Dokumentation Dungeon Explorer

Von Sanjivan Selvarasa

Sanjivan Selvarasa
Oberhofstrasse 41
6020 Emmenbrücke
Sanjivan_selvarasa@sluz.ch

Inhaltsverzeichnis

1. Summary

Mein Ziel war es, ein Roguelike-Spiel in Top-Down-Perspektive zu entwickeln, das sich durch eine prozedural generierte Map, verschiedene Gegnerwellen, ein animiertes UI und einen Highscore auszeichnet. Das Spiel sollte Spass machen, visuell ansprechend sein und grundlegende Spielmechaniken wie Angriff, Bewegung, Level-Up und Wellenfortschritt beinhalten. Zusätzlich wollte ich Technologien wie ScriptableObjects, A*-Pathfinding und Unit-Tests sinnvoll einsetzen.

Ich habe mein Ziel erreicht. Das Spiel ist vollständig spielbar, die Map wird bei jeder Welle neu generiert und die Gegner greifen den Spieler intelligent an. Der Spieler hat ein Health- und XP-System, es gibt ein Menü mit Startscreen und Deathscreen, und ein funktionierendes Highscore-System mit Langzeitspeicherung. Zudem sind Unit-Tests integriert, die bestimmte Funktionalitäten automatisiert prüfen.

Besonders gut lief die Einarbeitung in Unity und C#. Ich konnte durch gezielte Tutorials schnell lernen, wie man mit Tilemaps, UI-Elementen und Player Controls umgeht. Auch die Umsetzung von ScriptableObjects war hilfreich, um Objektdaten wie Gegnerwerte strukturiert zu speichern. Die Arbeit in einzelnen Teilprojekten (z. B. für KI oder Map-Generierung) hat sich ebenfalls gelohnt, weil ich so effizienter debuggen konnte und externe Fehlerquellen eliminieren konnte.

Weniger gut lief die Einbindung grosser Systeme wie der A*-Algorithmus oder die Map-Generierung in das Hauptprojekt. Ausserdem war die Entwicklung dieser Systeme sehr aufwendig und zeitintensiv, was ich davor nicht so miteingeplant hatte. Es gab viele unerwartete Fehler mit bestehenden Scripts. Auch die GitHub-Veröffentlichung scheiterte, da manche Dateien zu gross waren. Teilweise habe ich dadurch Zeit verloren, die ich lieber für Feinschliff und Tests verwendet hätte.

Beim nächsten Mal würde ich früher mit einem Versionskontrollsystem wie Git arbeiten und den Code von Anfang an modularer schreiben. Auch würde ich häufiger testen, statt viele Änderungen auf einmal zu machen. Zudem würde ich den Aufbau der UI früher planen, da die spätere Einbindung über mehrere Szenen hinweg sehr aufwändig war. Trotzdem bin ich mit dem Endergebnis sehr zufrieden und habe in kurzer Zeit viel über Game-Development gelernt.

1.1 Reflektion

Technischer Lerngewinn
In diesem Projekt habe ich mein technisches Wissen über Unity und C# deutlich vertieft und verbessert. Besonders der Umgang mit ScriptableObjects, das Einbinden von UI-Elementen und das Verständnis von Unit Tests haben mir sehr geholfen ein besseres Verständnis in der Spiele Entwicklung zu erhalten. Ich habe gelernt, wie man Klassen strukturiert aufbaut, wie man zwischen mehreren Szenen sinnvoll navigiert und wie man Daten (zb. Highscores) langfristig speichert. Der A*-Algorithmus war eine besondere Herausforderung, aber ich konnte ein funktionierendes System implementieren, das ich jetzt auch in anderen Projekten anwenden kann.

Persönlicher Lerngewinn
Ich habe gelernt, strukturiert und selbstständig an einem grösseren Projekt zu arbeiten. Besonders die Aufteilung in einzelne Teilprojekte hat mir geholfen, den Überblick zu behalten. Ich habe gemerkt, dass ich mit Planung und Fokus viel mehr schaffen kann, als ich zu Beginn

gedacht habe. Auch beim Debuggen habe ich mehr Geduld entwickelt und erkannt, wie wichtig systematisches Vorgehen ist. Im Nachhinein habe ich gemerkt, dass ich kreativer und lösungsorientierter und ein besserer Programmierer geworden bin.

Konkrete Verbesserungsvorschläge für zukünftige Projekte

- **Früher mit Git arbeiten:** Ich habe zu spät angefangen, mein Projekt versionsbasiert zu sichern. Nächstes Mal werde ich gleich von Anfang an mit Git arbeiten, um Änderungen nachverfolgen und Fehler einfacher rückgängig machen zu können.
- **UI-Konzept vorher planen:** Die UI habe ich zu spät als komplettes Bauelement eines Spieles betrachtet. Beim nächsten Projekt mache ich zuerst Skizzen und entscheide dann, wie die Logik aufgeteilt werden soll.
- **Testen während der Entwicklung:** Ich habe oft zu viel Code geschrieben, ohne ihn direkt zu testen. Das hat später zu Fehlern geführt, die schwer nachvollziehbar waren. Daher werde ich zukünftige Features immer direkt nach der Implementierung testen.
- **Dateigrössen im Blick behalten:** Die Veröffentlichung auf GitHub ist an einer grossen Datei gescheitert. Ich werde künftig darauf achten, welche Dateien zwingend ins Repository gehören und welche nicht.

Zusammenfassung

Ich bin sehr zufrieden mit dem Ergebnis. Ich konnte viel Wissen aufbauen, mein können verbessern und ein funktionierendes, spielbares Game abgeben. Trotzdem sehe ich viele Punkte, wo ich effizienter und sauberer hätte arbeiten können. Es hat viel Spass gemacht und ich werde dieses Wissen für spätere Projekte gut anwenden können.

2. Grobziele

2.1 Projektorientierte Ziele

- Ich erstelle ein Roguelike Spiel von einer Top-Down Ansicht, dass einwandfrei funktionieren soll und Spass macht.
- Das Spiel soll mehrere Wellen haben an Gegnern, die den Spieler angreifen.
- Der Spieler sollte von Gegnern verletzt werden und demnach Schaden kriegen und sterben können
- Wenn man das Spiel startet, soll eine prozentual Generierte Map erstellt werden, die bei jedem durchlauf anderes ist.
- Das Spiel soll ansehnlich sein und lebendig wirken.
- Jeder Gegner und Spieler soll animiert sein in Geh-, Idle-, und Angriff Animationen.
- Das Spiel soll einen Title Screen haben mit einem kleinem Menu, wodurch man zum Spiel kommt, wenn genug Zeit ist werden da noch paar Einstellmöglichkeiten implementiert.

2.2 Userstories

Priorität	Business Value	Titel des Features	User Story
1	950	Startseite	Als Spieler will ich das Spiel über einen Startbildschirm starten können, um in das eigentliche Spiel zu gelangen.
2	900	Gegner Wellen	Als Spieler möchte ich gegen Wellen von Gegnern kämpfen, um eine Herausforderung und ein Ziel im Spiel zu haben.
3	850	Health & XP Stats	Als Spieler will ich jederzeit meine Lebenspunkte und meinen Erfahrungsstand sehen können, um meine aktuelle Spielsituation einzuschätzen.
4	800	New Generation	Als Spieler will ich bei jeder Runde eine neu generierte Map erhalten, um Abwechslung und Wiederspielwert zu erleben.
5	780	Gegner AI	Als Spieler will ich, dass Gegner mich gezielt verfolgen, um eine spannende und realistische Herausforderung zu haben.
6	700	Highscore	Als Spieler will ich meinen Highscore speichern und anzeigen lassen können, um mich mit meinen früheren Leistungen zu vergleichen.
7	680	Upgrades	Als Spieler will ich nach paar Wellen ein Upgrade auswählen können, um meinen Charakter zu verbessern.

2.3 Akzeptanzkriterien

1.)

- Startscreen mit “Start”-Button vorhanden
- Spiel beginnt nach Klick auf Start
- **Grobaufwand: 3 SP**

2.)

- Wellen erscheinen automatisch
- Anzeige für verbleibende Gegner sichtbar
- Gegnerzahl pro Welle steigt mit dem Fortschritt
- **Grobaufwand: 5 SP**

3.)

- Healthbar reduziert sich bei Schaden
- XP-Leiste füllt sich bei Gegnerbesiegung
- Leisten sind sichtbar und funktionieren
- **Grobaufwand: 4 SP**

4.)

- Map ist jedes Mal anders
- Kein Softlock oder unpassierbare Gebiete
- Spawnpoint immer erreichbar
- **Grobaufwand: 8 SP**

5.)

- Gegner verwenden Pfadfindung
- Umgehen Hindernissen korrekt
- Bleiben nicht hängen
- **Grobaufwand: 6 SP**

6.)

- Highscore wird gespeichert (lokal)
- Neuer Highscore wird erkannt
- Highscore bleibt nach Neustart erhalten
- **Grobaufwand: 5 SP**

7.)

- Upgrade-Auswahl erscheint nach paar Wellen
- Auswahl beeinflusst Gameplay zb. mehr Schaden
- Auswahl ist permanent für das Spiel
- **Grobaufwand: 6 SP**

3. Planung

Projekt Planung

ZIELE	Soll Aufwand (Stunden)	Ist Aufwand (Stunden)	Aufgabe Status Geschafft / in Arbeit	Bemerkungen
Einstieg	2.5	3.0		
Welches Engine ist geeignet?	0.5	0.2	Erreicht	Unity
Einstieg in Unity	2.0	3.0	Erreicht	
Spieler erstellen	3.0	7.0		
Visuallisierung Spielfigur	0.5	1.0	Erreicht	Ritter Spielfigur
Bewegung der Spielfigur	1.0	3.0	Erreicht	Rigidbody Methode
Logik Spielfigur (Collider)	1.0	1.0	Erreicht	
Animationen erstellen	1.0	2.0	Erreicht	
Health System Einfügen	9.0	12.0		
Health Bar Spieler	2.0	1.0	Erreicht	
Health System Gegner	2.0	4.0	Erreicht	
Logik Health System	4.0	6.0	Erreicht	
Health System UI	1.0	1.0	Erreicht	
Gegner erstellen	3.0	4.0		
Visuallisierung Gegner	1.0	1.0	Erreicht	
Logik Gegner (Collider)	1.0	1.0	Erreicht	
Animationen erstellen	1.0	2.0	Erreicht	
Gegner KI hinzufügen (A*)	12.5	41.0		
beste Option finden	0.5	0.5	Erreicht	
Algorithymus programmieren	8.0	35.0	Erreicht	komplexer als Erwartet
Testen	2.0	2.0	Erreicht	
Debugen	1.0	1.5	Erreicht	
ins Original Spiel implementieren	1.5	2.0	Erreicht	
Prozedural Map Generation	18.0	48.5		
Normale Statische Map erstellen	2.0	3.5	Erreicht	
best. Case Map definieren	0.5	2.5	Erreicht	
Prozedural Map programmieren	10.0	38.0	Erreicht	komplexer als Erwartet
Testen	4.0	3.0	Erreicht	
Debugen	2.0	1.5	Erreicht	
Gegner Wellen erstellen	3.0	6.0		
programmieren	3.0	6.0	Erreicht	
Player Leveling	8.0	6.0		
Logik levelling	5.0	5.0	Erreicht	
Effekte	3.0	1.0	Erreicht	
UI erstellen	5.0	1.3		
Gegner Wellen UI	1.0	0.1	Erreicht	
Player Health Bar UI	1.0	0.1	Erreicht	
Player Leveling UI	1.0	0.1	Erreicht	
andere	2.0	1.0	Erreicht	
Startbildschirm	9.0	3.0		
Prototyp erstellen	2.0	1.0	Erreicht	
Logik implementieren	4.0	1.5	Erreicht	
Finaler Startscreen	3.0	0.5	Erreicht	
Restliche Logik	2.0	1.0		
Restart	2.0	1.0	Erreicht	
Summary in Stunden	89.5	132.8		

4. Benutzerhandbuch

4.1 Einleitung

Ziel: Das Ziel vom Spiel ist es, dass du so viele Wellen an Gegnern besiegst wie möglich, um so weit zu kommen, wie du kannst, um einen neuen Highscore aufzustellen. Die Wellen werden dabei immer schwerer und du bist dich upgraden, um standzuhalten.

1. Wenn die Applikation geöffnet wird, dann sieht du die Startseite

2. Auf der Startseite gibt es verschiedene Optionen (Start, Einstellungen)

3. Wenn du alles eingestellt hast, klickst du auf Start

4. Dann landest du auf einer zufällig generierten Map.

5. Unten siehst du die Health Bar und eine Experience Leiste. Die Health Bar zeigt dein aktuelles Leben und die Experience Leiste zeigt an, wie ob du ein Level Up bekommen hast oder wie viel du noch brauchst.

6. Oben von der UI siehst du eine grosse, lange Leiste, die zeigt an in welcher Welle du gerade bist und wie viele Gegner noch auf der Map sind die du noch besiegen musst.

7. Unten links siehst du verschiedene Upgrade Möglichkeiten, die du später im Spiel auswählen kannst, um deinen Charakter zu verbessern.

8. Aktuell bist du auf Level 1, das heisst du musst Gegner mit dem Schwert bzw. jeder anderen Waffe die dir aktuell zur verfügen steht besiegen.

9.Du musst aber nicht alle Gegner besiegen um weiterzukommen, nur beinahe alle.

10. Wenn du das getan hast, hast du jetzt mehr XP und kannst jetzt ein Upgrade auswählen in der unteren linken Ecke.

11.Dann musst du zurück zum Spawnpoint um die neue Welle zu starten.

12. Jetzt lädt sich eine neue Gegner Welle und die neu generierte Map auf, die du schaffen musst, um weiterzukommen.

13. Auf der Map spawnen manchmal Kisten, die dir Belohnungen geben.

14. Also jetzt weisst du alles, was du wissen musst, besiege so viele Gegner wie möglich und stelle einen neuen Highscore auf. Viel Glück!

4.2 Controls

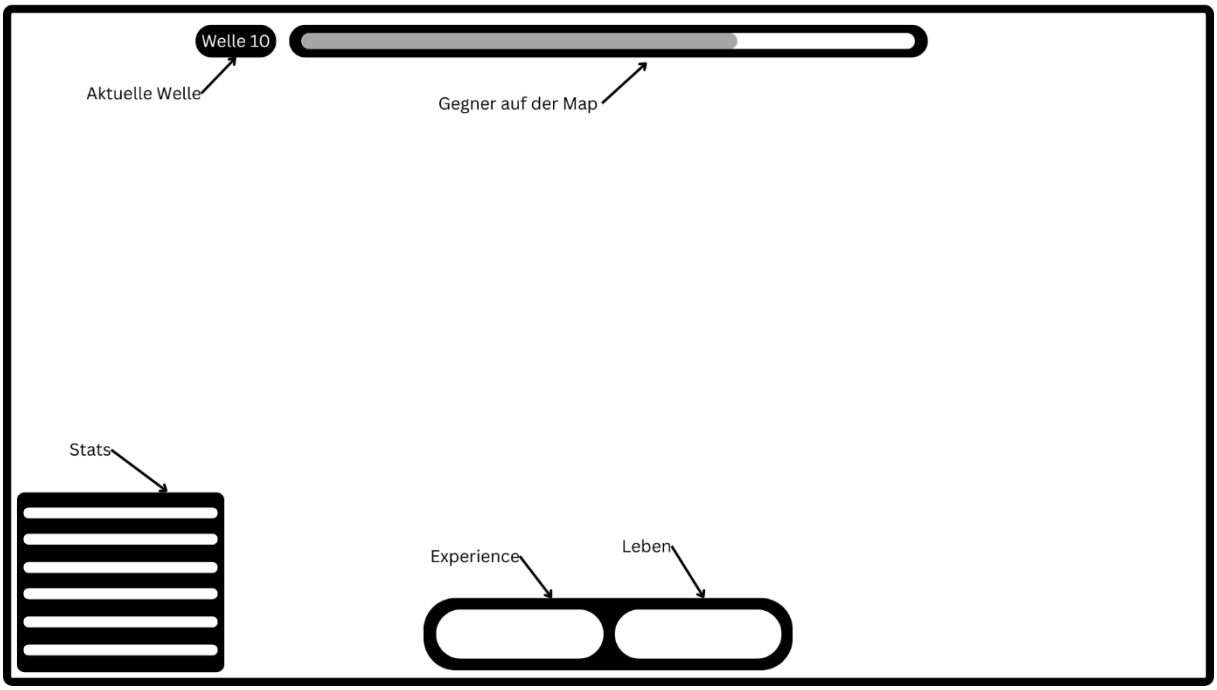
Bewegung

Bewegung	Tasten
Oben / Links / Unten / Rechts	W / A / S / D
Zielen zum Schiessen	Maus
Schiessen von Projektil	Maus Linksklick
Einsammeln von Gegenständen	Mit Character rüberlaufen

4.2 Visuelle Darstellung Spiel

4.2.1 Start Screen

4.2.2 Im Spiel



5. Umsetzung und Unit-Tests

5.1 Einführung Unit Tests

In Unity gibt es verschiedene Arten von Unit-Tests es gibt zum einen Unit Tests im Edit Mode und im Play Mode, die für unterschiedliche Zwecke verwendet werden und unterschiedlich gehandhabt werden müssen. Die Unit Test im Edit Mode sind unabhängig und können jederzeit getestet werden es kommt nicht drauf an, ob das Spiel gerade yläuft oder nicht. Bei den Play Mode Unit Tests hingegen kann die Klasse oder Methode nur korrekt getestet werden, wenn das Spiel am laufen ist, sonst werden womöglich falsche Ergebnisse angezeigt.

5.2 EditMode Unit Tests

In diesem Unit Test zeige ich, ob die IncreaseHighScore Methode von der Klasse HighScoreScript richtig funktioniert und den richtigen Wert angibt. Dafür wird eine neue Klasse angelegt mit einer Methode wo drüber [Test] steht. Danach wird ein neues GameObject erstellt mit dem der Input getestet werden kann. Dem GameObject wird die Klasse HighScore hinzugefügt und danach wird ein Backup vom Highscore angefertigt das Daten nicht verloren gehen beim Test. Der HighScore wird dann zurückgesetzt und ein vorher Bild wird gemacht danach wird HighScore um einen bestimmten Wert erhöht. Am Schluss wird verglichen, ob die beiden Werte übereinstimmen und das Backup wird geladen (im TearDown), um Datenverlust zu vermeiden.

...tRed\RealMap\Assets\Tests\EditMode\PlayableTests.cs 1

```
1  using System;
2  using NUnit.Framework;
3  using UnityEngine;
4
5  public class PlayableTests
6  {
7      private HighScoreScript _HighScoreScript;
8      private int _backupHighScore;
9
10     [Test]
11     public void CheckIncreaseHighScore()
12     {
13         GameObject obj = new GameObject();
14         _HighScoreScript = obj.AddComponent<HighScoreScript>();
15
16         _backupHighScore = Convert.ToInt32
             (_HighScoreScript.GetHighScoreText());
17
18         _HighScoreScript.ResetHighScore();
19
20         int before = Convert.ToInt32(_HighScoreScript.GetHighScoreText
             ());
21         Assert.AreEqual(0, before);
22
23         int increase = 3;
24         _HighScoreScript.IncreaseHighScore(increase);
25         int after = Convert.ToInt32(_HighScoreScript.GetHighScoreText
             ());
26
27         Assert.AreEqual(before + increase, after);
28     }
29
30     [TearDown]
31     public void TearDown()
32     {
33         _HighScoreScript.ChangeHighScore(_backupHighScore);
34     }
35 }
```

5.3 PlayMode Unit Tests

5.3.1 SetUp Method

Hier beginne ich mit einem SetUp für alle Methoden, um die nicht immer wieder neu schreiben zu müssen.

```
...ed\RealMap\Assets\Tests\PlayMode\PlayModeTesting.cs                    1
  1 using NUnit.Framework;
  2 using System.Collections;
  3 using UnityEngine;
  4 using UnityEngine.TestTools;
  5
  6 public class PlayModeTesting
  7 {
  8     private GameObject healthObj;
  9     private HealthScript healthScript;
 10     private UnityEngine.UI.Slider slider;
 11
 12     [SetUp]
 13     public void SetUp()
 14     {
 15         healthObj = new GameObject("Health");
 16         slider = healthObj.AddComponent<UnityEngine.UI.Slider>();
 17         healthScript = healthObj.AddComponent<HealthScript>();
 18         GameObject textObj = new GameObject("ValueText");
 19         textObj.transform.SetParent(healthObj.transform);
 20     }
```

5.4 Test Player Movement

In dem UnityTest wird das Movement getestet, ob es richtig funktioniert. Dafür wird ein Spieler double erstellt, der wie ein Spieler manipuliert werden kann. So wird dann die X-Achse geändert und geschaut ob der RigidBody2D den Input in bewegung umwandelt.

```
 22
 23     [UnityTest]
 24     public IEnumerator PlayerMovementTest()
 25     {
 26         GameObject fakePlayer = new GameObject();
 27         Rigidbody2D rb = fakePlayer.AddComponent<Rigidbody2D>();
 28         PlayerScript playerScript =                                    ➥
              fakePlayer.AddComponent<PlayerScript>();
 29
 30         playerScript.Start();
 31         playerScript.xAxis = 1;
 32         playerScript.yAxis = 0;
 33
 34         playerScript.FixedUpdate();
 35
 36         Assert.AreEqual(playerScript.playerSpeed, rb.linearVelocity.x,  ➥
              0.01f);
 37         Assert.AreEqual(0f, rb.linearVelocity.y, 0.01f);
 38         yield return null;
 39     }
```

5.5 Upgrade Klasse Test

In der Methode wird geschaut, ob Methoden von UpgradeScript richtig funktionieren und die Werte aktualisiert werden.

```
41      [UnityTest]
42      public IEnumerator UpgradeTesting()
43      {
44          GameObject healthObj = new GameObject();
45          HealthScript _HealthScript =
              healthObj.AddComponent<HealthScript>();
46
47          GameObject xpObj = new GameObject();
48          XPScript _XPScript = xpObj.AddComponent<XPScript>();
49
50          GameObject UpgradeObj = new GameObject();

...ed\RealMap\Assets\Tests\PlayMode\PlayModeTesting.cs                    2
51          UpgradeScript _UpgradeScript =                                ➲
              UpgradeObj.AddComponent<UpgradeScript>();
52
53          _XPScript.currUpgrades = 1;
54
55          int newHealth = 200;
56          _HealthScript.SetMaxHealth(200);
57
58          yield return null;
59
60          _UpgradeScript.UpgradeHealth(newHealth);
61          Assert.AreEqual(newHealth, _HealthScript.GetMaxHealth());
62      }
```

5.6 Unterschiedliche Health Funktionen

In den Methoden werden unterschiedliche Funktionen von der Klasse Health sich angeschaut, da diese besonders wichtig für das Spielerleben sind und keinesfalls fehlschlagen dürfen. Ich nutze den IEnumerator bei vielen Methoden, um nach der Eingabe der Inputs einen Frame warten zu können mit «yield return null;» und dann den vergleich starten zu können. So vermeidet man unnötige Fehler und hat eine sichere Testing Methode.

```
64      [UnityTest]
65      public IEnumerator IncreaseHealthTest()
66      {
67          healthScript.SetHealth(20);
68          healthScript.IncreaseHealth(30);
69          yield return null;
70          Assert.AreEqual(50, slider.value);
71      }
72
73      [UnityTest]
74      public IEnumerator DecreaseHealthTest()
75      {
76          healthScript.SetHealth(70);
77          healthScript.DecreaseHealth(30);
78          yield return null;
79          Assert.AreEqual(40, slider.value);
80      }
81
82      [UnityTest]
83      public IEnumerator SetHealthTest()
84      {
85          healthScript.SetHealth(55);
86          yield return null;
87          Assert.AreEqual(55, slider.value);
88      }
89
90      [UnityTest]
91      public IEnumerator SetMaxHealthTest()
92      {
93          healthScript.SetHealth(40);
94          healthScript.SetMaxHealth();
95          yield return null;
96          Assert.AreEqual(100, slider.value);
97      }
98
99      [UnityTest]
100     public IEnumerator SetMaxHealthTestWithParam()
101     {
102         healthScript.SetHealth(60);
```

...ed\RealMap\Assets\Tests\PlayMode\PlayModeTesting.cs 3

```
103         healthScript.SetMaxHealth(50);
104         yield return null;
105         Assert.AreEqual(60, slider.value);
106         Assert.AreEqual(150, slider.maxValue);
107     }
108 
109     [Test]
110     public void GetMaxHealthTest_Return()
111     {
112         var result = healthScript.GetMaxHealth();
113         Assert.AreEqual(100, result);
114     }
115 }
116 
```

6. Individuelle Lernjournale

6.1 Halbtag 1.

Datum und Dauer: 02.04.2025 – 4 Lektionen

6.1.1 Ziele:

- Überlegen ob Alleine oder im Team
- Ideen finden welches Projekt zu mit passt
- Ziele stecken für das Projekt
- Passende Engine finden
- Einleitung in Engine

6.1.2 Was wurde gemacht?

- Ich habe mich dazu entschieden das Projekt allein umzusetzen, weil ich glaube das ich dann effektiver und schneller bin als in einer Gruppenarbeit.
- Dann habe ich noch Unity als Engine bestimmt, weil Unity gut geeignet ist, um 2D-Projekte umzusetzen und weil Unity mit C# arbeitet.
- Ich habe dann noch definiert was für ein Projekt ich umsetzen möchte und habe es spezifiziert, um genau auf das Ziel hinzuarbeiten.
- Die Ziele waren klar, nachdem ich nachgeschaut habe wie dieses Genre an Videospielen heisst. Es handelt sich um ein Roguelike-2D Top-Down Shooter.
- Zum Schluss habe ich Videos angeschaut um mich mit Unity vertraut zu machen.
- Da habe ich Videos von Game Maker's Toolkit angeschaut, da habe ich die Einführung von Unity angeschaut und ich fand dieses Tutorial recht gut, weil er die Benutzeroberfläche und Einstellungen anschaulich und einfach erklärte.

6.1.3 Lessons Learned

- Ich habe gelernt wie die Benutzer Oberfläche von Unity zu bedienen.
- Wie man in der Hierarchy Objekte erstellt
- Wie man in Unity Skripte hinzufügt und zu Objekten anfügt
- Die ersten Methoden von Unity kennengelernt.

6.2 Halbtag 2.

Datum und Dauer: 09.04.2025 – 4 Lektionen

6.2.1 Ziele:

- Prototyp programmieren
- Spieler als Sprite einfügen
- Bewegung des Spielers via W / A/ S / D implementieren
- Spieler Collider anfügen

6.2.2 Was wurde gemacht?

- Das Ziel für heute war es den Charakter, den nachher der Spieler spielt einzufügen. Er sollte sich bewegen lassen vom Spieler in vier Richtungen.
- Dafür habe ich ein Tutorial angeschaut, welches zeigt wie ich den Spieler dieses Verhalten hinzufügen kann per Skript. Das hat dann auch funktioniert

- Als nächstes habe ich noch mit Collidern getestet, ob der Spieler diesen annimmt und bei wänden stehenbleibt. Das hat dann nach mehreren Versuchen funktioniert.
- Für das Unterfangen habe ich das gleiche Tutorial wie letztes Mal von Game Maker's Toolkit geschaut, aber weiter als zuvor da hat er erklärt, wie ich mit Collidern arbeiten konnte und sie im Skript verwenden konnte, immer noch ein sehr gutes Tutorial, einfach und verständlich

6.2.3 Lessons Learned

- Ich habe gelernt, wie ich Input Signale annehmen kann.
- Ich habe gelernt, wie ich Inputs verarbeiten kann von unterschiedlichen Eingabegeräten
- Ich habe gelernt, wie ich den Spieler bewegen kann.
- Ich habe gelernt, wie ich die Bewegung interpolieren kann, um eine flüssigere Bewegung zu schaffen.

6.3 Halbtag 3.

6.3.1 Ziele

- Gegner hinzufügen
- Gegner greift Spieler an
- Health System einfügen
- Gegner design (Sprite)
- Gegner Animation

6.3.2 Was wurde gemacht?

- Ich habe heute angefangen den ersten Gegner in meinem Spiel hinzuzufügen
- Dafür habe ich einen Gegner Sprite heruntergeladen
- Dann habe ich ihn erstellt mit den Spriten
- Als nächstes habe ich den Gegner animiert in verschiedenen Situationen wie Idle, gehen, angreifen und diese dann wiederum in verschiedenen Richtungen (oben, unten, links, rechts).
- Dann habe ich im Skript programmiert, das der Gegner bei einer bestimmten Distanz angreifen soll.
- Am Schluss habe ich dann noch das Health System vom Gegner eingefügt das er bei einer bestimmten Hitanzahl sterben soll.
- Um Gegner hinzuzufügen mit Animationen etc. habe ich die Tutorial Reihe von Pandermonium angeschaut, wo er dies erklärt. Ich fand diese Tutorials recht gut. Aber es dauert recht lange bis man das Wissen hat, weil die Videos lang gehen. Dennoch empfehlenswert, weil er langsam und schritt nach schritt erklärt.

6.3.3 Lessons Learned

- Ich habe gelernt, wie ich ein Health System schaffen kann in Unity.
- Ich habe gelernt, wie der Gegner den Spieler auswählen kann und auf ihn loslaufen kann.
- Ich habe gelernt, wie der Gegner dem Spieler bei unterschiedlichen Bedingungen schaden kann.

6.4 Halbtag 4.

6.4.1 Ziele

- Assets für Tiles herunterladen und ins Projekt

- Benutzung von Tilemaps lernen
- Design auswählen
- Rule Tiles erstellen
- Map erstellen

6.4.2 Was wurde gemacht?

- Erstmal wurde festgelegt welche Tilemap in meinem Spiel passte.
- Danach wurde dieses Tilemap heruntergeladen und in Unity eingefügt
- Als Nächstes musste, gelernt werden, wie man mit Tilemaps arbeitet
- Das nächste Ziel war es zu experimentieren, wie es funktioniert und wie man mit Tiles allgemein arbeitet.
- Dann habe ich überlegt, wie ich die Map Erstellung einfacher machen kann, dass ich nicht alles selber einzeln einfügen muss.
- Die Idee war es dann Regeln einzufügen für jedes Tile wann es wo platziert werden muss, dass es Endeffekt eine logische Map herauskommt.
- Dies war notwendig das die Map Erstellung durch Random Prozedural Generation gemacht werden konnte, was der nächste Schritt war in den nächsten Wochen.
- Dann habe ich mich informiert durch YouTube Tutorials wie man Rules erstellt und habe dies dann auch umgesetzt.
- Die restliche Zeit war ich dann damit beschäftigt die Map mit meiner Vorlage zu erstellen, um ein Ideal Bild zu haben, auf die ich bei der Random generierten Map hinarbeiten kann.
- Am Schluss war ich noch dran die Hitboxen hinzuzufügen für einzelne Layer
- Für heute habe ich die Tile Videos von NotSlot, Velvary und MoreBBlakeyyy geschaut. Alle fand ich einfach zu verstehen und nachzumachen, aber Velvary hat mir am besten gefallen, weil sie es am besten erklärt hat. Und der Schnittstyle vom Video angenehmer war nachzumachen.

6.4.3 Lessons Learned

- Ich habe gelernt mit Tiles und Tilemaps zu arbeiten
- Ich habe gelernt, wie ich grids erstelle mit dem Tile Palette Editor
- Ich habe gelernt Regeln mit Ruletiles zu erstellen, um den Prozess einfacher zu machen
- Ich habe die Logik hinter Ruletile bildung verstanden
- Ich habe gelernt, wie ich mit Tiles Welten erstellen kann.

6.5 Halbtag 5.

6.5.1 Ziele

- Dokumentation weiter schreiben
- Unterkapitel in der Dokumentation hinzufügen
- Tilemap debuggen

6.5.2 Was wurde gemacht?

- Ich habe die Dokumentation mit einigen Unterkapiteln ergänzt für die nachträglichen Ergänzungen für die Schlussabgabe.
- Dann habe ich bei der Planung viele Punkte hinzugefügt und verbessert.
- Ich habe überlegt was ich noch Umsetzung muss und die Zeiteinteilung dementsprechend angepasst.

- Danach gab es noch einige Probleme mit den Tilemaps die ich noch beheben musste. Wie die Hitboxen ich musste 2 Layer voneinander trennen um die Hitboxen nur auf die Wände geltend zu machen, denn am Anfang wäre der Boden auch davon betroffen gewesen, was nicht optimal wäre.
- Dann musste ich noch überlegen, wie ich die Gegner intelligenter programmieren könnte, dass sie um Objekte gehen können, um den Spieler angreifen zu können. Deshalb habe ich mich für den A* Algorithmus entschieden. Er ist zwar nicht einfach in der Programmierung, aber sehr effizient und schnell.
- Ich habe das Video von Shack Man angeschaut, um zu schauen wie ich Informationen in Tiles speichern kann und verwenden kann für Interaktionen, dies wird für später noch wichtig werden.

6.5.3 Lessons Learned

- Ich habe gelernt, wie ich vorgehen kann, um Probleme in Unity fixen zu können.
- Ich habe gelernt, welche Algorithmen existieren für Pathfinding Algorithmen von Gegnern.
- Ich habe grobe Kenntnisse über den A* Algorithmus gesammelt.

6.6 Halbtag 6.

6.6.1 Ziele:

- Deathscreen hinzufügen
- Startscreen hinzufügen
- UI-Elemente fürs Spiel erstellen
- Logik des Spiels visualisieren (Health, XP usw.)

6.6.2 Was wurde gemacht?

- Als erstes habe ich heute angefangen die wichtigen UI-Elemente hinzuzufügen, die ich in Canva skizziert habe.
- Davor habe ich UI-Assets ausgesucht und heruntergeladen das ich passende Elemente habe die ich zusammenstecken konnte.
- Als ich mit der Oberfläche des Hauptscreens fertig war, habe ich angefangen die Slider hinzuzufügen, um Health und XP anzuzeigen.
- Danach habe ich mit der Implementation der Logik angefangen um die Health Bar anzuzeigen. Dies konnte ich recht schnell realisieren, weil ich davor schon die passenden get Methoden gesetzt habe im Skript.
- Als nächstes war die Implementierung von der XP-Aufnahme und Verwertung an der Reihe, dafür habe ich ein neues Skript angefertigt und ein Grundgerüst für weitere Funktionen angefertigt.
- Dann habe ich noch den Startscreen designt mit einem Highscore und zwei Buttons, die ich später nutzbar machen werde, diesen habe ich in einer neuen Szene erstellt.
- Zu guter Letzt habe ich den Deathscreen eingebaut auch in einer neuen Szene, wo ich die Funktionalität nächstes Mal einbauen werde.

6.6.3 Lessons Learned

- Ich habe heute gelernt wie ich UI gestalten kann.
- Ich habe gelernt wie UI zu implementieren ist und wie ich es verwenden kann um Informationen attraktiv anzuzeigen.
- Dazu habe ich noch gelernt wie ich neue Szenen erstellen kann und verwenden kann.

- Ich habe dann noch gelernt wie ich bestehende Methoden verwenden kann um zukünftige Features effektiv einzubauen und Zeit zu sparen.

6.7 Halbtag 7.

6.7.1 Ziele:

- Enemy AI vom Subprojekt ins Hauptprojekt importieren
- Testing und Debuggen der Funktionen
- Anpassungen der Funktionalität des AI Systems
- Hinzufügen und Implementation von Object Data (Scriptable Object)
- Funktionalität von Startscreen und Deathscreen implementieren

6.7.2 Was wurde gemacht?

- Ich habe damit angefangen mein erstelltes KI System von einem Subprojekt was ich Zuhause fertig gestellt habe ins Hauptprojekt einzubinden.
- Hier einige YouTuber von den ich die Entwicklung des Algorithmus gelehrt haben: Code Monkey, Challacade
- Als ich alle Gefahren gescannt hatte die passieren konnten und diese beseitigt hatte habe ich begonnen die Skripte vom Subprojekt ins Hauptprojekt reinzukopieren.
- Dann gab es noch einige Probleme im Debug.Log, aber danach dem fixen dieser Probleme hat der Anfang schon mal funktioniert.
- Ich habe dann noch einige Settings geändert so das es im grösserem Projekt auch noch wie gewünscht funktioniert.
- Als nächstes To-do habe ich mir die einfügen von Scriptable Objects vorgenommen, was ich von Code Monkey gelernt habe. Es ist sehr nützlich um generelle Attribute von Objekten festzuhalten. Zumbeispiel bei Enemies diese haben mehrere Eigenschaften wie Health, XP, Damage usw. da kann es nützlich sein sie schnell anpassen zu können wie auch für weitere Gegner wenn man neue erstellt.
- Schlussendlich habe ich die Funktionalitäten von Start- und Deathscreen hinzugefügt. Wie die Button events dafür habe ich mehrere Onclick Events von einem Script aus aufgerufen wenn der Button geklickt wird.
- Dazu habe ich noch hinzugefügt das wenn man 0 HP hat, zum Deathscreen Szene rübergebracht wird und neustarten oder zum Hauptscreen zurückehren kann.

6.7.3 Lessons Learned

- Ich habe gelernt, wie ich effektiv Debuggen kann und die Debug Funktionen zu meinem Vorteil nutzen kann.
- Ich habe gelernt, wie ich Scriptable Object nutzen kann und sie implementieren kann in das Spiel.
- Ich habe gelernt, wie ich Button Events verarbeiten kann.
- Ich habe gelernt, wie ich OnClick Events für Buttons hinzufügen kann.

6.8 Halbtag 8.

6.8.1 Ziele:

- Einbauen der Procedural Map Generation
- Testen der Funktionen der Map Generation
- Scripts der Spawner anpassen
- Spawn Point zu der Map hinzufügen

- Random Obstacle generation programmieren

6.8.2 Was wurde gemacht?

- Ich habe damit begonnen meinen Algorithmus, den ich bereits entwickelt habe und funktioniert in mein richtiges Projekt einzufügen.
- Dieses Unterfangen erwies sich schwerer als erwartet, aber nach einigen Tests ging das Generieren auch richtig.
- Danach habe ich alle Scripts debuggt die mit der Map etwas zu tun haben, wie der EnemySpawner oder die Enemy AI, ob sie unter der neuen Map richtig funktionieren.
- Deshalb musste ich dann noch recht viel an den anderen Scripts anpassen, um der neuen Map Generierung gerecht zu werden.
- Ich musste anpassen das die Map bei jeder neuen Welle neu gescannt wird und daraus ein grid erstellt wird wo einerseits die Gegner sich orientieren müssen und andererseits die Gegner auch spawnen werden.
- Als ich das Debuggen der Probleme vorbei war und alles in der Richtung problemlos funktionierte habe ich begonnen die Map Generierung anzupassen an meine Vorstellungen.
- Dann habe ich in die Map integriert das man in einem Spawn Point spawnt und sich dieser nicht auflöst, wenn die Map dann neu generiert wird.
- Schlussendlich wollte ich die Map noch interessanter Gestalten und habe einige GameObjects hinzugefügt die Random generiert werden können auf den Ground Tiles, das es interessanter fürs Auge ist.

6.8.3 Lessons Learned

- Ich habe gelernt, wie ich Algorithmen von Subprojekten ins Hauptprojekt einfügen kann.
- Ich weiss wie ich Klassen effektiv miteinander verbinde und deren Methoden verwenden kann.
- Ich habe gelernt, wie ich besser debuggen kann und erkenne jetzt, wo Fehlerquellen sein könnten.
- Ich habe gelernt, wie ich Maps interessanter gestalten kann mit Prozentualer Generierung.
- Ich habe gelernt wie ich Prefabs als Tiles nutzen kann, um zb. Spawnpoints zu setzen.

6.9 Halbtag 9.

6.9.1 Ziele:

- Highscore System einfügen
- Projekt auf Github veröffentlichen
- Informieren über Unit Tests und einbauen
- Datenspeicherung auf Festplatte (Langzeitspeicher)
- Fehler Debugen

6.9.2 Was wurde gemacht?

- Das erste Ziel für heute war es das Highscore System einzufügen. Dafür habe ich das Video von Brackeys mit angeschaut siehe link im Quellenverzeichniss.
- Ich habe den Code nach meinen ansprüchen dann angepasst und passende Methoden für die weiterverarbeitung eingefügt.

- Mit dem Integrieren des Highscore Systems habe ich gleichzeitig angeschaut wie man datenschnipsel wie in einem Highscore zumbeispiel abspeichern kann auf dem lokalem Computer.
- Danach habe ich die intergrierte Struktur mit den bestehenden Highscore Anzeigen verbunden und getestet.
- Als nächstes wollte ich mein Projekt veröffentlichen auf Github, aber dies hat nicht geklappt, weil ein file zu gross war. Deswegen habe ich mich dazu entschieden das Projekt auf einer anderen Cloud hochzuladen.
- Um sicherzustellen das der Highscore auch immer funktioniert, habe ich mich über Unit tests informiert mit zwei Videos von Infallible Code und Kodo Linija siehe Quellenverzeichniss.
- Dann habe ich die Unit Tests im EditMode für das Highscore System eingefügt und getestet.
- Es gab noch einige Probleme damit und paar andere Bugs wie die stärke der Gegner oder des Wave Counters die ich gefixt habe.

6.9.3 Lessons Learned

- Ich habe gelernt wie ich Daten für längere Zeit speichen kann und sie wieder abrufen kann .
- Ich habe gelernt für welche Zwecke Unit Tests hinfreich sind in der Spiele Entwicklung.
- Dann habe ich gelernt wie Unit Tests gecodet werden in Unity und wann man welche der beiden Methoden nutzen sollte (Play- und EditMode).
- Ich habe gelernt wie ich Code schreibe den ich jederzeit wiederverwenden kann, wie Methoden die von anderen Klassen angewendet werden können.
- Ich habe gelernt wie ich Code nützlich auslagern kann um die übersicht über das projekt zu behalten.
- Ich habe gelernt für was man Assemply benutzt um unit Tests zu schreiben.
- Ich habe gelernt wie ich Unit Tests effektiv anwenden kann um frühzeitig probleme zu sehen und darauf reagieren zu können.

7. Quellenverzeichniss

Quelle	Link	Bemerkung
YouTube: First Tutorial Series	Link	
YouTube: Infos in Tile	Link	
YouTube: Autotilling	Link	
YouTube: Autotilling	Link	
YouTube: Tilemaps	Link	
YouTube: Tilemap Guide	Link	
YouTube: Beginner Tutorial	Link	
YouTube: AStar Algorithmus	Link	
YouTube: AStar Algorithmus	Link	
YouTube: AStar Algorithmus	Link	
YouTube: Prozedural Generation Map	Link to Series	
Youtube: Highscore Unity	Link	
YouTube: Unit Tests	Link	
YouTube: Unit Tests	Link	
YouTube: Proce. Gener.	Link	
YouTube: Smart Enemy	Link	
YouTube: Scriptable Obj.	Link	
Google: Unity Dokumentation	Link	
Google: Coroutine	Link	
Google: Unity Learn	Link	

8. Dokumentenversionen

Versionen	Änderung	Datum
V1.0	Hinzufügen von Journal erster Halbtag	02.04.2025
V.1.1	Hinzufügen von Journal zweiter Halbtag	09.04.2025
V1.2	Hinzufügen von Journal dritter Halbtag	16.04.2025
V1.3	Hinzufügen von Journal vierter Halbtag	07.05.2025
V1.4	Planung hinzufügen, Unterkapitel für später ergänzen, Benutzerhandbuch, Grobziele	14.05.2025
V1.5	Hinzufügen von Journaleintrag fünfter Halbtag, Titelseite bearbeiten, Visuelle Darstellung Spiel hinzugefügt	17.05.2025
V1.6	Hinzufügen Journal 6. Halbtag	24.05.2025
V1.7	Hinzufügen Journal siebter Halbtag	30.05.2025
V1.8	Hinzufügen Journal 8. Halbtag	07.06.2025
V1.9	Hinzufügen vom 9. Journal	11.06.2025
V.2.0	Summary schreiben, Titelbild ändern	12.06.2025
V2.1	Unit-Test Dokumentation hinzugefügt	13.06.2025