivierende Spielerfahrung ausgelegt:

### **1. Start des Spiels:**

- o Der Spieler beginnt im Hauptmenü, wo Optionen wie „Play“, „Settings“, „Sound“ oder „Restart“ zur Verfügung stehen. Nach der Auswahl des Spiels wird der Spieler ins erste Level gebracht.

### **2. Level-Ablauf:**

- o Der Spieler bewegt sich durch die Level, sammelt Münzen und besiegt Gegner. Hindernisse müssen durch präzise Sprünge überwunden werden.
- o Jeder besiegte Gegner bringt den Spieler einen Schritt näher zur Freischaltung des nächsten Bereichs.
- o Es gibt visuelles Feedback, wie z. B. Animationen bei Treffern oder Münzen, sowie Soundeffekte für Angriffe und andere Aktionen.

### **3. Übergang zwischen Leveln:**

- o Sobald alle Gegner in einem Level besiegt sind, öffnet sich der Zugang zum nächsten Level. Dieser Fortschritt ist linear, wodurch ein Gefühl der Belohnung und des Vorankommens entsteht.

### **4. Steigende Schwierigkeit:**

- o Mit jedem Level steigen die Komplexität und die Herausforderung. Neue Gegnerarten, wie „Damon“ oder der „Höhlendrache“, werden eingeführt, die komplexere Angriffsmuster und höhere Trefferpunkte haben.

### **5. Finale Herausforderung:**

- o Der Höhepunkt des Spiels ist der Kampf gegen den Höhlendrachen, der als Endgegner fungiert. Hier muss der Spieler alle Mechaniken meistern und strategisch vorgehen, um den Drachen zu besiegen und den Ausgang aus der Höhle zu erreichen.

## 6. Game Over oder Neustart:

- o Wenn die Lebenspunkte aufgebraucht sind, endet das Spiel und der Spieler kann das Level erneut versuchen. Durch die Möglichkeit des Neustarts bleibt der Spieler motiviert, die Herausforderung erneut anzugehen.

Das Zusammenspiel dieser Elemente sorgt für einen konstanten Rhythmus aus Herausforderung, Belohnung und Fortschritt, der den Spieler in der Welt von "Super Ninja 2D" hält.

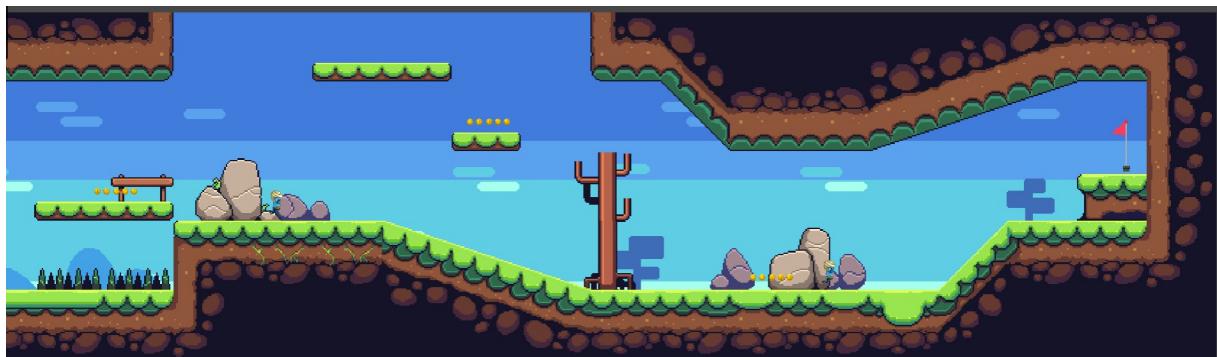
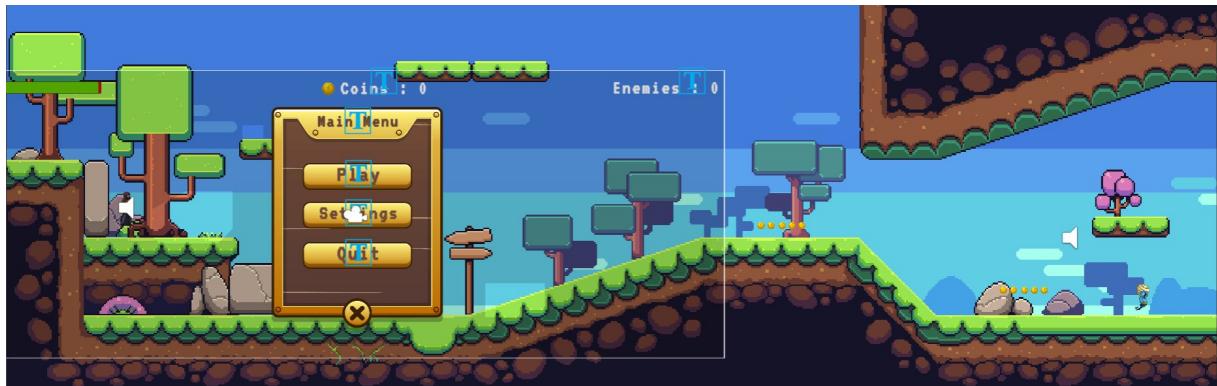
# Levels, Design, Asthetic

## Levels

### Allgemeiner Aufbau der Levels

Das Spiel "Super Ninja 2D" besteht aus insgesamt vier Levels. Jedes Level ist einzigartig gestaltet und folgt einer fortlaufenden Schwierigkeitssteigerung:

- **Level 1:** Einführung in die Spielmechaniken. Die Gegner sind relativ einfach zu besiegen, und das Level dient dazu, dem Spieler die Steuerung und das grundlegende Gameplay näherzubringen.



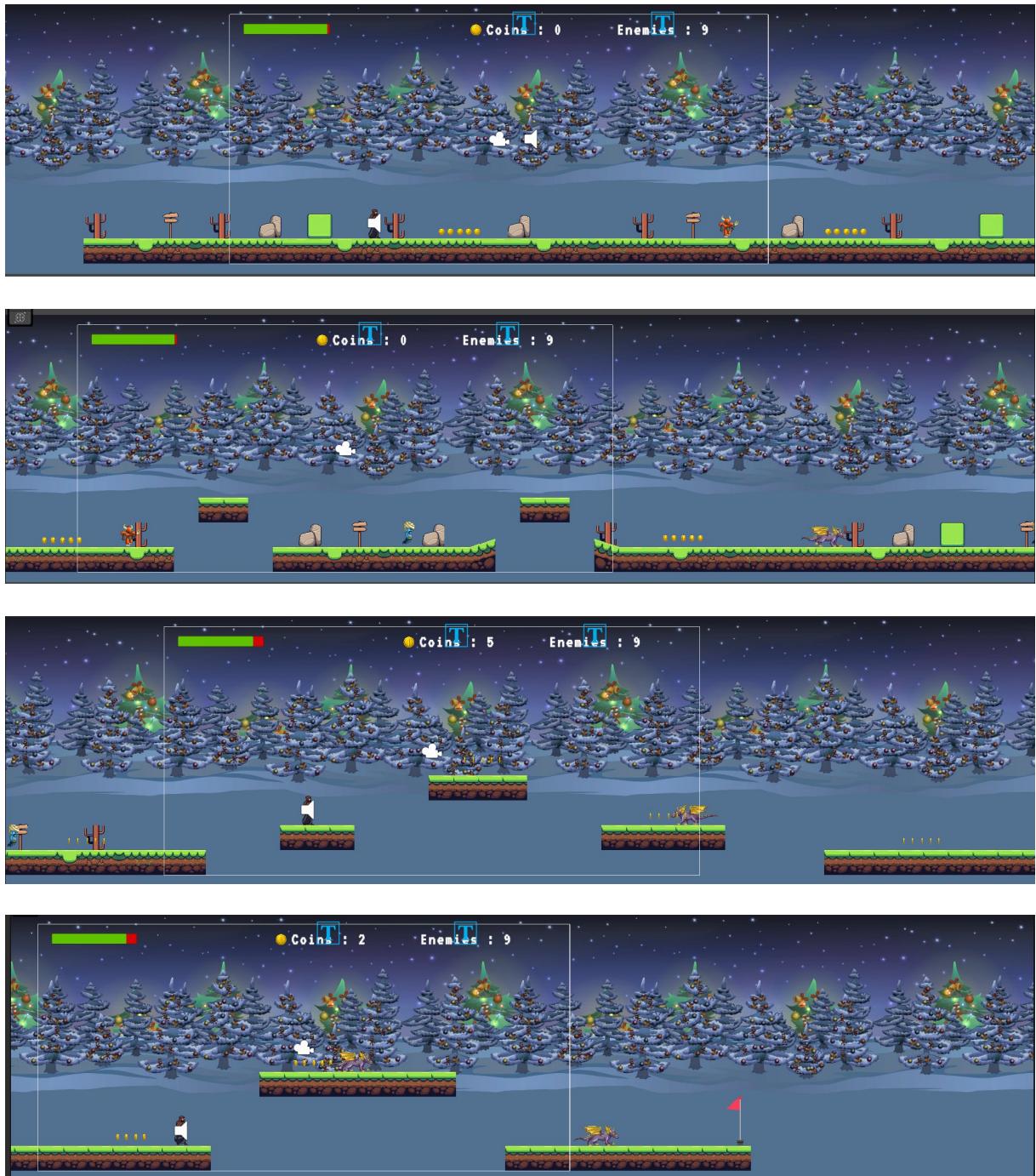
„Abbildung 1: Level 1“

- **Level 2:** Einführung neuer Gegner, wie "Damon", mit erhöhter Komplexität (Schwierigkeitsgrad 2). Die Herausforderungen werden größer, und der Spieler muss erweiterte Fähigkeiten nutzen, um voranzukommen.



„Abbildung 2: Level 2“

- **Level 3:** Weitere neue Gegner, darunter der "Drache", mit einem Schwierigkeitsgrad von 3. Das Level erfordert präzise Sprung- und Angriffsfähigkeiten, um Hindernisse zu überwinden.



„Abbildung 3: Level 3“

- **Level 4:** Das Finale mit dem Endboss "Höhlendrache" (Schwierigkeitsgrad 6). Dieses Level kombiniert alle bisher gelernten Fähigkeiten und fordert den Spieler mit komplexen Angriffsmustern und einer intensiven Bossmechanik.



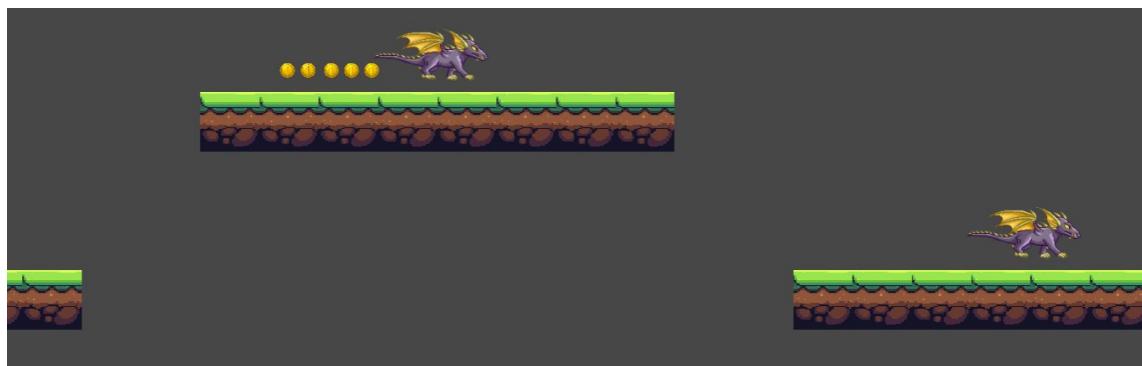
„Abbildung 4: Level 4“

Die Levels zeichnen sich durch eine illusionäre Höhlenlandschaft aus, die durch eine friedliche Atmosphäre und steigende Gefahren geprägt ist. Es gibt keine animierten Übergänge oder Ladebildschirme zwischen den Levels, was das Gameplay nahtlos macht.

### *Design und Herausforderungen*

Die Levels sind so konzipiert, dass sie mit jeder Stufe anspruchsvoller werden:

- **Hindernisse und Gegner:** Jedes Level enthält einzigartige Hindernisse, wie Plattformen, Abgründe und Gegner mit unterschiedlichen Angriffsmustern. Zu den Gegnern zählen unter anderem "Jins", "Damon" und der "Höhlendrache".



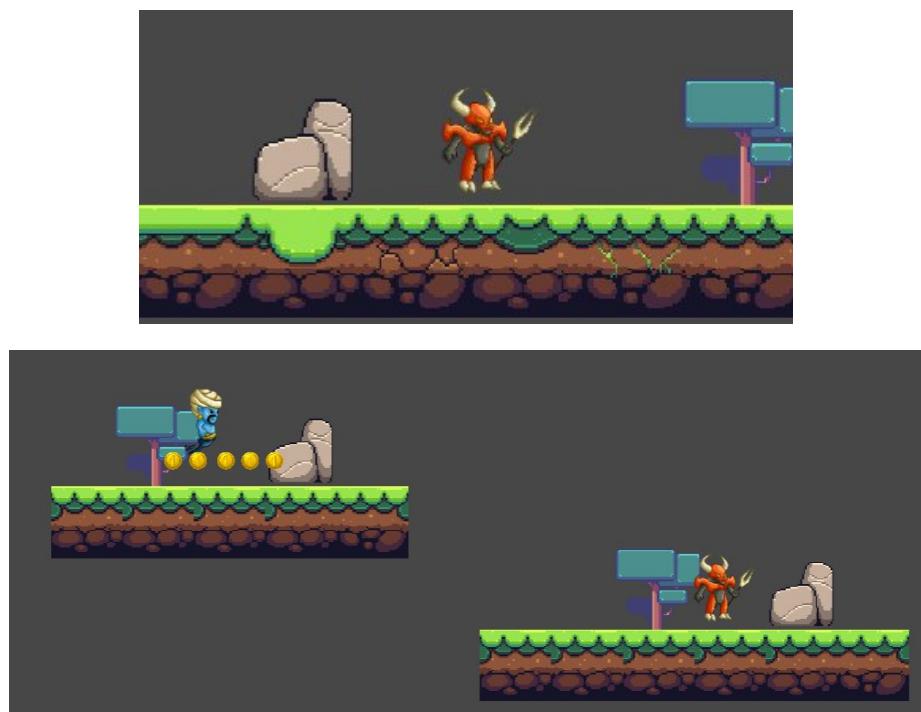
„Abbildung 5: Hindernisse & Gegner aus Level 3“

- **Bosskämpfe:** In Level 4 steht der Spieler dem mächtigen Höhlendrachen gegenüber. Dieser Kampf erfordert eine Kombination aus präzisem Timing, geschicktem Ausweichen und strategischen Angriffen.



„Abbildung 6: Bosskämpfe“

- **Schwierigkeitssteigerung:** Die Gegner werden mit jedem Level stärker und komplexer, was dem Spieler eine stetige Lernkurve bietet.



„Abbildung 7: aus der Level 2“

## Interaktive Elemente

Die Levels enthalten einige interaktiver Elemente:

- **Münzen:** Überall in den Levels sind Münzen verteilt, die der Spieler sammeln kann. Diese fördern die Erkundung und bieten zusätzliche Motivation.

Vorher:



„Abbildung 8: Level 1“

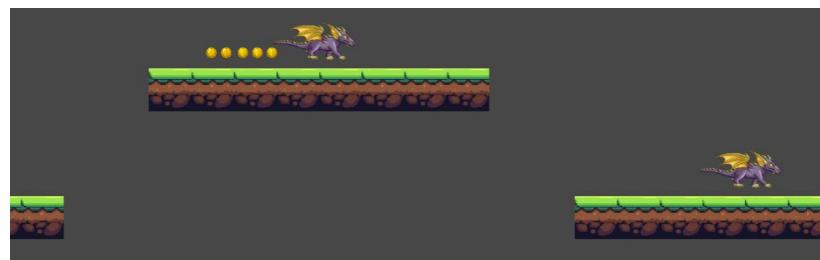
Nachher:



„Abbildung 9: Level 1“

- **Mechaniken:** Einige Levels enthalten kleinere Hindernisse, die nur durch geschicktes Springen und Angreifen überwunden werden können.

**Beispiel 1:** Hindernisse.



„Abbildung 10: Hindernisse aus Level 3“

**Beispiel 2:** Die Anzahl der Gegner ändert sich automatisch nach dem Tod eines Gegners, und der Ninja erhält als Belohnung 10 Münzen und einen Potion.

Vorher:



„Abbildung 11: Coins aus Level 2“

**Nachher:** Der Gegner ist besiegt, und die Anzahl der Gegner ist direkt auf 3 gesunken. Die Anzahl der Münzen ist von 39 auf 49 gestiegen, und der Ninja hat zusätzlich einen Potion erhalten, um seine Lebenspunkte zu erholen.



„Abbildung 12: Münzen sammeln aus Level 2“

- **Health-System:** Während des Spiels kann der Spieler durch das Besiegen von Gegnern oder andere Mechaniken seine Lebenspunkte wieder auffüllen, was den Spielfluss unterstützt.

Beispielsweise hat der Drache mich hier angegriffen und meine Lebenspunkte auf ein niedrigeres Niveau reduziert.



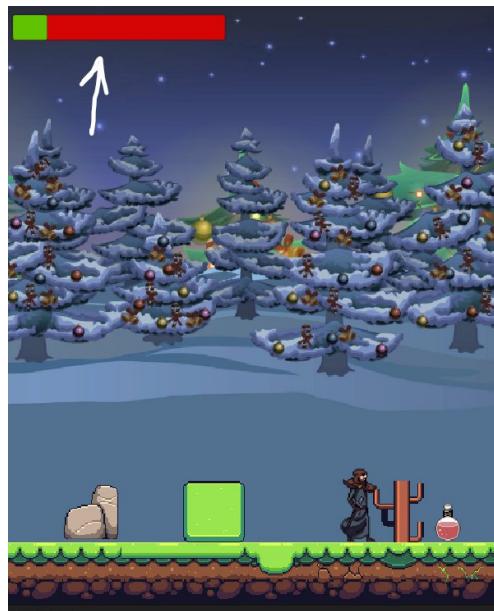
„Abbildung 13: Health-Bar aus Level 4“

Und wenn ich die Potion vom Boden aufhebe, steigen meine Lebenspunkte so an:



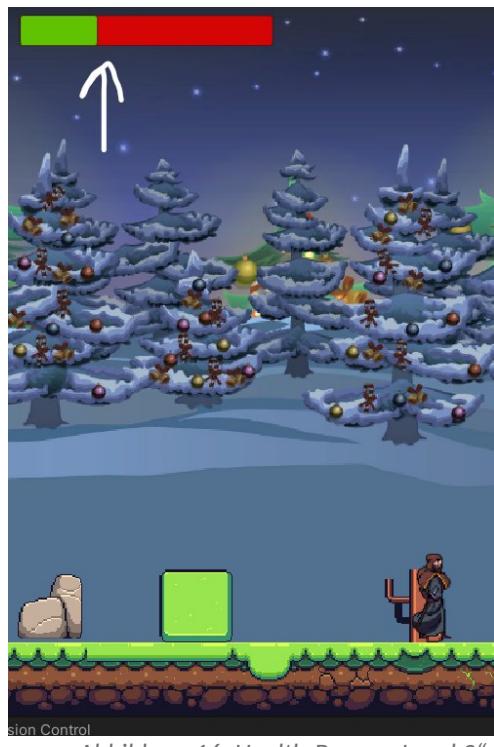
„Abbildung 14: Potion“

Vorher:



„Abbildung 15: Health-Bar aus Level 3“

Nachher:

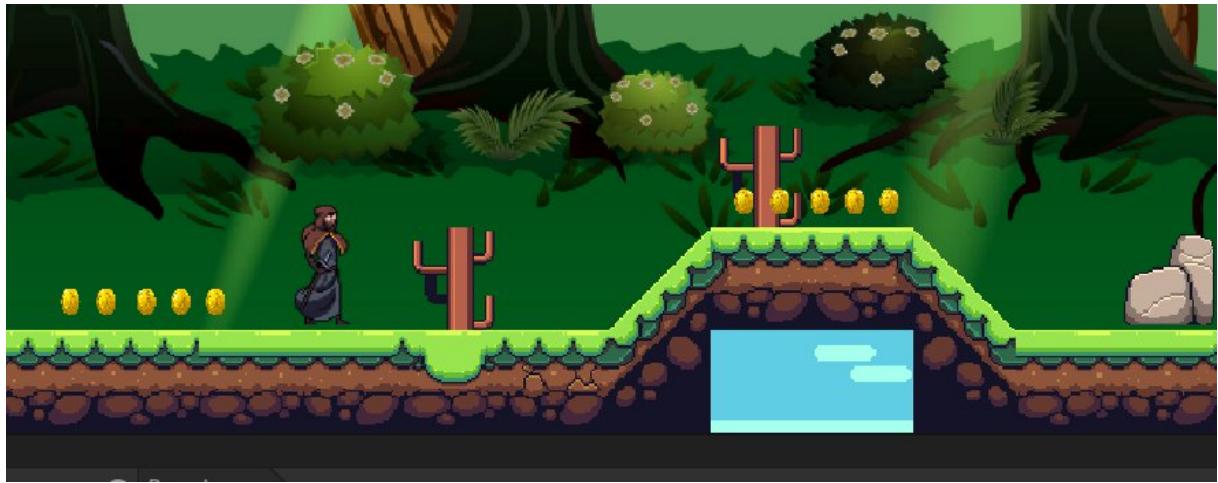


„Abbildung 16: Health-Bar aus Level 3“

### Grafik und Atmosphäre

Jedes Level nutzt eine einzigartige Kombination aus visuellen und akustischen Elementen, um die Spielerfahrung zu intensivieren:

- **Grafik:** Die Levels verwenden Tilemaps, Flüsse und eine illusionäre Höhlenlandschaft, die mit Unity-spezifischen Tools wie Licht- und Partikeleffekten gestaltet wurden.



„Abbildung 17: Grafik aus Level 2“

- **Sounddesign:** Jedes Level verfügt über eine Hintergrundmusik und Soundeffekte, die die Stimmung und Spannung verstärken, insbesondere in Kampfsituationen.

## *Entwicklungsdetails*

### **Tileset: Nature Tiles 01**

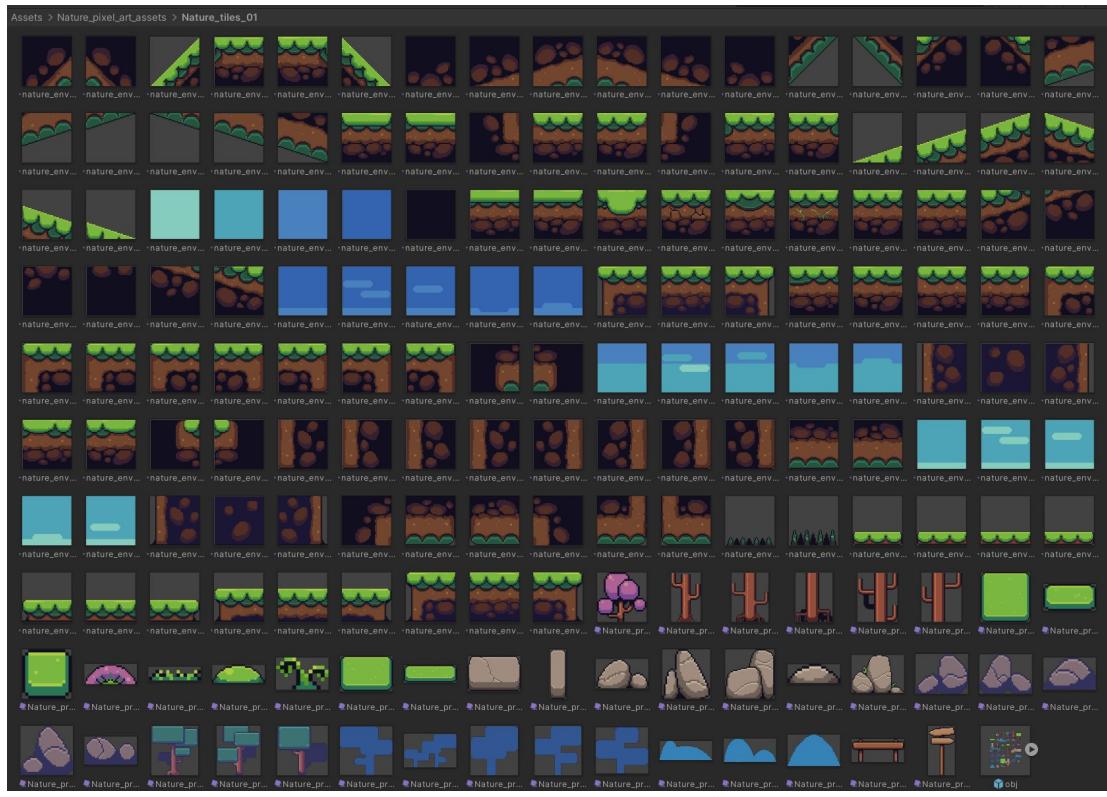
Das Nature Tiles 01-Tileset stellt die Grundlage für die Gestaltung der Spielumgebungen dar. Es handelt sich um ein modular aufgebautes Set von Pixel-Art-Kacheln, die so gestaltet sind, dass sie nahtlos miteinander kombiniert werden können. Dieses Tileset wurde sorgfältig entwickelt, um eine flexible Erstellung von Landschaften zu ermöglichen und gleichzeitig eine einheitliche visuelle Ästhetik sicherzustellen.

Das Tileset umfasst eine Vielzahl von Elementen, die für den Aufbau und die Dekoration der Level entscheidend sind. Die Geländekacheln, die in unterschiedlichen Varianten von Erdböden mit grasbedeckten Oberflächen verfügbar sind, bilden die Grundstruktur der Plattformen. Ergänzt werden diese durch dekorative Elemente wie Bäume, Büsche und Steine, die eine immersive visuelle Tiefe erzeugen. Dynamische Objekte wie Dornen, Kaktusse und Flüssigkeitselemente wie Wasser oder Lava fügen dem Gameplay zusätzliche Herausforderungen und optische Abwechslung hinzu.

Die Integration der Kacheln erfolgt in Unity über das Tilemap-System, das es ermöglicht, große Level effizient zu erstellen und anzupassen. Mit diesem System können Kacheln präzise positioniert werden, was eine flexible Gestaltung der Umgebung erlaubt. Gleichzeitig sorgt die optimierte Performance der Tilemap für eine reibungslose Spielerfahrung, da die Anzahl der einzelnen GameObjects minimiert wird.

Technisch gesehen sind die Kacheln auf eine Größe von 48x48 Pixel ausgelegt, ein Standardmaß, das eine hohe Kompatibilität mit den Unity-Tilemaps gewährleistet. Das Tileset ist im PNG-Format verfügbar, was durch die transparente Darstellung eine problemlose Schichtung ermöglicht. Im Projekt ist das Tileset klar strukturiert und unter Assets/Nature\_pixel\_art\_assets/Nature\_tiles\_01 abgelegt, was einen schnellen Zugriff auf die Ressourcen erlaubt.

Die Gestaltung des Tilesets folgt klaren Designrichtlinien, die eine optimale Nutzung gewährleisten. Die Kacheln sind so konzipiert, dass sie sich optisch ansprechend und nahtlos verbinden lassen, wodurch visuelle Brüche vermieden werden. Gleichzeitig wurde bei der Farbgebung darauf geachtet, dass die Spielfläche gut lesbar bleibt und das Gameplay nicht durch unnötige visuelle Ablenkungen beeinträchtigt wird.

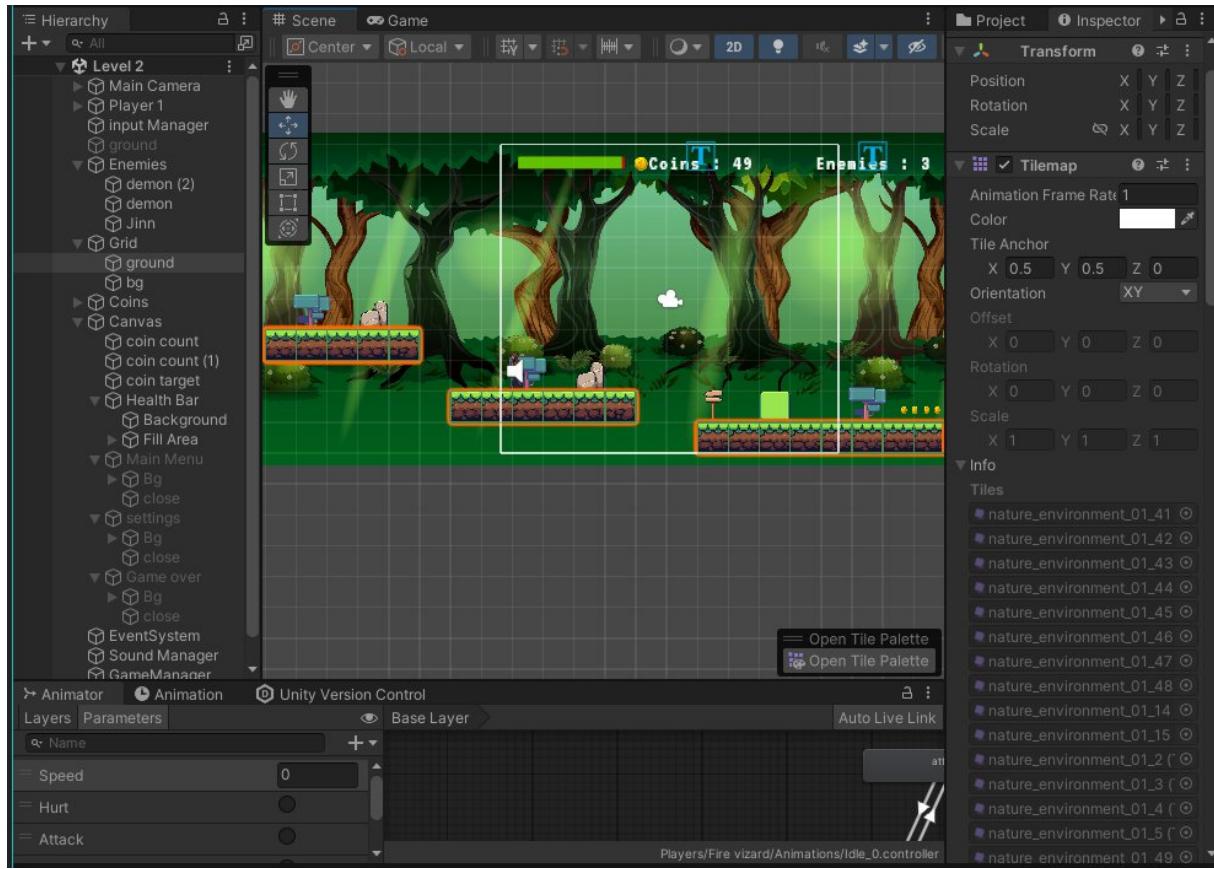


„Abbildung 18: Asset Tilesets“

### Tilemap und Levelstruktur:

Die Tilemap in Super Ninja 2D bildet die Grundlage für die Gestaltung der Level. Im unteren Bild ist die Levelstruktur von Level 2 im Unity-Editor zu sehen. Die einzelnen Elemente, wie Plattformen, Hintergrund und interaktive Objekte, wurden mithilfe der Tilemap und der Nature\_pixel\_art\_assets erstellt. Die Tilemap ermöglicht eine präzise Anordnung von Plattformen und Hindernissen, was sowohl die Funktionalität als auch die visuelle Gestaltung der Levels unterstützt.

Im rechten Bereich des Unity-Editors sind die Eigenschaften der Tilemap sichtbar, einschließlich der verwendeten Tiles aus dem Set Nature\_tiles\_01. Diese Tiles werden in der Szene angeordnet, um eine natürliche und spielbare Umgebung zu schaffen. Zusätzlich wurden interaktive Elemente, wie Münzen, und Gegner wie demon und Jinn strategisch in der Levelstruktur platziert.



„Abbildung 19: Levelstruktur“

Das untere Bild zeigt ein Beispiel, wie die Tiles aus dem Nature\_tiles\_01-Set genutzt wurden, um Plattformen, Übergänge und dekorative Elemente zu erstellen. Die Vielfalt der verfügbaren Tiles ermöglicht es, abwechslungsreiche und detaillierte Umgebungen zu gestalten. Dabei wurde darauf geachtet, dass die Levelstruktur nicht nur optisch ansprechend, sondern auch spielerisch herausfordernd ist.



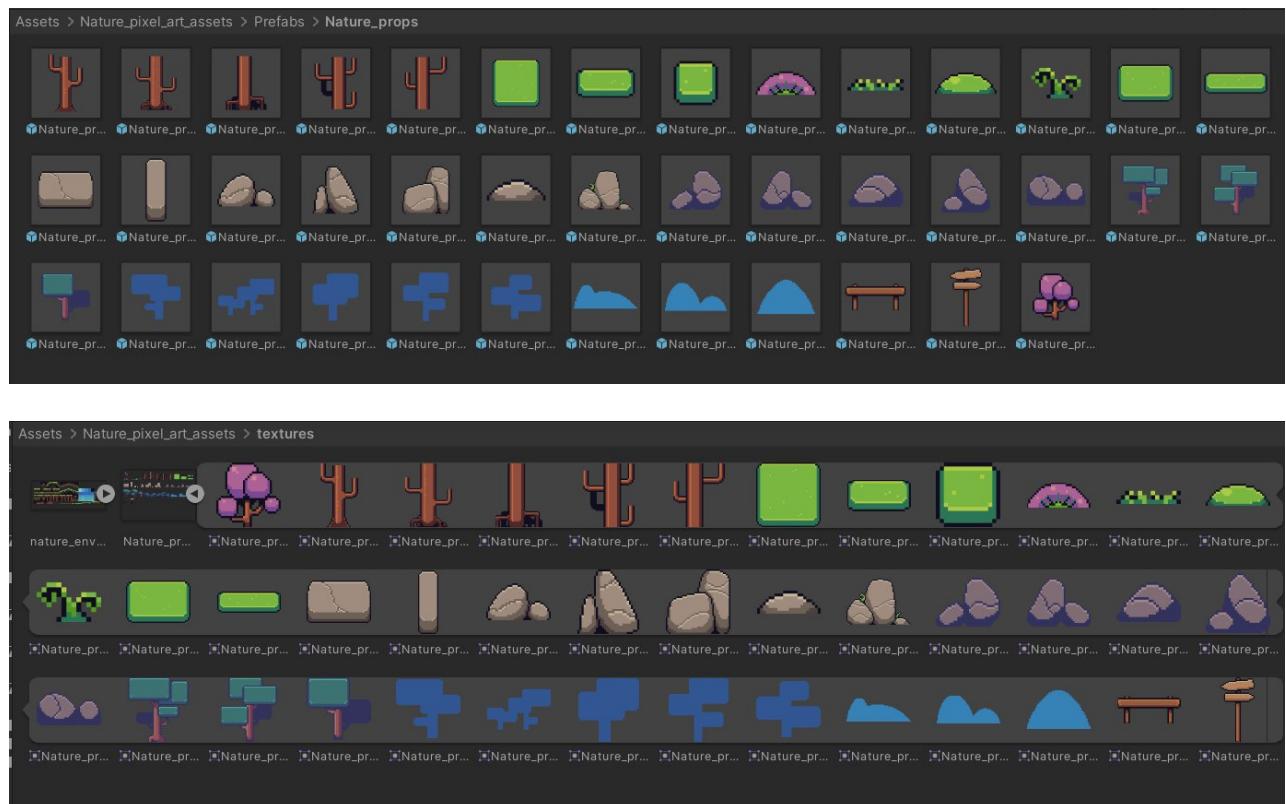
„Abbildung 20: Tiles“

Die gezeigten Elemente stammen aus den "Nature\_pixel\_art\_assets" und umfassen Prefabs und Texturen, die für die Gestaltung der Level in Super Ninja 2D verwendet wurden.

Die unteren zwei Bilder zeigen die Nature Props, welche hauptsächlich dekorative und funktionale Elemente wie Bäume, Felsen, Sträucher und Hinweisschilder beinhalten. Diese Props wurden verwendet, um die Level lebendiger und natürlicher wirken zu lassen, während sie gleichzeitig als Orientierungspunkte für den Spieler dienen. Besonders die Felsen und Bäume sind nicht nur dekorativ, sondern helfen dabei, die illusorische Höhlenlandschaft zu definieren.

Das untere Bild zeigt die Texturen, die als Bausteine für Plattformen, Übergänge und Hintergrundelemente dienen. Diese Texturen werden in Kombination mit den Tiles aus der Tilemap genutzt, um ein nahtloses und optisch ansprechendes Design zu gewährleisten. Sie unterstützen die Schaffung einer immersiven Umgebung, die den Spieler sowohl visuell anspricht als auch herausfordert.

Die Kombination dieser Assets ermöglicht es, Levels abwechslungsreich und detailreich zu gestalten, wobei sowohl funktionale als auch ästhetische Anforderungen berücksichtigt werden.



„Abbildung 21: Natur Props“

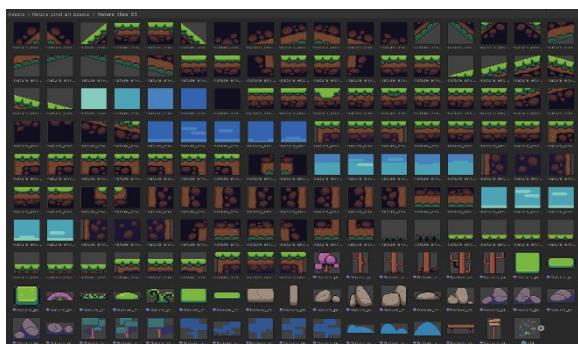
# Assets

## fremde Assets

Für die Entwicklung unseres Spiels "Super Ninja 2D" haben wir ausschließlich Assets aus dem Unity Asset Store verwendet. Dies hat uns ermöglicht, hochwertige Ressourcen effizient einzusetzen und uns auf die Programmierung und das Level-Design zu konzentrieren.

### o Natur-Pixel-Art-Tileset

Das Tileset wurde für die Gestaltung der Plattformen und Wände in allen vier Levels verwendet. Für die Leveln wurde ein Hintergrundbild ausgewählt.



„Abbildung 22: Tileset“



„Abbildung 23: Background“

### o Player

Der Spieler übernimmt die Rolle eines mutigen Ninjas, der durch die Höhlenwelt navigiert. Der Ninja hat die Fähigkeit zu springen, zu kämpfen und Münzen zu sammeln.



„Abbildung 24: Player“

### o Gegner (Enemy)

wir haben drei Arten von Gegner verwendet, um die Herausforderung das Spiel zu erhöhen:

- **Jin:** Ein schneller Nahkampf-Gegner.
- **Drache:** Der mächtige Endgegner mit zerstörerischen Angriffen.
- **Dämon:** Ein fliegender Fernkampf-Gegner mit magischen Fähigkeiten.
- Zusätzlich zu den Gegnern haben wir ein spezielles **Potion** verwendet, das als Belohnung für das Besiegen von Gegnern dient.

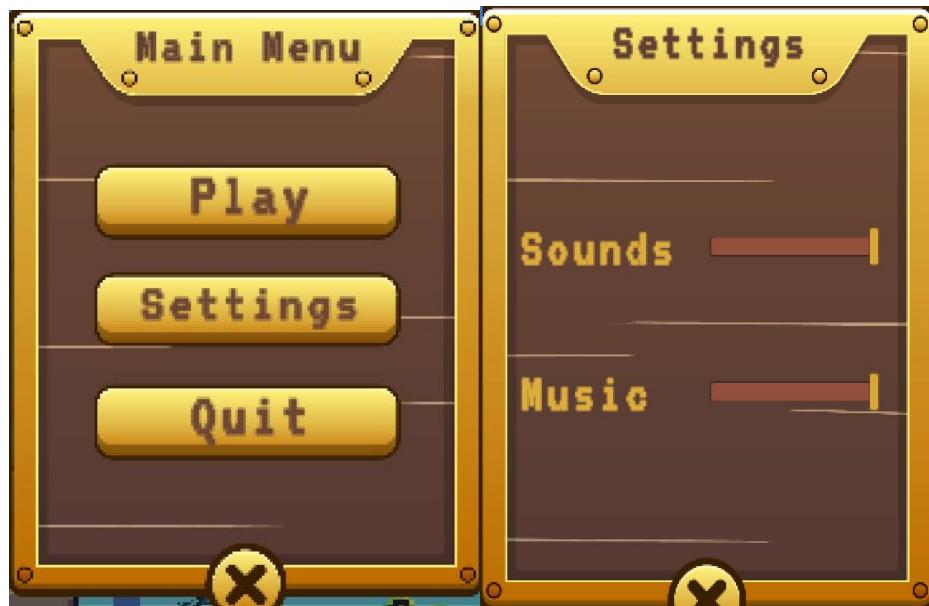


„Abbildung 25: Enemies“

„Abbildung 26: Potionen“

## O Spiel Menü

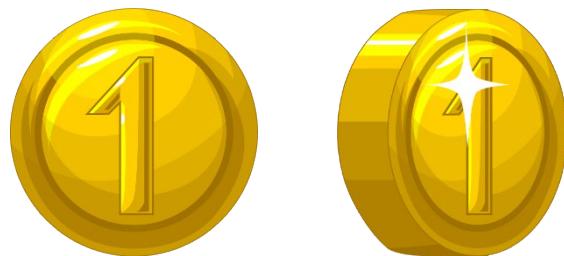
Das Hauptmenü bietet dem Spieler eine einfache und klare Navigation, um das Spiel zu starten oder Einstellungen vorzunehmen.



„Abbildung 27: Menu“

## O Münzen

Münzen sind sammelbare Objekte, die der Spieler aufsammeln kann.



„Abbildung 28: Coins“

## O Health-Bar

Diese Leiste zeigt den aktuellen Gesundheitsstatus des Spielers an und verändert sich dynamisch.



„Abbildung 29: Health-Bar“

## O Flagge

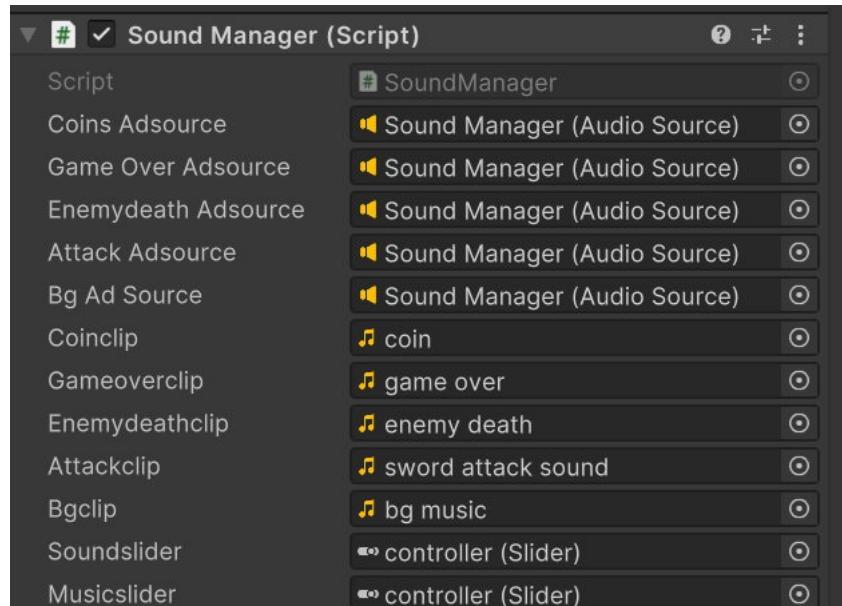
Am Ende jedes Levels gibt es eine Flagge, die das Ziel symbolisiert. Der Spieler kann jedoch erst zur Flagge gelangen und das nächste Level freischalten, nachdem alle Gegner im aktuellen Level besiegt wurden.



„Abbildung 30: Flag“

## O Soundeffekte und Musik

Hintergrundmusik und Geräusche wie Münzen sammeln, Gegnerangriffe und die Sterben.



„Abbildung 31: SoundManger“

# Projektübersicht

## Team

Unser Team besteht aus Amanullah Naderi, Mohammad Taiba und Eman Al-Mesri.

## Arbeitsweise.

### 1. Kommunikation und Organisation

Für die effiziente Zusammenarbeit in unserem Team haben wir **Discord** und **Trello** genutzt:

- **Discord:**

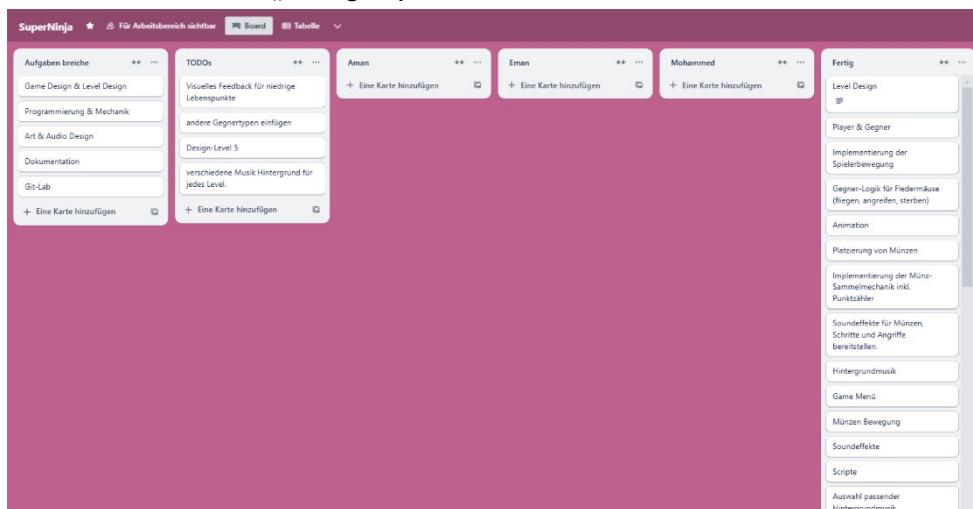
- Wir haben Discord für die Kommunikation in Echtzeit verwendet.
- Wöchentliche Teamtreffen fanden in einem gemeinsamen Sprachkanal statt, um Aufgaben zu besprechen und Fortschritte zu präsentieren.
- Zusätzlich nutzen wir den Chat, um Fragen zu klären, Ideen auszutauschen und wichtige Informationen zu teilen.

- **Trello:**

- **Trello-Board-Struktur:**

Unser Trello-Board bestand aus fünf Spalten (Tickets):

1. **To-Do:** Enthielt allgemeine übersicht.
2. **Aufgabenliste:** Enthielt alle Aufgaben, die erledigt werden mussten.
3. **Teammitglieder-Tickets:** Jeder von uns hatte eine eigene Spalte mit seinem Namen, um persönliche Aufgaben zu verwalten.
4. **Fertig:** Nach Abschluss einer Aufgabe wurde diese in die „Fertig“-Spalte verschoben.



„Abbildung 32: Arbeitsaufteilung Trello“

## 2. Aufgabenverteilung

- **Wöchentliche Treffen:**

Wir trafen uns jede Woche, um die aktuellen Aufgaben zu besprechen und neue Aufgaben zu verteilen. Jeder präsentierte seinen Fortschritt und erhielt Feedback von den anderen Teammitgliedern.

- **Aufgabenverteilung:**

- o Während der Meetings haben wir neue Aufgaben gemeinsam erstellt und den jeweiligen Personen zugewiesen.
- o Sobald jemand seine Aufgaben abgeschlossen hatte, wurden diese in die „Fertig“-Spalte verschoben, und derjenige übernahm neue Aufgaben aus der - Aufgabenliste.

| Person           | Themengebiete   |
|------------------|---|
| Amanullah Naderi | { Basis spiel<br>{ Level-Design für Level 1 und 4<br>{ Player-Logik<br>{ Soundeffekte<br>{ Testing die Levels |
| Mohammad Taiba   | { Level-Design für Level 2<br>{ Gegner-Logik<br>{ Kamerasteuerung<br>{ Soundeffekte<br>{ Dokumentation        |
| Eman Al-Mesri    | { Level-Design für Level 3<br>{ Menü und UI<br>{ Gameplay-Mechaniken<br>{ Soundeffekte<br>{ Dokumentation     |

## Offene ToDos

- **Visuelles Feedback für niedrige Lebenspunkte** – Blinken des Health Bars oder ein Warnton.
- **Design-Level 5**
- **Mehr Gegnertypen einfügen** – Unterschiedliche Gegner mit eigenen Angriffsmustern und Verhalten.
- verschiedene **Musik Hintergrund** für jedes Level.

## Ideen für Weiterentwicklung

- Ein **Pausensymbol**, um das Spiel während des Spielens anzuhalten.
- Eine **Lebensanzeige (Health Bar) für die Gegner**, damit der Spieler sieht, wie viel Schaden sie noch aushalten.
- **Neue Power-Ups**: z. B. ein Schild, das den Spieler für kurze Zeit unverwundbar macht.
- **Individuelle Soundeffekte für verschiedene Gegnertypen**, um das Spielerlebnis realistischer zu gestalten.
- **Tag-Nacht-Zyklus**: Das Spiel verändert sich je nach Tageszeit.

# Code-Ausschnitte

*Name des Scripts: PlayerControllers.cs*

**Beschreibung:** Das PlayerController-script steuert die grundlegenden Bewegungen und Aktionen des Spielers in einem 2D-Spiel. Es verarbeitet Spielerbewegung, Sprünge, Angriffe, Animationen und stellt sicher, dass der Spieler nicht aus der Spielwelt fällt.

```
// Handle jumping input
if (inputManager.GetJumpInput())
{
    if (jumpsLeft > 0)
    {
        rb.velocity = new Vector2(rb.velocity.x, jumpForce);
        jumpsLeft--;
        audioSource.PlayOneShot(jumpSound);
    }
}
```

„Abbildung 33: PlayerControllers“

**Beschreibung des Code-Ausschnitts:**

Dieser Code ermöglicht es dem Spieler zu springen.

- **Springen wird nur ausgelöst**, wenn die Sprungtaste gedrückt wird (`inputManager.GetJumpInput()`).
- Falls noch **Sprünge verfügbar sind** (`jumpsLeft > 0`), wird der Spieler nach oben bewegt (`rb.velocity`).
- Nach jedem Sprung wird die **Anzahl der verbleibenden Sprünge reduziert**.
- Dadurch kann ein **Doppelsprung oder Mehrfachsprung** ermöglicht werden, falls `maxJumps > 1` gesetzt ist.

```
void Update()
{
    if(isdead)
    {
        return;
    }
    float distance = Vector2.Distance(transform.position, player.position);

    if (distance <= attackRange)
    {
        if (!isAttacking)
        {
            StartCoroutine(Attack());
        }
    }
    else if (distance <= chaseRange)
    {
        Chase();
    }
    else
    {
        Patrol();
    }
}
```

### *Name des Scripts: EnemyControllers.cs*

**Beschreibung:** Das EnemyController-Script steuert das Verhalten von Gegnern in einem 2D-Spiel. Es ermöglicht Gegnern, zu patrouillieren, den Spieler zu verfolgen, anzugreifen und Schaden zu nehmen. Zudem sorgt es für Animationen und Soundeffekte bei Angriffen und dem Tod des Gegners.

„Abbildung 34: EnemyControllers“

### **Beschreibung des Code-Ausschnitts:**

Diese [Update\(\)-Methode](#) steuert das Verhalten des Gegners in jedem Frame:

1. **Überprüfung, ob der Gegner tot ist:**
  - o Falls `isdead == true`, wird die Methode gestoppt.
2. **Distanzberechnung zum Spieler:**
  - o `Vector2.Distance()` berechnet den Abstand zwischen Gegner und Spieler.
3. **Angriff:**
  - o Falls der Spieler in Reichweite (`attackRange`) ist und der Gegner nicht bereits angreift, wird die [Attack\(\)-Routine](#) gestartet.
4. **Verfolgung:**
  - o Falls der Spieler sich in der **Chase-Range** (`chaseRange`) befindet, beginnt der Gegner, sich in Richtung des Spielers zu bewegen ([Chase\(\)](#)).
5. **Patrouillieren:**
  - o Falls der Spieler nicht in der Nähe ist, bewegt sich der Gegner auf einer festgelegten Strecke hin und her ([Patrol\(\)](#)).

### *Name des Scripts: ObjectSpawner.cs*

**Beschreibung:** Das ObjectSpawner-Script sorgt für das zufällige Spawning von Objekten zwischen zwei definierten Punkten. Die Objekte werden in regelmäßigen, aber zufälligen Zeitintervallen generiert. Dies kann für Gegner, Power-ups oder Hindernisse verwendet werden.

```
void Update()
{
    // Check if it's time to spawn a new object
    if (Time.time >= nextSpawnTime)
    {
        SpawnObject();
        // Set the next spawn time within the specified interval
        nextSpawnTime = Time.time + Random.Range(minSpawnInterval, maxSpawnInterval);
    }
}
```

„Abbildung 35: ObjectSpawner“

### Beschreibung des Code-Ausschnitts:

1. Zeitprüfung für das Spawning:
  - o `Time.time` überprüft, ob der aktuelle Zeitpunkt die geplante Spawn-Zeit (`nextSpawnTime`) erreicht hat.
2. Objekt-Spawning:
  - o Falls der Zeitpunkt erreicht ist, wird `SpawnObject()` aufgerufen, um ein neues Objekt zu erstellen.
3. Neue Spawn-Zeit festlegen:
  - o `nextSpawnTime` wird mit einem zufälligen Wert zwischen `minSpawnInterval` und `maxSpawnInterval` aktualisiert, um das nächste Erscheinen des Objekts unregelmäßig zu gestalten.

### Name des Scripts: PlayerHealth.cs

**Beschreibung:** Das `PlayerHealth`-Script verwaltet die Gesundheit des Spielers und steuert die Reaktionen auf Schaden und Heilung. Es bietet Funktionen für Heilung durch Tränke, Schadensannahme und den Tod des Spielers. Zudem wird eine Gesundheitsanzeige (Health Bar) aktualisiert, um die verbleibende Gesundheit anzuzeigen.

```
public void TakeDamage(int damage)
{
    currentHealth -= damage;
    // Debug.Log("Player took " + damage + " damage. Current health: " + currentHealth);
    HealthBar.value = currentHealth;
    if (currentHealth <= 0)
    {
        if (SoundManager.Instance != null)
        {
            SoundManager.Instance.PlayGameOverSound();
        }
        Die();
    }
    else
    {
        animator.SetTrigger("Hurt");
    }
}
```

„Abbildung 36: PlayerHealth“

### Beschreibung des Code-Ausschnitts:

1. Schaden annehmen:
  - o Die Methode `TakeDamage(int damage)` wird aufgerufen, wenn der Spieler Schaden erleidet. Der Wert der aktuellen Gesundheit (`currentHealth`) wird um den angegebenen **Schaden** verringert.
2. Aktualisierung der Gesundheitsanzeige:

- o Die **Health Bar** wird nach jedem Schaden aktualisiert, sodass die verbleibende Gesundheit visuell angezeigt wird.

### 3. Tod des Spielers:

- o Wenn die **Gesundheit des Spielers auf 0 oder weniger fällt**, wird die Methode [Die \(\)](#) aufgerufen, und der Sound für das **Spielende** wird abgespielt.

### 4. Verletzungsanimation:

- o Wenn der Spieler Schaden erleidet, aber noch lebt, wird eine **Verletzungsanimation** ([Trigger "Hurt"](#)) abgespielt, um visuelles Feedback zu geben.

## Anhang

- Tools

| Tools             | Link  |
|-------------------|---|
| Unity             | <a href="https://unity.com/download">https://unity.com/download</a>                               |
| Visual Studio     | <a href="https://visualstudio.microsoft.com/de/vs/">https://visualstudio.microsoft.com/de/vs/</a> |
| Visual code       | <a href="https://code.visualstudio.com/">https://code.visualstudio.com/</a>                       |
| Trello            | <a href="https://trello.com/">https://trello.com/</a>   |
| Unity Asset Store | <a href="https://assetstore.unity.com/">https://assetstore.unity.com/</a>                         |

- Asset

| Asset                   | Link   |
|-------------------------|--|
| Natur-Pixel-Art-Tileset | <a href="https://assetstore.unity.com/packages/2d/environments/nature-pixel-art-base-assets-free-151370">https://assetstore.unity.com/packages/2d/environments/nature-pixel-art-base-assets-free-151370</a>  |
| Player                  | <a href="https://www.behance.net/gallery/160947935/Free-Wizard-Sprite-Sheets-Pixel-Art?locale=nb_NO">https://www.behance.net/gallery/160947935/Free-Wizard-Sprite-Sheets-Pixel-Art?locale=nb_NO</a>  |
| Enemy                   | <a href="https://opengameart.org/content/pixel-art-rpg-monster-sprites">https://opengameart.org/content/pixel-art-rpg-monster-sprites</a>  |
| Menu / Bonus / Door     | <a href="https://ca.pinterest.com/pin/free-vector-42502790225557511/">https://ca.pinterest.com/pin/free-vector-42502790225557511/</a><br><a href="https://www.deviantart.com/eviscus/art/Some-Potions-413459093">https://www.deviantart.com/eviscus/art/Some-Potions-413459093</a> |

|                     |   |
|---------------------|---|
|                     | <a href="https://opengameart.org/content/animated-bamboo-door-sprite">https://opengameart.org/content/animated-bamboo-door-sprite</a>                             |
| <b>Coins</b>        | <a href="https://gameify-07c652175e42.herokuapp.com/">https://gameify-07c652175e42.herokuapp.com/</a>   |
| <b>Flag</b>         | <a href="https://www.kindpng.com/imgv/wboRRw_mario-flag-pixel-art-hd-ong-download/">https://www.kindpng.com/imgv/wboRRw_mario-flag-pixel-art-hd-ong-download/</a> |
| <b>Soundeffekte</b> | <a href="https://pixabay.com/sound-effects/search/">https://pixabay.com/sound-effects/search/</a>   |

- **Link**

|                 | <b>Link</b>   |
|-----------------|---|
| <b>Git-Lab</b>  | <a href="https://git.ai.fh-erfurt.de/em4167al/super-ninja-2d">https://git.ai.fh-erfurt.de/em4167al/super-ninja-2d</a>   |
| <b>Tutorial</b> | <a href="https://youtube.com/playlist?list=PLgOEwFbvGm5o8hayFB6skAfa8Z-mw4dPV&amp;si=l935T1lepjxg69pO">https://youtube.com/playlist?list=PLgOEwFbvGm5o8hayFB6skAfa8Z-mw4dPV&amp;si=l935T1lepjxg69pO</a> |