TDD – Technical Design Document

„Post-Apocalypse"

Semesterabschlussprojekt 4

GDS23 – Macromedia Leipzig, von Linnea Hoppe

Beihilfe von Mika Neukirch

Letzter Eintrag: 19.06.2025

Letzte Bearbeitung: 26.07.2025

INHALT

1 VORWORT

1.1 WAS IST DAS TDD UND FÜR WEN?

- Das TDD ist ein Dokument was die technischen Sachen in einem Spiel wiedergibt
  - Also was in Engine alles passiert (Scripte, Verlinkungen, Verknüpfungen)
- Ist für jede Person im Team einsehbar
- Muss für alle Departments verständlich sein

1.2 TECHNISCHE ECKDATEN

Unity Version: 2022.3.12f1

Code: Visual Scripting & C#

Importierte Packages: Cinemachine, TextMeshPro, 2D Sprite

Erforderliche Windows:

Game View: zeigt dir wie es im Game aussieht

Animation: Damit erstellt man die Keys für Animationen

Project: Dort sind alles Folders drinnen

Console: zeigt dir Fehler während des Playtestings an

Scene: Ist die View wo man drin arbeitet, Objekte einfügt und platziert

Animator: Zum Verknüpfen bei mehreren Animationen (z.B. Es gibt eine Idle Animation und eine Lauf Animation auf einem Objekt, dann wird zuerst die Idle als Start gelegt und eine Transition gemacht zum Laufen)

Inspector: zeigt dir die Parameter zu jedem erdenklichen Objekt, Funktion oder Material an

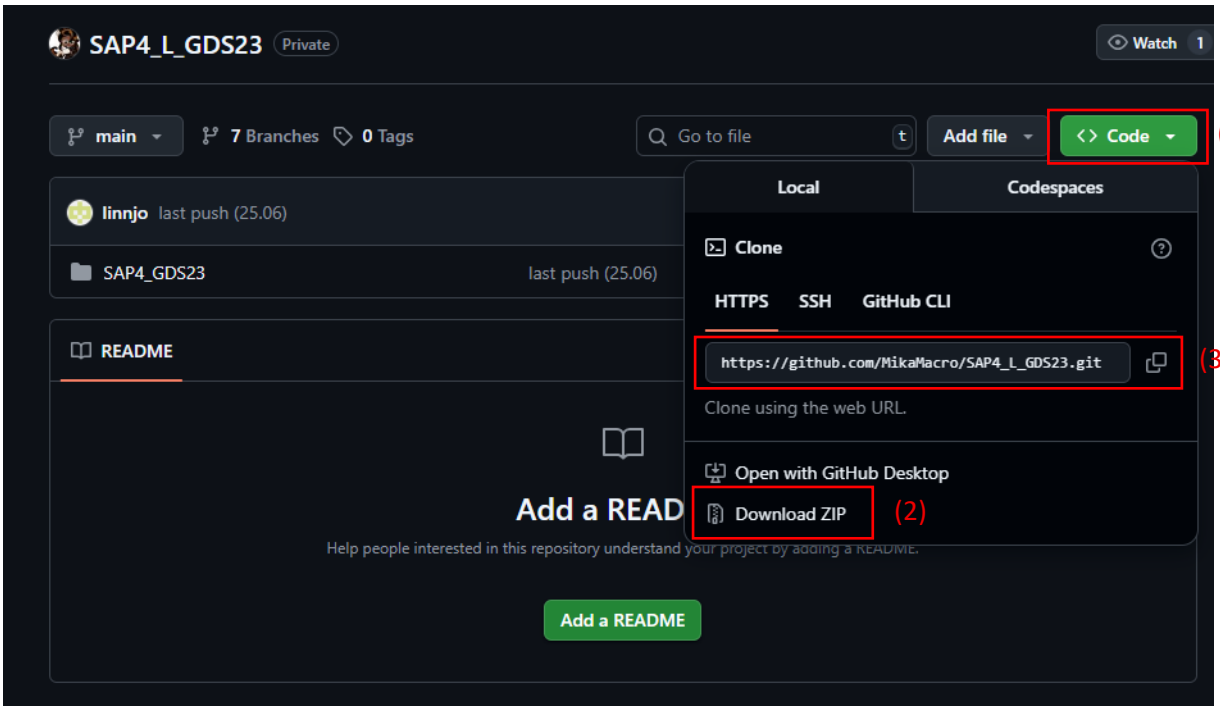
Erwähnenswert: Lightning, Sprite Editor, Project Settings, Build Settings, Package Manager

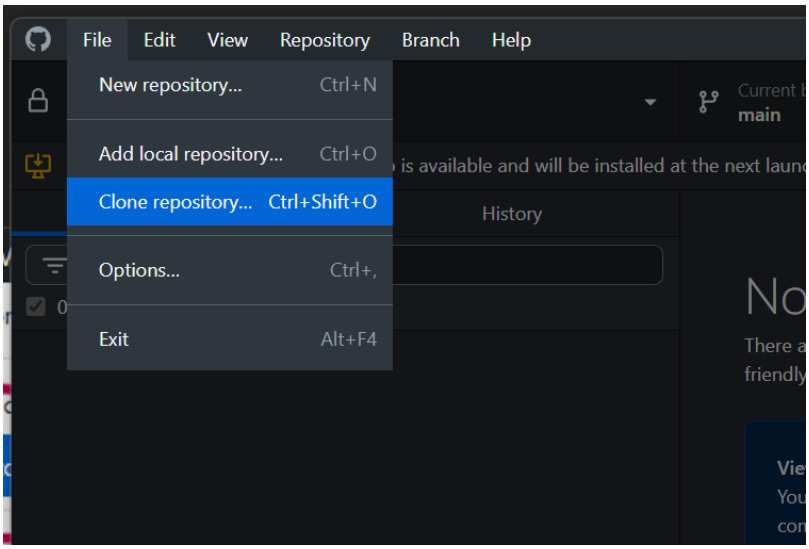
(Wie findet man Windows? Unter Window → General)

1.3 Projekt Herunterladen und GitHub einrichten

Wenn man auf GitHub über die E-Mail eingeladen wurde + die E-Mail akzeptiert kommt man auf die Übersicht des Projekts

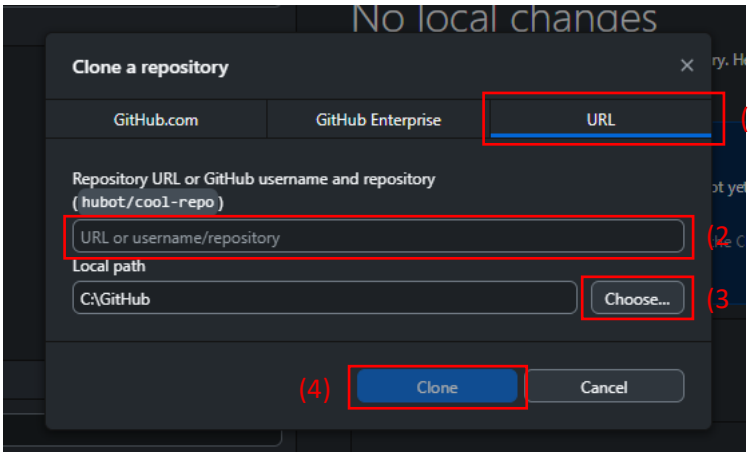
1. Klicke auf den Code Buttons der dir ein kleines Fenster öffnet
2. Lade dir die Zip runter + entpacke sie (für Unity hub)
3. Kopiere die URL (Für GitHub)





Dann gehst du auf GitHub Desktop

1. Files > Clone repository…

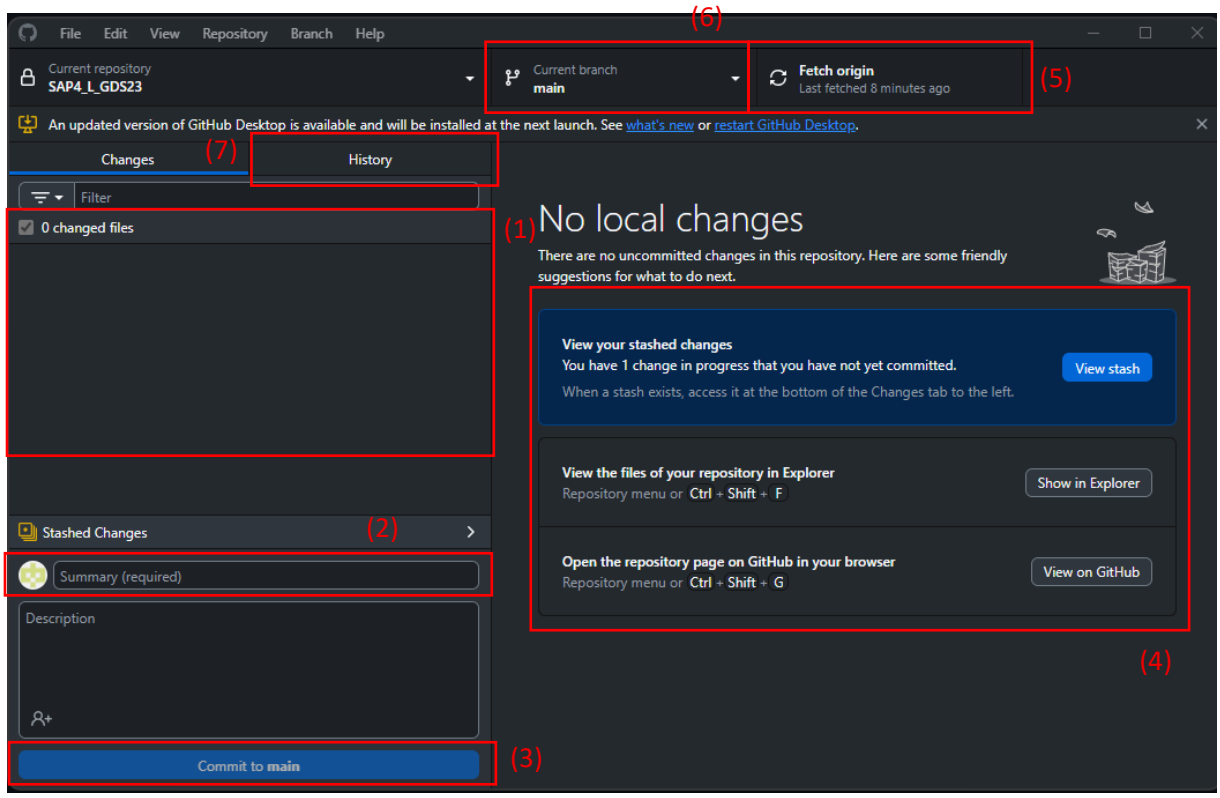


Danach öffnet sich dieses Fenster

1. Du wählst zwischen den 3 Reitern URL aus
2. Du kopierst den URL link von der GitHub Seite da rein
3. Du wählst auf deinen Desktop dein Speicher Ort
4. Du klonst diese Repo

Falls dies fertig ist kannst du über den Unity Hub das Projekt adden, falls in der Repo kein Projekt drin ist, nimm die ZIP und ziehe sie in den Repo Folder und adde dieses.

GitHub Arbeit:



Für die Zusammenarbeit ist es wichtig zu verstehen was man tun muss + was welche Funktion dir zeigt:

1. Wenn du in Unity etwas eingefügt, geändert oder gelöscht hast wird dies hier als Change angezeigt
2. Um dein Change zu speichern, musst du in Summary eine kurze Beschreiben erstellen (z.B. „Scenes eingefügt“, „Art Assets in Level 1 placed“)
    a. Wenn du einen oder alle Change nicht speichern willst, machst du rechts Klick auf den Change/auf den Reiter wo die Anzahl steht und wählst „Discard changes“/bzw. „Discard all changes“ aus
3. Nach dem Beschreiben klickst du auf den blauen Button und alle Änderungen an dem Projekt werden gespeichert
4. Nach dem commiten wird ein Button „Push“ Button erscheinen, beim Drücken werden die gespeicherten Changes auch in der öffentlichen Repo hochgeladen
5. Im Projekt ist es wichtig zu verstehen das viele Leute in der gleichen Engine arbeiten, deswegen ist es wichtig vor dem Bearbeiten zu fetchen und nach einen push zu fetchen damit frühzeitige Fehler erkannt werden
6. Zur Sicherheit kann ein neben Branch erstellt werden, dieser kann später vom Coder in die „main“ (der Haupt-Branch für den Coder) gemerged werden
7. In der History sieht man, was wer wann gemacht hat, dort kann man auch zu einem älteren Stand zurückgehen

2 ERSTES SETUP

2.1 TAGS

- → "Player": besitzt "Player2D" in beiden Leveln
- → "Box": besitzt sowohl "Box1" aus Level 1 als auch "Box2" aus Level 2
- → "Enemy": besitzen die Prefabs "Enemy", "Enemy2" und "Boss", die unter dem Resource Folder angelegt sind
- → "Goal1": besitzt das Empty GameObject "Goal1" aus Level 1
- → "Goal2": besitzt das empty GameObject "Goal2" aus Level 2
- → "Exit1": besitzt sowohl das empty GameObject "Exit1" als auch das Child "HardCollider" aus Level 1
- → "Exit2": besitzt sowohl das empty GameObject "Exit2" als auch das Child "HardCollider" aus Level 2
- → "Start": besitzt (momentan nur) das empty GameObject "StartLevel1" aus der Cutscene TutorialLvl1 – Narrativ
- → "Note": besitzt das Prefab "Note" oder generell alle Notiz-Objekte in beiden Leveln
- → "NeedsXRay": besitzen alle 3D Modelle, die auf irgeinem Weise den Spieler verdecken könnten und an dieser Stelle eine "Röntgen-Ansicht" gewünscht ist
- → "BossTrigger": besitzt der eine "Boss Trigger" in Level 2
- → "EnemyTrigger": besitzen alle Enemy Trigger in beiden Leveln
- → "Boarder": besitzt das Prefab "Boarder"

2.2 LAYERS

- → „Player": besitzt die Spielfigur und alle seine Child-Objekte in beiden Leveln
- → „Box": besitzt sowohl "Box1" aus Level 1 als auch "Box2" aus Level 2 + das jeweilge Child-Objekt "Collider"
- → „UI": besitzen alle UI Elemente, mit dem Canvas-Objekt eingeschlossen (wird aber automatisch gesetzt)
- → „Enemys": besitzen die Prefabs "Enemy", "Enemy2" und "Boss", die unter dem Resource Folder angelegt sind

2.3 SCRIPT MANAGEMENT IN SCENES

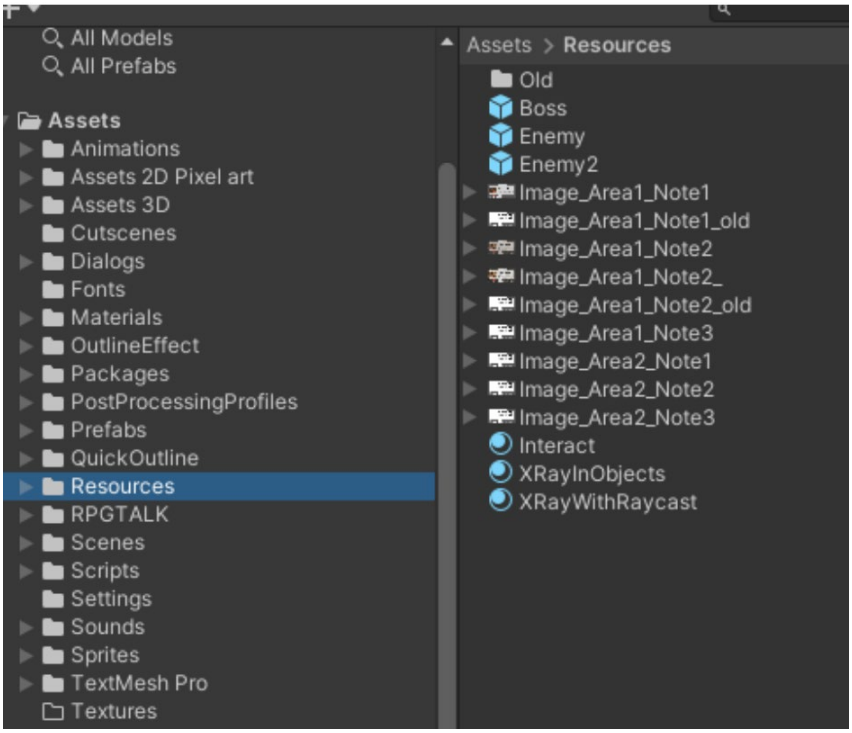
(momentan wird jede funktion per script zugeteilt)

3 CODING UND ENGINE LEXIKA

3.1 ENGINE

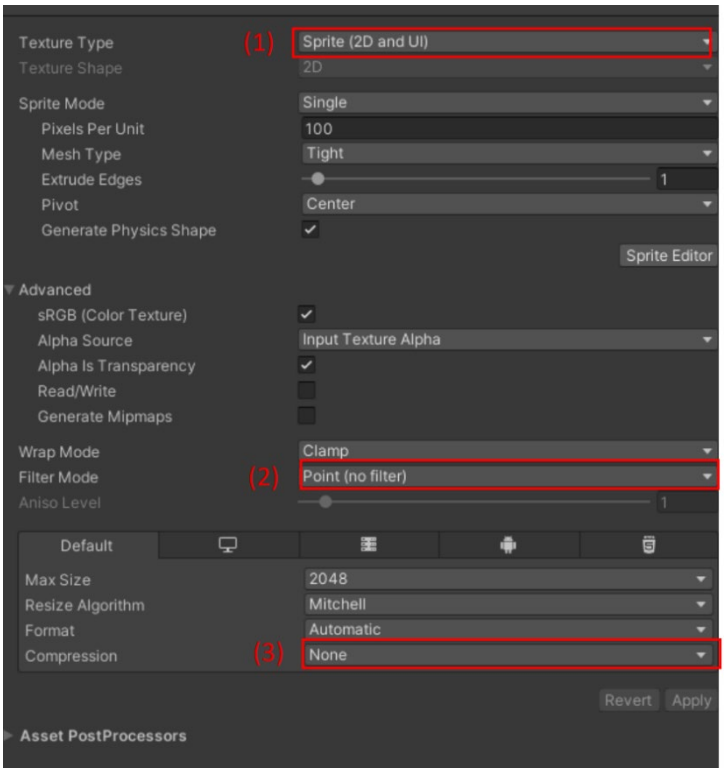
3.1.1 „Resources Folder“

- Dieser Folder ist unter dem üblichen „Assets“ Folder gelistet. Inhalte daraus können über einfachen Weg in Scripten abgerufen werden. Dieser Folder ist insbesondere wichtig für das Laden und Spawnen von Elementen wie Prefabs, Materialien oder Sprites, die so nicht in den Szenen vorhanden sind.



3.1.2 Einrichten von Sprites (Art relevent!)

1. „Texture Type“ muss auf „Sprite (2D and UI) umgestellt werden
2. „Filter Mode“ muss auf „Point (no filter)“ gesetzt werden
3. Die „Compression“ muss auf „None“ stehen
4. Als letztes nicht vergessen, unten auf „Apply“ zu klicken, um die Einstellungen zu bestätigen



3.1.3 Post Processing/Global Volume

→ Post-Processing = visuelle Effekte zur Verschönerung des Spiels.

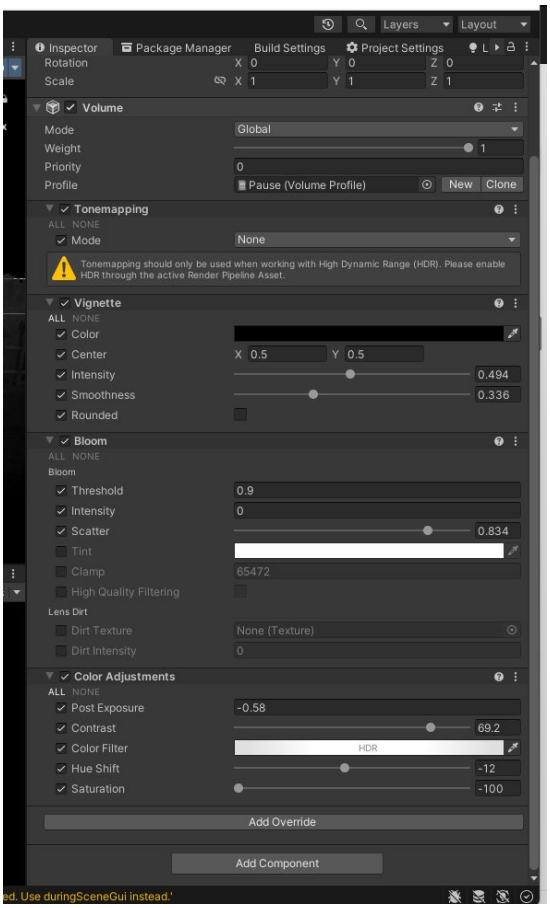
→ Global Volume = ein Ort, wo du diese Effekte für die ganze Szene einstellst

3.1.3.1 Post Processing

- Post-Processing sind grafische Effekte, die nach dem eigentlichen Rendern des Spiels angewendet werden, also "nachbearbeitet" werden, wie bei einem Filter über einem Foto.
- Diese Effekte verbessern die Bildqualität und das visuelle Feeling in Spielen.
- Typische Effekte sind: Bloom, Color Grading, Vignette, Motion Blur

3.1.3.2 Global Volume

- Die Global Volume ist ein Objekt in Unity, das Post-Processing-Effekte auf die ganze Szene anwendet, also global.
- Darauf kann man Effekte aktivieren und zB. ihre Intensität einstellen.



3.1.4 Is Trigger/Trigger Collider:

- Eine Checkbox auf Collider Komponenten, die mit einem Häkchen aktiviert wird
- Ist ein Collider eines Objektes „Is Trigger“, kann dieser nun verwendet werden, um Überlappungen oder Berührungen mit anderen Collidern zu erkennen
- Trigger Collider verlieren aber damit die Fähigkeit der physikalischen Kollision

3.2 SCRIPT

3.2.1 Raycast:

- Ist eine unsichtbare Linie (die man sichtbar machen kann), die zur erkennen von getroffenen Objekten/Collider benutzt wird
- Wir meistens bei Waffen genutzt und mit ifs genutzt, um eine funktion aufzurufen

3.2.2 String, float, int, bool:

➔ Variablen, die den Typen eines Elementes im Script definieren

- string = Text
- int = ganze Zahlen
- float = Kommazahlen, werden mit einem f gekennzeichnet
- bool = ist ein Wahr oder Falsch Indikator

3.2.3 Scene/Panel Load:

- Scenes kann man im Script laden lassen, über den Namespace „UnityEngine.SceneManagement“ können Szenen aufgerufen, geladen, entladen und gewechselt werden
- Beispiel:
    - SceneManager.LoadScene(“Level1”); //Läd scene 1

- Panels kann man im Script Aktivieren oder Deaktivieren lassen
- Beispiel:
    - PausePanel.SetActive(true); //Aktiviert
    - PausePanel.SetActive(false); //Deaktiviert

3.2.4 Time.deltaTime:

- Ist eine Zeitspanne, die in Sekunden berechnet wird, diese zählt wie viel Zeit vom Letzen frame vergangen ist
- wird meistens in Timern verwendet die zurück zählen, oder in Ranglisten um deine Bestzeit zu zeigen

3.2.5 Voids/Start/Update/Awake:

- Voids sind Methoden in denen Beschreibungen ausgeführt werden
- Start ist eine Methode, die einmal zu Beginn aufgerufen wird
- Update ist eine Methode, die zu jedem frame aufgerufen wird
- Awake ist eine Methode, die zu jedem Laden von instancen aufgerufen wird

4 SCRIPTS

4.1 GAME MECHANIKEN UND CONTROLLER

4.1.1 PlayerMovement.cs

→ Dieses Script ist für die Bewegung des Spielers zuständig und dafür, dass das richtige Pixel-Sprite der Figur in der passenden Laufrichtung gezeigt wird.

4.1.1.1 Funktionen

- Das Bewegen der Spielfigur über WASD und Pfeiltasten
- Das Auswechseln der Animation, je nachdem in welche Richtung gelaufen wird
- Das Einspielen der Idle Animation, wenn der Spieler nicht von einem Spielerinput bewegt wird
- Das Sprinten (schnelles Laufen), wenn zusätzlich die Shift-Taste gedrückt wird, der Spieler in Level 2 ist und das Paket nicht in der Hand hält
- Das permanente Prüfen und Abgleichen der Position vom Spieler (im Script als „WayPoint“ bezeichnet), damit die Mutanten später wissen, welchen Punkt sie verfolgen müssen
- Das Einblenden des Game Over Screens, wenn der Spieler von einem Mutanten berührt wird
- Das Spawnen des Bosses in Level 2, wenn der Spieler in einen bestimmten Radius von der Kirche läuft

- Das Einblenden des Hinweises, das der Spieler zuerst alle aktiven Mutanten in der Szene besiegen muss, wenn er davor probieren sollte das Paket abzugeben
- Das Aktivieren des Fenster-Dialogs, sobald das Paket beim Zielort abgegeben wurde
- Das Einblenden des Hinweises, das zuerst das Paket abgegeben werden sollte, wenn der Spieler den Ausgang benutzen möchte
- Die interne Speicherung, wenn das Level erfolgreich beendet wurde sowie das darauffolgende Einblenden von der Delivery-Time und der Anzahl der eingesammelten Files (Werte werden aus dem Screenlogic.cs entnommen)

4.1.1.2 Properties

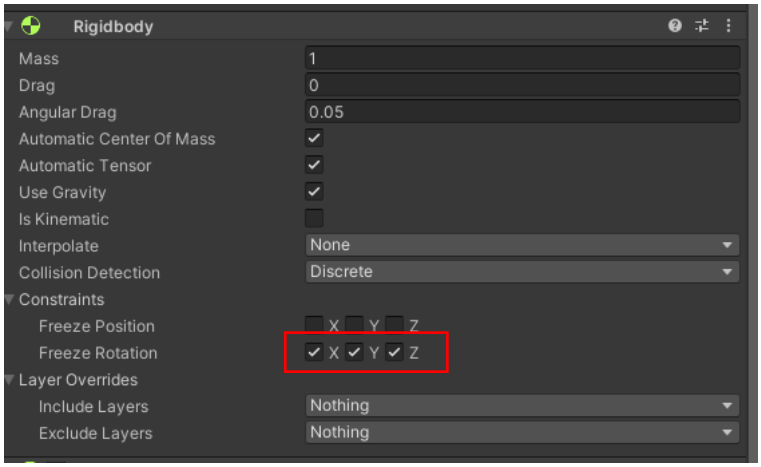
- Default Speed: Die normale Bewegungsgeschwindigkeit des Spielers (default setting: 5)
- Sprint Speed Add: Die Geschwindigkeit, die zu der normalen Laufgeschwindigkeit dazu addiert wird, wenn man Shift drückt (default setting: 5)
- Directions: Eine Sammlung von allen Richtungs-Sprites der Spielerfigur (ist weggefallen!! Statt „Standbildern“ gibt es jetzt natürlich Animationen, die mithilfe von den Verknüpfungen im Animator aufgerufen werden können)

4.1.1.3 Setup

- Liegt auf: Player2D
- Benötigt: Sprite Renderer, Rigidbody, Collider

Hier bitte beachten!

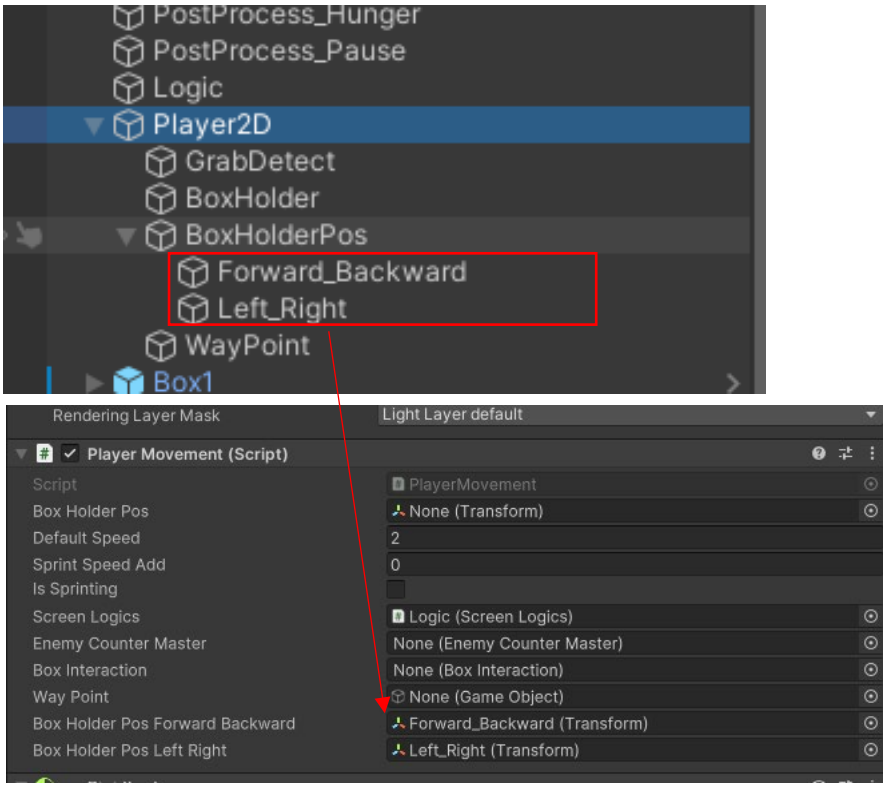
➢ Alle Achsen sollen bei der Rotation des Rigidbodies gefreezed sein



- Das GameObject „Player2D“ hat ein Child Empty Objekt namens „WayPoint“
- Das GameObject „Player2D“ hat ein Child Empty Objekt namens „GrabDetect“
- Das GameObject „Player2D“ hat ein Child Empty Objekt namens „BoxHolder“
- Das GameObject „Player2D“ hat ein Child Empty Objekt namens „Forward_Backward“ sowie „Left_Right“

Hier bitte beachten!

➢ Diese sollten bitte händisch unter „Box Holder Pos Forward Backward“ und „Box Holder Pos Left Right“ hinterlegt sein! (einfach mit Drag’n’Drop)



- Das GameObject „EnemySpawner“ mit dem Script „EnemySpawner“ muss unter „Enemy Spawner“ referenziert sein (-> ausgewechselt durch das Script „EnemyCounterMaster“!! + keine händische Referenzierung erforderlich)

4.1.2 Enemy.cs

→ Dieses Script ist für die Bewegung der Mutanten und ihre Lebensdauer zuständig.

4.1.2.1 Funktionen

- Das Aufsuchen von der Position des Spielers (über „WayPoint“, siehe PlayerMovement.cs) direkt zum Start, wenn sie spawnen
- Das Setzen der „Lebensdauer“ für Mutanten, die mehr als einen Schaden aushalten, auf den Wert, wie viel Schaden sie aushalten
- Das Ändern von Sound und Cam Effects, wenn der Boss-Mutant gespawnt ist
- Das Bewegen in die Richtung des Spielers, indem seine Position immer 1x pro Frame abgefragt und berechnet wird
- Das Anzeigen der richtigen Bewegungs-Animation, indem der Wert von WayPoint abgefragt wird und in welchem Winkel der Mutant zu dem Spieler steht
- Das kurze rot aufleuchten eines Mutanten, wenn er von dem Paket getroffen wurde
- Das Runterrechnen der „Lebensdauer“ um jeweils 1, wenn ein Mutant von dem geworfenen Paket getroffen wurde
- Das Despawnen eines Mutanten und den Mutanten-Counter im UI um 1 runterzählen, wenn er von der geworfenen Box des Spielers getroffen wurde
- Das Deaktivieren der „Special Effects“, sobald der Boss-Mutant gesiegt wurde

4.1.2.2 Properties

- Move Speed: Die Bewegungsgeschwindigkeit der Mutanten (default setting: 0.5 (Enemy, Enemy2), 1 (Boss))
- Max Damage Takes: Die benötigten Treffer mit dem Paket, um den Mutanten besiegen zu können (default setting: 0 (Enemy), 2 (Enemy2), 7 (Boss))

4.1.2.3 Setup

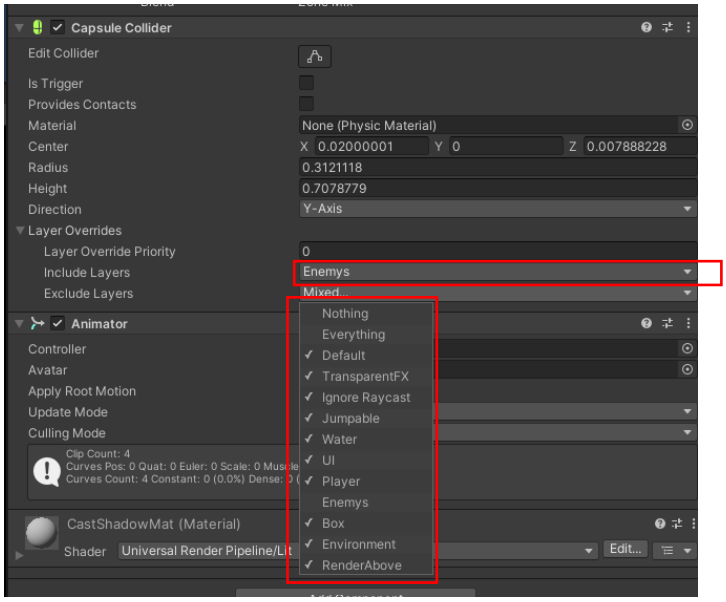
- Liegt auf: Enemy, Enemy2, Boss

  Hier bitte beachten!

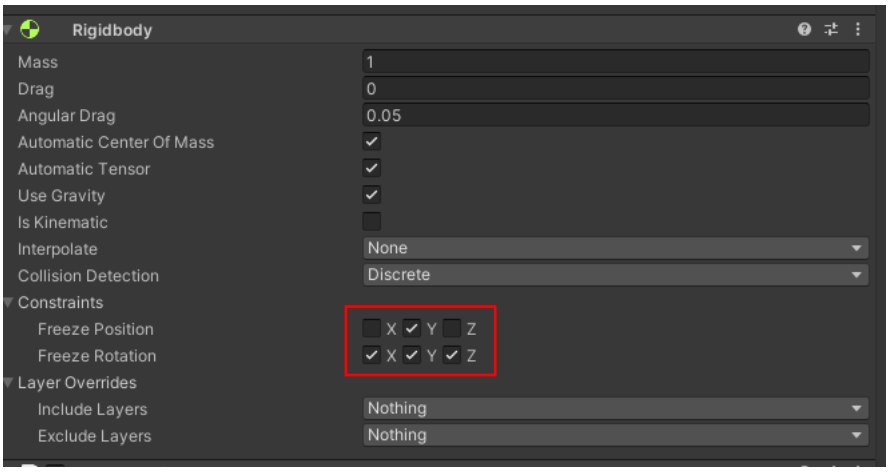
  - Alle vorher genannten GameObjects sind Prefabs und müssen im Ordner „Resources“ liegen
- Benötigt: Rigidbody, Capsule Collider, Animator, Audiosource

  Hier bitte beachten!

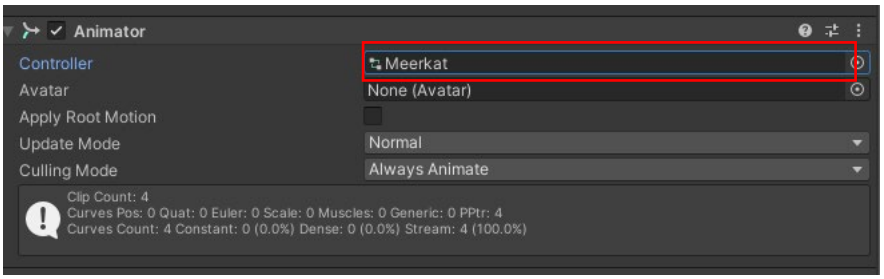
  - Es ist wichtig, das der Capsule Collider einen gewissen Abstand besonders an den Seiten aufweist, da dieser dafür da ist, das die Mutanten nicht zu nah aneinander kommen können
  - Ebenfalls die Layer Auswahl beachten! (siehe Bild)



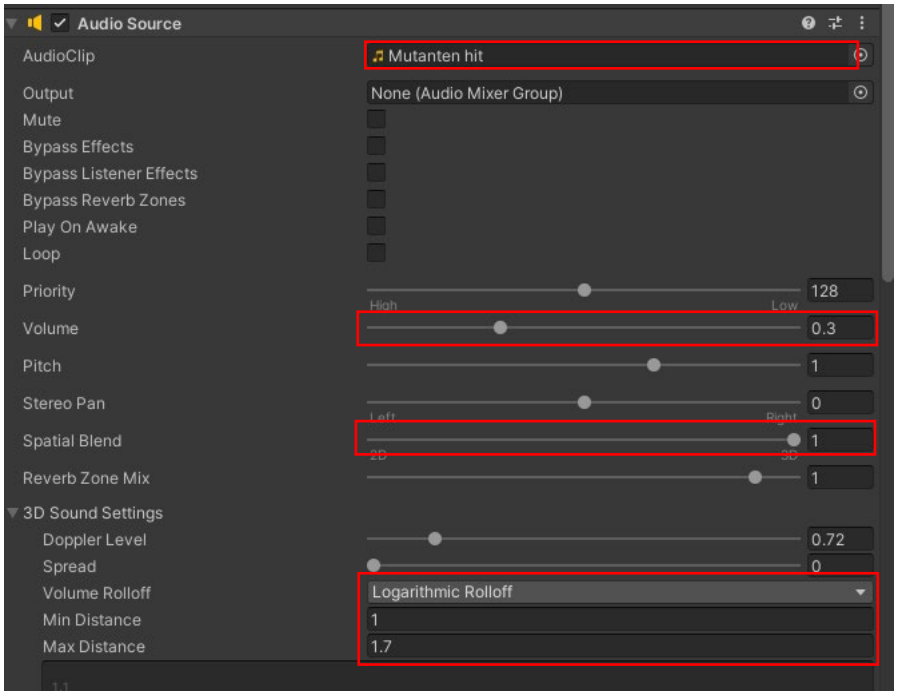
  - Alle Achsen sollen bei der Rotation des Rigidbodies gefreezed sein sowie die Y-Achse bei Position



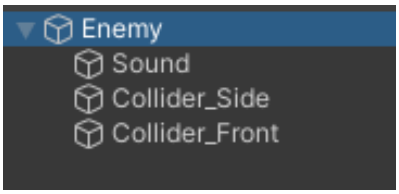
- Der Animator erhält den Controller des jeweiligen Mutanten Typen („Enemy“ -> „Meerkat“, „Enemy2“ -> „DeerFemale“, „Boss“ -> „DeerMale“)



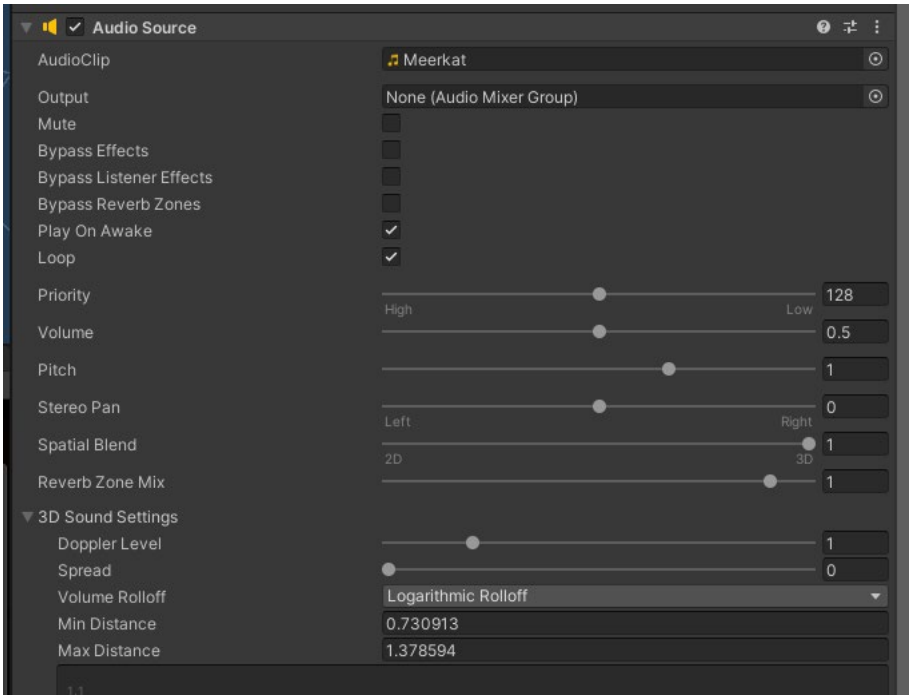
- Die Audiosource benötigt bei allen Enemy-Typen den Audioclip „Mutanten hit“ (hier bitte auch die Einstellungen des Sounds beachten -> siehe Bild)

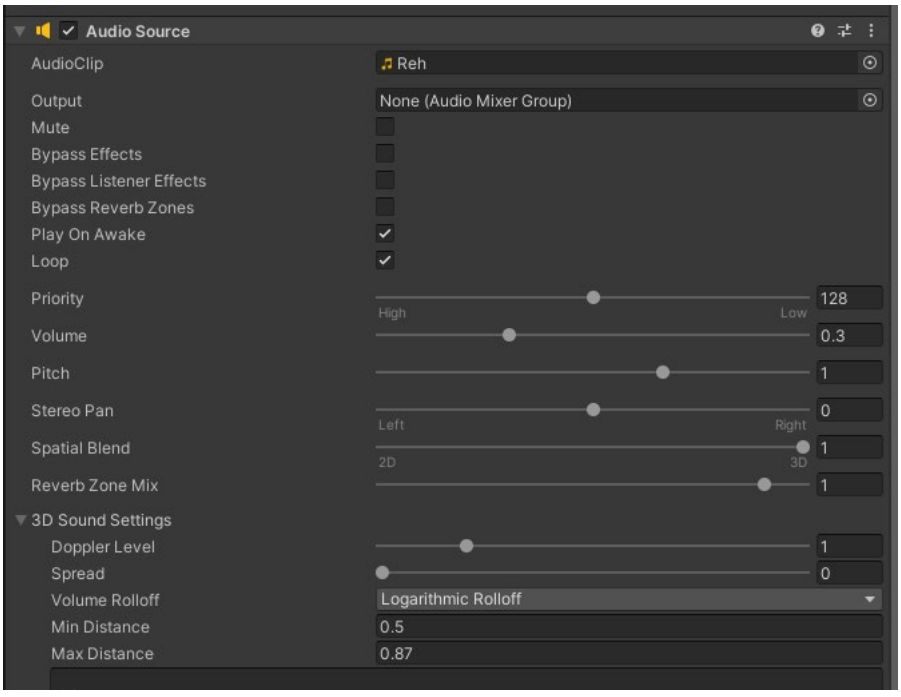


- Das GameObject sollte außerdem noch die Emtpy GameObjects „Sound“, „Collider_Side“ und „Collider_Front“ als Child-Objekte bei sich tragen

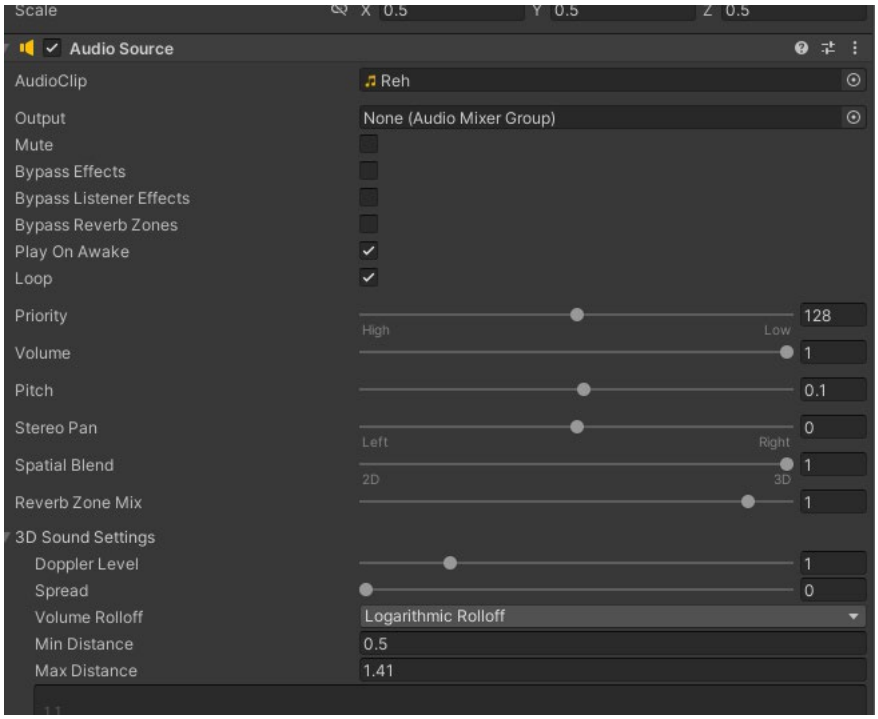


- „Sound“ trägt eine Audiosource Komponente mit dem Soundclip des Enemy-Typen („Enemy“ -> „Meerkat“, „Enemy2“ und „Boss“ -> „Reh“). Hier bitte auch die jeweilige Soundeinstellung beachten!



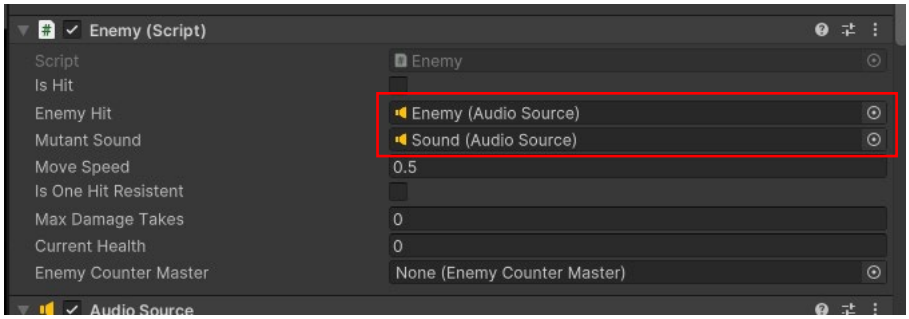


(Enemy2)

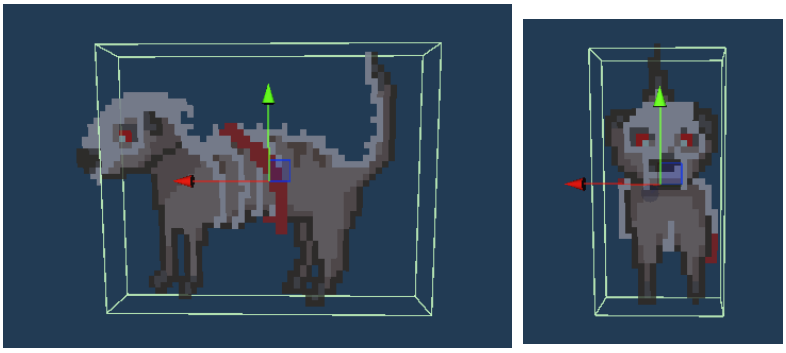


(Boss)

- Nun müssen noch „Enemy Hit“ und „Mutant Sound“ bei dem Script zugewiesen werden. Der erste Sound wird direkt von dem GameObject benutzt, während bei dem zweiten das Child-Objekt „Sound“ in das Feld gezogen werden muss



- „Collider_Side“ und „Collider_Front“ tragen einen Box Collider auf sich. Hier wird dann der Collider von „Side“ auf die passige Größe eines Seiten-Profils des Mutanten angepasst, während „Front“ auf ein Front-Profil angepasst wird. (Zur Erklärung: Je nachdem, welche Animations-Laufrichtung dann abgespielt wird, wird der Collider mit gewechselt. Sonst wäre die Hit-Box in 50% der Fälle nicht korrekt und würde den Spieler frustrieren).



4.1.3 EnemySpawner.cs

→ Dieses Script ist für das Spawnen und Zählen der Mutanten verantwortlich, und gibt den Wert des Mutantenzählers im UI wieder.

4.1.3.1 Funktionen

- Das Laden der passigen Mutanten-Prefabs aus dem Resources Folder
- Das Spawnen/Initiieren der Mutanten an der Position, wo sich das Objektes in der Szene befindet, das das Script trägt (benannt als „EnemySpawner“)
- Den Mutanten Zähler im UI um 1 addieren, wenn ein Mutant gespawnt wurde (diese Information wird an das EnemyCounterMaster.cs geleitet, wo die Informationsverarbeitung nun stattfindet)

4.1.3.2 Properties

- Time Between Spawns: Die Sekunden, die bis zum nächsten Spawn eines Mutanten verstreichen sollen (default setting: 2)
- Max Enemys: Die insgesamte Anzahl an Mutanten, die an diesem Spawn-Punkt erscheinen sollen

4.1.3.3 Setup

- Liegt auf: GameObjects mit dem Namen „EnemySpawner“
- Benötigt: nichts (das Objekt ist ein empty GameObject)

4.1.4 BossSpawner.cs

→ Dieses Script ist ausschließlich für das Spawnen des Bosses (in Level 2) verantwortlich, und zählt die aktuelle Mutanten Anzahl im UI um 1 hoch.

4.1.4.1 Funktionen

- Das Laden der Boss-Prefabs aus dem Resources Folder
- Das Spawnen/Initiieren der Mutanten an der Position, wo sich das Objekt in der Szene befindet, das das Script trägt (benannt als „BossSpawner“)
- Das permanente Prüfen, ob die Anzahl der Mutanten bei 0 liegt (hier wird insbesondere „EnemyCounterMaster.cs“ abgefragt) und damit den Spawn einzuleiten
- Das Abgleichen des Parameters, das der Boss mehr als einen Schaden aushalten kann
- Nach dem Spawn des Bosses den Counter aus EnemyCounterMaster.cs um 1 erhöhen und damit „nicht 0“ setzen (was dann automatisch für den Spieler bedeutet, das er erst den Boss besiegen muss, bevor das Paket abgegeben werden kann)

4.1.4.2 Properties

- Max Enemys: Die insgesamte Anzahl an Mutanten, die an diesem Spawn-Punkt erscheinen sollen (default setting: 1)

4.1.4.3 Setup

- Liegt auf: GameObject mit dem Namen „BossSpawner“ (Level 2)

- Benötigt: nichts (das Objekt ist ein empty GameObject)

4.1.5 EnemyCounterMaster.cs

→ Dieses Script ist, wie der Name schon sagt, für den UI Zähler der Enemys in der Szene zuständig und zählt die gesamte Menge aller Enemy-Typen hoch (sowohl die normalen, also auch mögliche Bosse).

4.1.5.1 Funktionen

- Permanentes Abgleichen und Überschreiben des UI EnemysCounter Textes mit dem Wert des internen Zählers
- **Hier bitte beachten!**
- Der interne Zähler wird über eine Verlinkung in den Spawner-Scripten hochgezählt

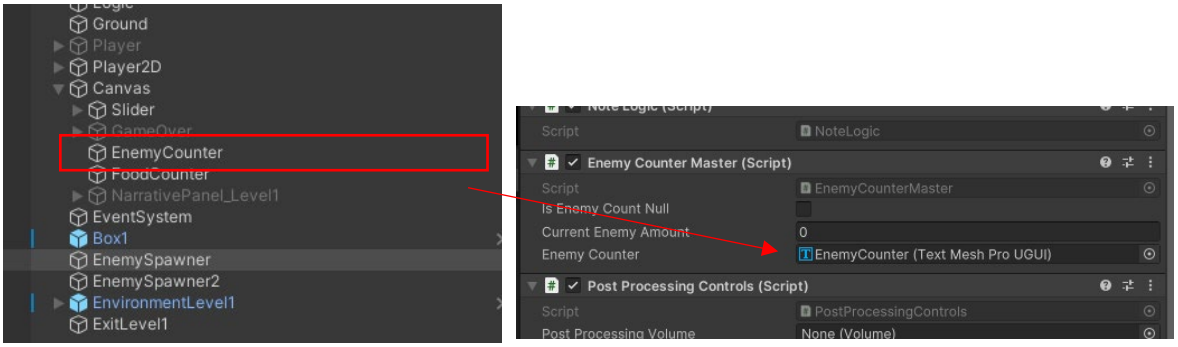
- Das Sicherstellen mithilfe einer bool „isEnemyCounterNull“, ob die Menge an Enemys bei 0 liegt (-> Diese Abfrage ist wichtig, damit der Spieler das Paket nur zum Zielort liefern kann, wenn alle aktiven Enemys beseitigt wurden)

4.1.5.2 Properties

- /

4.1.5.3 Setup

- Liegt auf: GameObjects mit dem Namen „Logic“
- Benötigt: /
- Das UI Element „EnemyCounter“ sollte bei dem Script unter „Enemy Counter“ referenziert sein (Canvas > EnemyCounter)



4.1.6 EnemyTrigger.cs

→ Dieses Script setzt bestimmte Enemy Spawner in der Szene aktiv, wenn der Spieler den Collider des Objektes berührt, welches das Script trägt. Es wird verwendet, um Enemys in der Szene erst spawnen zu lassen, wenn sich der Spieler in der Nähe befindet.

4.1.6.1 Funktionen

- Das Sammeln aller Enemy Spawner, die von diesem Trigger beeinflusst werden sollen (über eine List, die im Inspector gefüllt werden kann)
- Das Deaktivieren aller ihm zugewiesenen Enemy Spawner zu Beginn
- Das Aktivieren aller ihm zugewiesener Enemy Spawner
- Das permanente Prüfen, ob ein GameObject mit dem Tag „Player“ den Collider des eigenen Objektes berührt hat und das Auslösen der Aktivierungsfunktion, sobald diese Bedingung erfüllt wurde

4.1.6.2 Properties

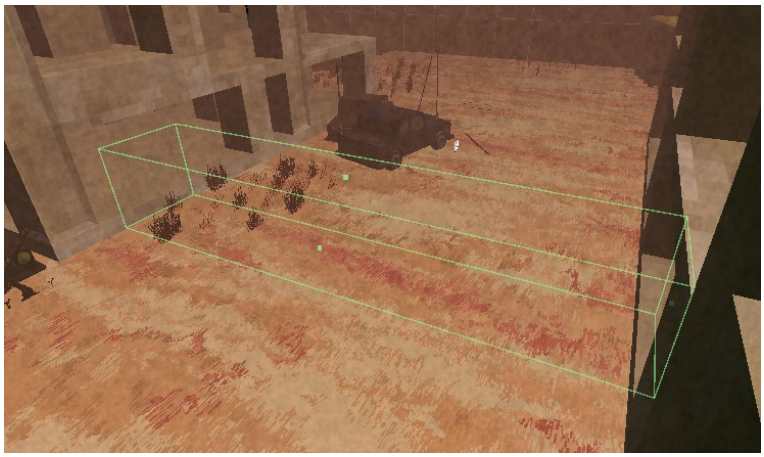
- Enemy Spawners: Eine dynamische Sammlung, in der alle Enemy Spawners per Drag’n’Drop verwiesen werden müssen, die für diesen Trigger beeinflusst werden sollen (-> mehr dazu unten)

4.1.6.3 Setup

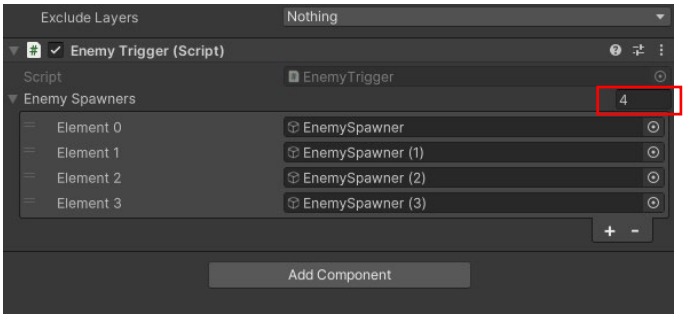
- Liegt auf: allen GameObjects mit dem Namen „EnemyTrigger“ (Level 1)
- Benötigt: Box Collider

Hier bitte beachten!

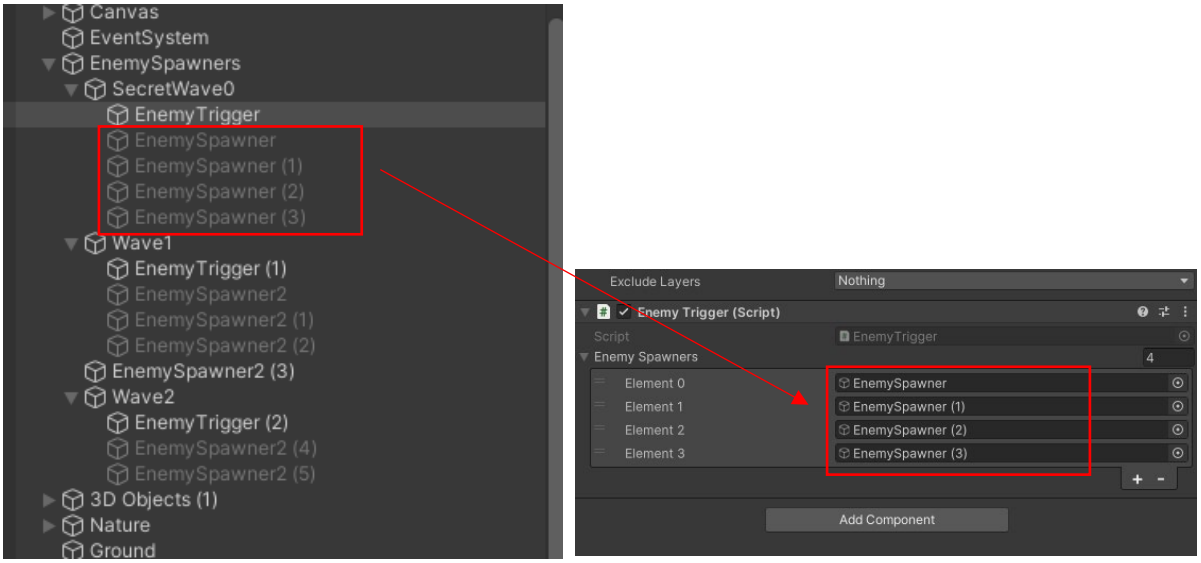
- Der Box Collider muss einen Haken bei „Is Trigger“ haben
- Bei der Größe und Platzierung des Box Colliders darauf achten, das er auch wirklich von dem Spieler erreicht werden kann und nicht ausweichbar ist (also möglichst mit beiden Enden in anderen Collidern steckt)



- Für das Füllen der Liste „EnemySpawners“ muss zuerst die Menge der Listenkomponenten angegeben werden



➢ Danach müssen die jeweiligen gewünschten Enemy Spawner GameObjects per Drag'n'Drop in die leeren Elementfelder gezogen werden

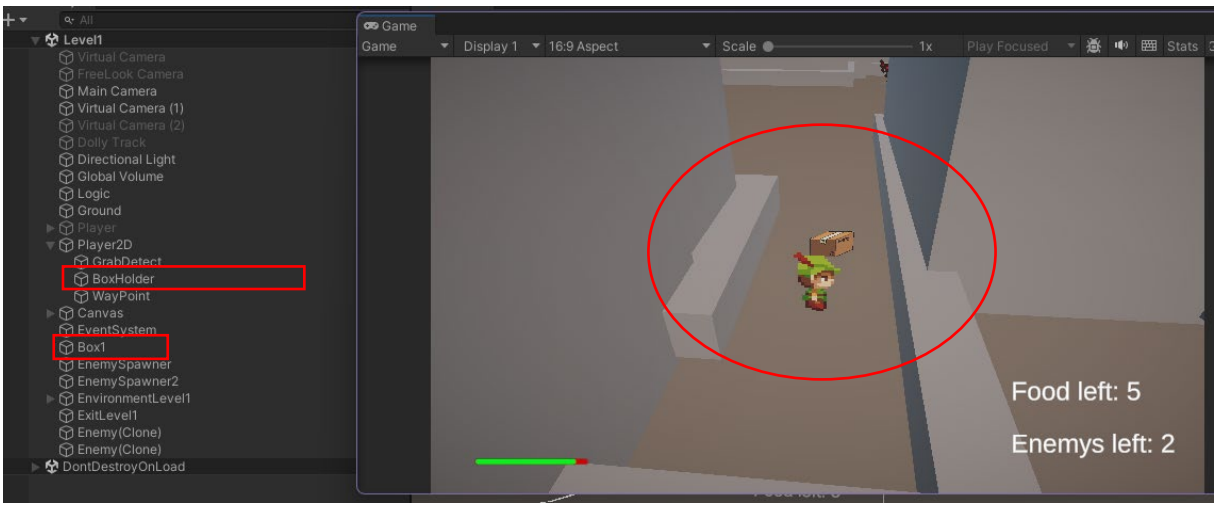


4.1.7 BoxInteraction.cs

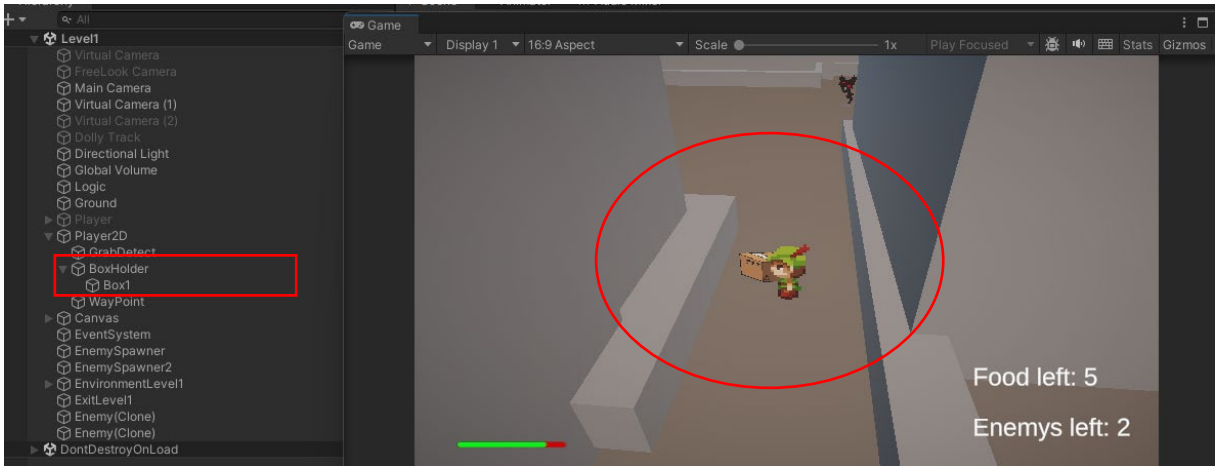
→ Dieses Script ist dafür zuständig, das das Paket vom Spieler aufgenommen, gehalten und in die Laufrichtung des Spielers geworfen werden kann, und gleichzeitig prüft, ob ein Mutant durch ein fliegendes Paket getroffen wurde.

4.1.7.1 Funktionen

- Das Prüfen der Position des GameObjects, welches das Script trägt, auf der y-Achse (damit kann bestimmt werden, ob sich das Objekt in der Luft oder auf dem Boden befindet)
- Das Unterordnen und Umpositionieren des Objektes an dem Child empty GameObject „BoxHolder" des Spielcharacters, wenn „LMB" gedrückt wird und der Spieler nah genug an dem Objekt dran ist



Spieler trägt das Paket nicht -> „BoxHolder" ist noch leer und „Box1" (das Paket) ist ein freies GameObject in der Hierarchie mit eigener Position



Spieler trägt das Paket -> „BoxHolder“ ist nun gefüllt und „Box1“ ist ein untergeordnetes Child von „BoxHolder“ geworden. Wenn sich nun der Spieler („Player2D“) bewegt, werden sich alle untergeordneten Objekte mitbewegen.

- Das „Freilassen“ (Unterordnung auflösen) des GameObjects von „BoxHolder“, wenn erneut „LMB“ gedrückt wird und der Spieler das Paket in den Händen hat
- Durch das „Freilassen“ wird der Wurf aktiviert, wodurch das Objekt einer bestimmten Kraft ausgesetzt wird und es in die Richtung fliegt, in die die Spielfigur schaut
- Durch das Werfen wird auch ein Partikel Effekt ausgelöst, der optisch den aufwirbelnden Sand in Level 1 darstellen soll, wenn das Paket auf dem Boden landet (ebenfalls visuelles Feedback)
- Das permanente Prüfen, ob sich das GameObject auf dem Boden befindet
- Das Erkennen, ob sich ein anderes GameObject mit dem Tag „Player“ nah genug an dem Paket befindet, um es aufzunehmen
- Das Erkennen, ob sich ein anderes GameObject mit dem Tag „Enemy“ nah genug an dem Paket befindet, um als „Getroffen“ registriert zu werden
- Durch das Treffen eines solchen GameObjects wird ein Rückstoß aktiviert, wodurch das Objekt einer bestimmten Kraft ausgesetzt wird und es in die Richtung zurückfliegt, aus der es gekommen ist
- Das Erkennen, ob das Paket ein Objekt mit dem Tag „Border“ in der Luft getroffen hat

4.1.7.2 Properties

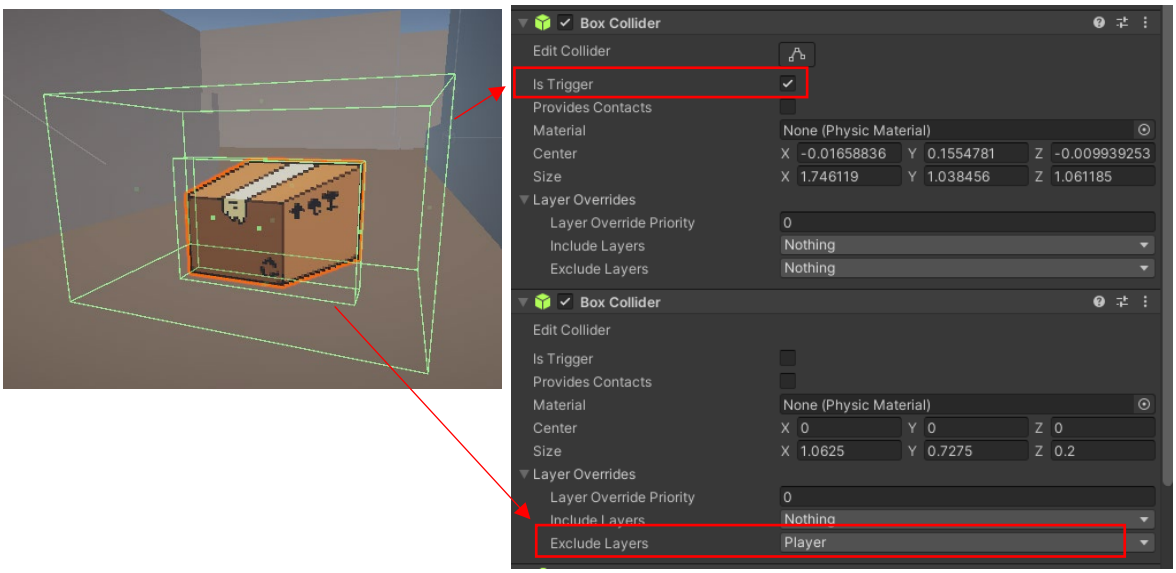
- Throw Speed: Die Geschwindigkeit/Kraft, mit der das Paket geworfen wird (default setting: 1000)
- Back Speed: Die Geschwindigkeit/Kraft, mit der das Paket zurückstößt, wenn es einen Mutanten trifft (default setting: 1500)

4.1.7.3 Setup

- Liegt auf: GameObjects mit dem Namen „Box“ (in Level 1 -> „Box1“, in Level 2 -> „Box2“, …)
- Benötigt: Rigidbody, Box Collider, Audiosource

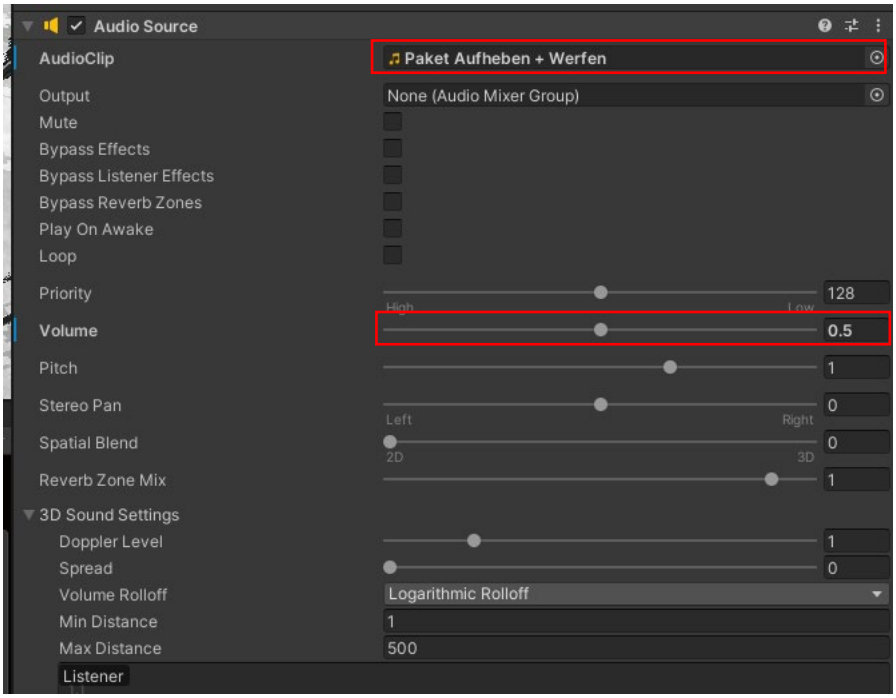
Hier bitte beachten!

- Es ist wichtig, das der Collider einen gewissen Abstand besonders an den Seiten aufweist, da dieser Collider den Interaktions-Radius für den Spieler definiert
- Der abstehende Collider muss einen Haken bei „Is Trigger“ haben
- Außerdem die Exclude Layers beachten

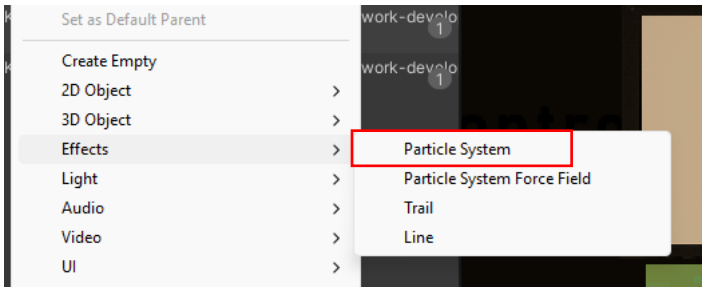


➢ ÄNDERUNG! -> der zweite Box Collider, der das Objekt an sich einschließt und mit anderen Collidern kollidieren kann (im Bild der kleinere Collider), ist als eigenes Child Objekt ergänzt worden. Die Einstellungen bleiben aber die selben!

- Die Audiosource benötigt den Audio Clip „Paket Aufnehmen + Werfen“ sowie die Einstellungen im Bild

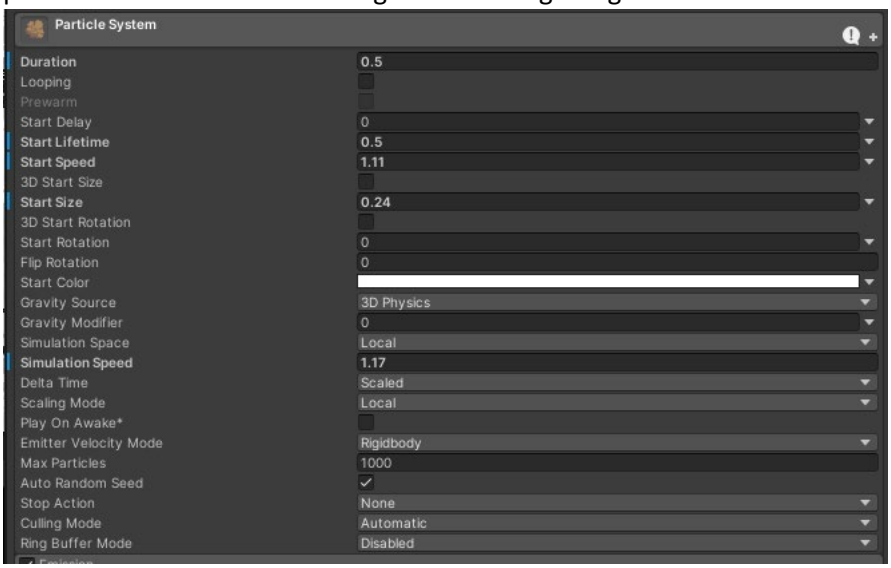


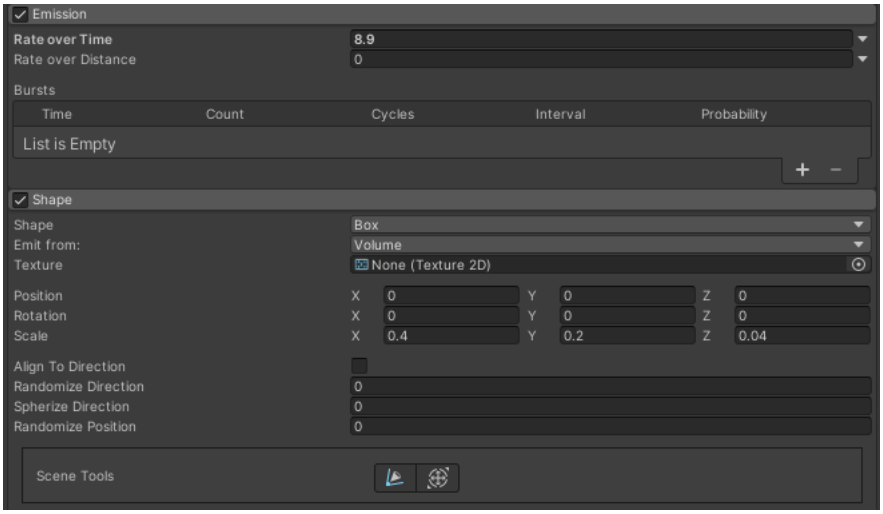
- Diese Audiosource Komponente muss dann unter „Box“ (gesuchter Sound im Script) referenziert werden
- Außerdem trägt das GameObject ein weiteres Child-Objekt mit der Particle System Komponente

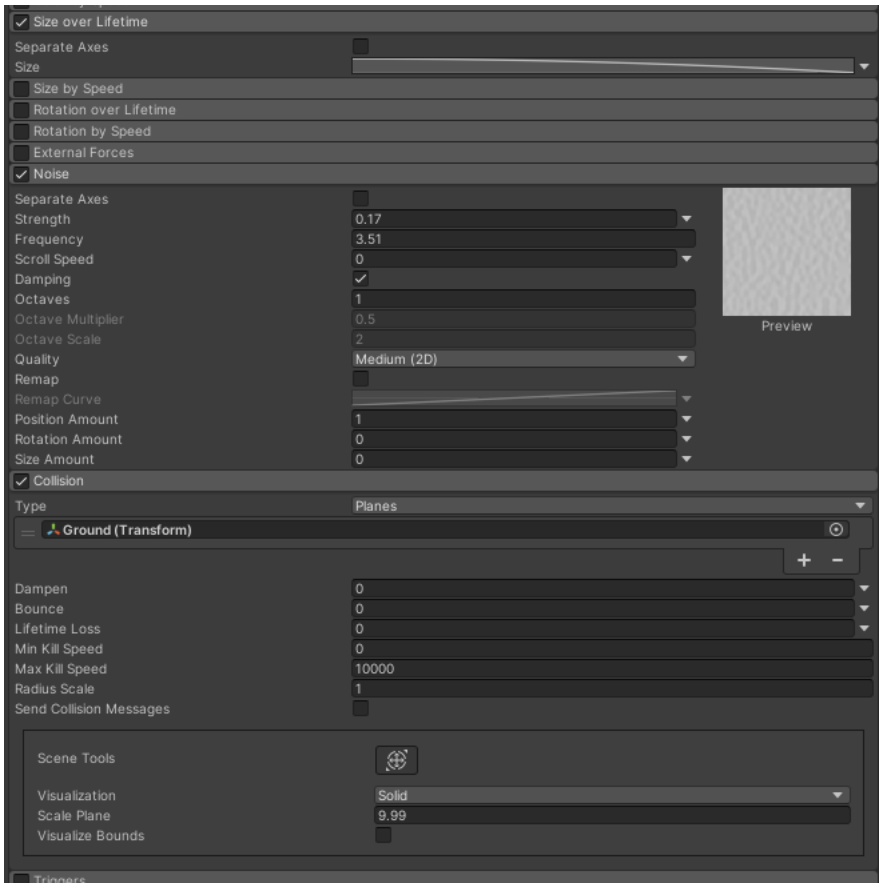


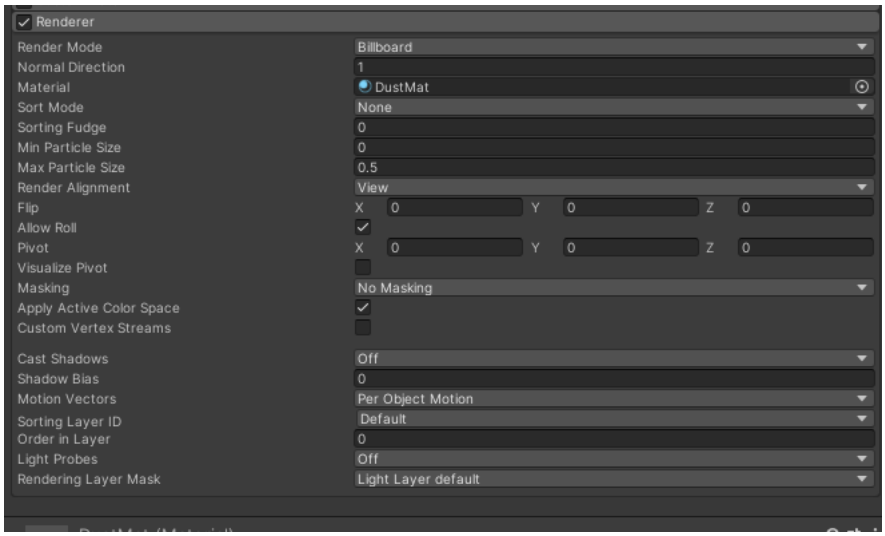
Hier bitte beachten!

- Für einen etwa gleichen Effekt sollte der Partikel Effekt unterhalb des Paketes positioniert werden und die folgenden Settings eingestellt werden:

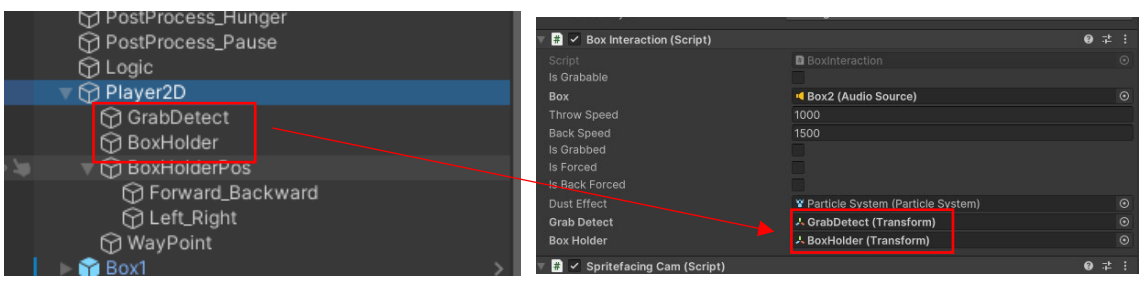








- Dieses Child muss unter „Dust Effect“ referenziert werden
- AUßERDEM WICHTIG!! -> Die Child-Objekte vom Player „GrabDetect“ und „BoxHolder“ müssen unter den gleichnamig gesuchten Variablen auf dem Script verwiesen werden



4.1.8 Boarder.cs

→ Dieses Script ist ähnlich aufgebaut wie das Enemy.cs und gibt sogenannten „Boarders“, also Barrieren, auf der Map die Fähigkeit, von dem Spieler kaputtgeschlagen werden zu können. Das Script ist für die Schadensaushaltbarkeit und das visuelle Feedback zuständig.

4.1.8.1 Funktionen

- Das Zurücksetzen des erhaltenen Schadens auf 0
- Das Prüfen, ob man von dem geworfenen Paket des Spielers getroffen wurde
- Das Hochzählen des Schaden Counters um 1, wenn man getroffen wurde und der Schadens Counter noch nicht die max. Menge an Schaden erreicht hat
- Das Despawnen der Boarder, wenn die max. Menge an Schaden erreicht wurde

4.1.8.2 Properties

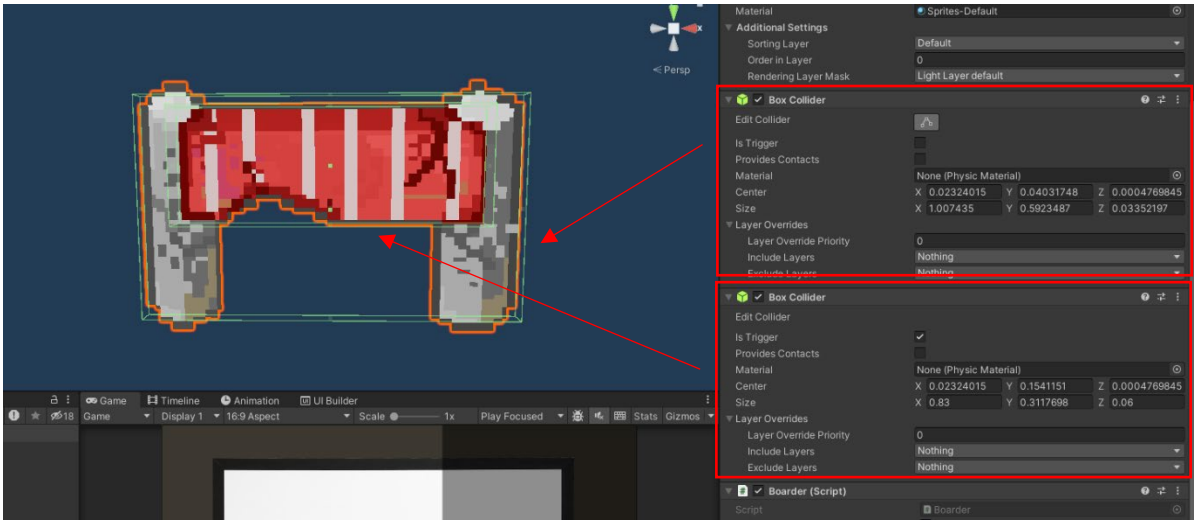
- Is One Hit Resistent: Kann das GameObject/die Boarder mehr als 1 Schaden aushalten?
- Max Damage Takes: Die maximale Menge an Schaden, die diese Boarder aushalten kann bzw. die Anzahl, wie oft der Spieler das Paket gegen die Boarder schmeißen muss, bis sie kaputt geht

4.1.8.3 Setup

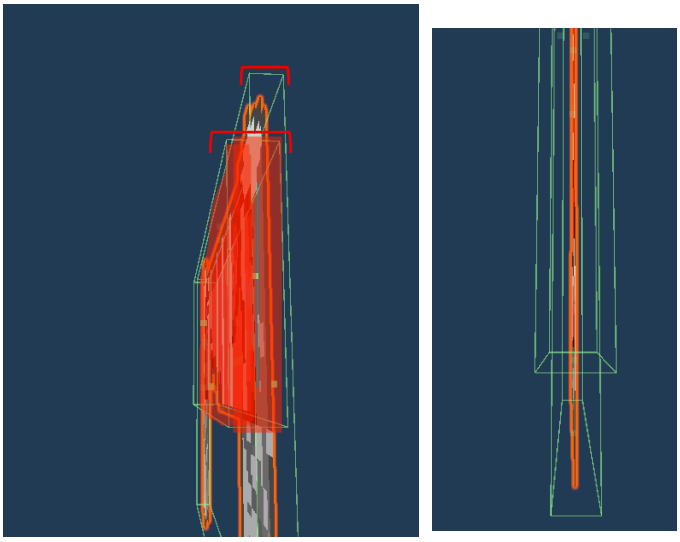
- Liegt auf: allen GameObjects mit dem Namen „Boarder“ (ist unter dem Ordner „Prefabs zu finden!)
- Benötigt: Box Collider (allgemein, damit man nicht durchlaufen kann), zweiter Box Collider (später für das Treffen mit dem Paket wichtig)

Hier bitte beachten!

➢ Beide Box Collider sollten in etwa wie in dem Bild unten eingestellt sein



➢ Der kleinere Collider sollte etwas weiter von dem Objekt abstehen bzw. breiter gezogen sein als der große Collider, damit dieser später noch von dem Paket beeinflusst werden kann



➢ Der kleinere Collider (im Bild der untere), muss einen Haken bei „Is Trigger“ haben

4.1.9 NoteInteraction.cs

→ Dieses Script ist für das Interagieren, Aufnehmen und Anzeigen von aufgesammelten Notizen in der Szene zuständig.

4.1.9.1 Funktionen

- Das Prüfen, ob der Spieler in der Nähe ist und „Q“ drückt
- Das Abfragen vom Namens des Notiz-Objektes und das darauffolgende Auslösen und Aktivieren des Notiz-Panels über das „NoteLogic.cs“ mit dem richtigen Image-Namen, wenn die Bedingung erfüllt wurde
- Das Deaktivieren des Notiz-Objektes in der Szene, damit es für den Spieler nicht mehr zu sehen ist
- Das Pausieren der Ingame-Zeit, sodass das Spiel wie angehalten ist
- Das Ändern des Notiz-Sprites und das Anzeigen der Interaktionstaste, sobald der Spieler in der Nähe steht und das Paket nicht in der Hand hält
- Das Zurücksetzen des Notiz-Sprites, wenn der Spieler nicht mehr in der Nähe ist

4.1.9.2 Properties

- (Interact Mat: Das Material, was auf den Sprite Renderer gelegt wird, wenn der Spieler in der Nähe ist -> ist bereits fertig gesetzt und über das Script verbunden (15.06.))

4.1.9.3 Setup

- Liegt auf: allen GameObjects mit einem eindeutigen Notiz-Namen -> ein Prefab zu den Notiz-Objekten liegt unter dem Ordner „Prefabs“

Hier bitte beachten!

➢ Die Notizen müssen nach einem geordnetem Schema benannt sein! Die Namen der Notiz-Objekte sind wichtig für das Script „NoteLogic.cs“. In diesem Script wird der Name des Notiz-Objektes, den er aus dem „NoteInteraction.cs“ bekommt, mit dem

Namen des Images abgeglichen, das er in das Fenster lädt, wenn mit der Notiz interagiert wird.

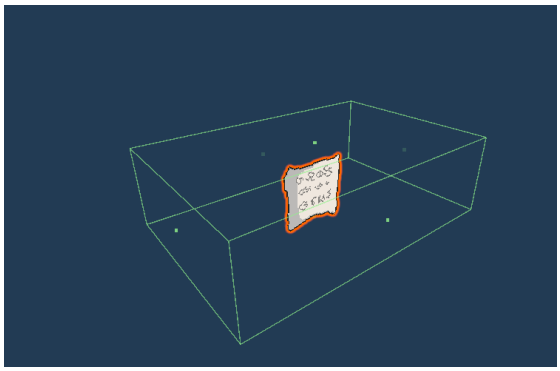
- Die Benennung: „Area(Nummer der Area)_Note(Nummer der Notiz)“
- Es gibt für Area 1 acht Notizen und für Area 2 sechs Notizen

- Eine genauere Erklärung folgt beim Abteil „NoteLogic.cs

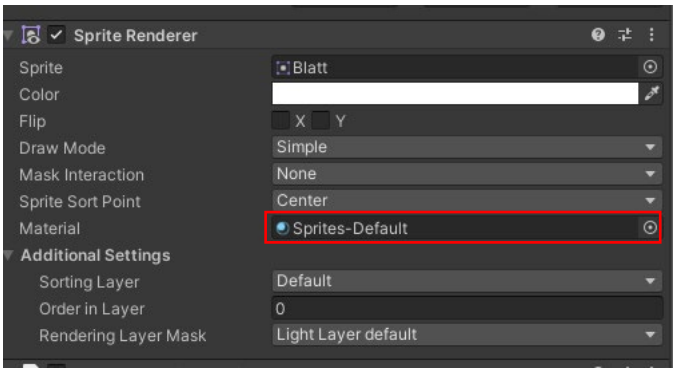
- Benötigt: Box Collider

  Hier bitte beachten!

  - Der Box Collider sollte nicht zu klein sein und recht weit vom eigentlichen Objekt abstehen (siehe Bild)



  - Der Box Collider muss einen Haken bei „Is Trigger“ haben
- Bitte auch sicherstellen, dass der Name des Materials im Sprite Renderer „Sprite-Default“ heißt



4.2 UI Elemente Logiken

4.2.1 EnergyBar.cs

→ Dieses Script ist für das Managing und korrekte anzeigen der übrigen Energie sowie Essensvorräte im UI zuständig, wie schnell die Energie im Spiel sinkt und wann der Game Over Screen durch Energiemangel abgespielt wird.

4.2.1.1 Funktionen

- Das Geben der maximalen Werte, wenn man in ein Level lädt
- Das Default setzen des maximalen Wertes für die Energiebar -> 100 wegen Einfachheitshalber und leichter zum Balancen und Berechnen der anderen Werte
- Das kontinuierliche Sinken der Energie in einer bestimmten Geschwindigkeit
- Das verschnellerte Sinken der Energie in einer bestimmten Geschwindigkeit, wenn der Spieler die Sprintfunktion verwendet
- Das Wiederherstellen einer bestimmten Energiemenge, wenn man „F“ (das Geben von Essen) drückt
- Das Runterzählen der Essensmenge um 1, wenn man „F“ gedrückt hat und das Deaktivieren des Kamera-Effektes bei starkem Hunger
- Das Sicherstellen, das die aufgefüllte Energie nicht höher sein kann als vorgegeben
- Das Laden des Game Over Screens, wenn die Energie-Bar auf 0 sinkt
- Das Aktivieren des Hunger-Cam Effekts, wenn die Bar bei 60% ist
- Das Blinken der Bar aktivieren bei 30%
- Das Abspielen vom Magenknurren bei 50%, 20% und 10%

4.2.1.2 Properties

- (Max Energie: Maximale Energie, die die Energiebar anzeigen kann)

  Hier bitte beachten!

  - Diesen Wert NICHT abändern
  - Er dient dazu, um das Berechnen und Balancen der unten aufgelisteten Werte zu erleichtern

- Food Energie: Die Energiemenge, die beim Drücken von „F“ der Energiebar hinzuaddiert wird (default setting: 100)
- Start Food: Die Menge an Essen, die der Spieler zu Beginn des Level zur Verfügung hat (default setting: 5)
- Energy Drain Rate: Die Menge, die der Energiebar pro Frame kontinuierlich abgezogen wird (default setting: 1)
- Sprint Drain Rate: Die Menge, die der Energie Drain Rate hinzuaddiert wird, wenn der Spieler sprintet (default setting: 3)

4.2.1.3 Setup

- Liegt auf: Logic (in den Level-Szenen)
- Benötigt: /

4.2.2 EnemyHealthbar.cs

→ Dieses Script ist für das Anzeigen der übrigen Leben eines Boss (in Level 2) auf einem UI Slider zuständig.

4.2.2.1 Funktionen

- Das Aufsuchen des Boss-Objektes in der Szene und dem darauf liegendem „Enemy.cs“ Scriptes (dort werden Informationen verwendet, die auch für den UI Slider von Bedeutung sind)
- Das selbstständige Aufsuchen des UI Sliders, auf dem später die Lebensanzeige des Bosses visualisiert wird

- Das Gleichsetzen des maximalen „Gesundheit“ des Bosses mit der Anzahl der nötigen Treffer, die der Boss zum Besiegen braucht (1)
- Das Gleichsetzen des maximalen Wertes von dem Slider mit der maximalen Gesundheit des Bosses (diesen Wert haben wir uns im Schritt davor geholt) (2)
- Das Default-Setzen des herunterzählenden Wertes vom Slider gleichgestellt mit der maximalen Gesundheit (sozusagen den Slider direkt am Anfang auf den maximalen Wert gebracht) (3)
- Einen dynamischen Wert „currentHealth“, der später weiter beeinflusst wird, ebenfalls mit der maximalen Gesundheit abgleichen (dieser Wert wird später die Übersetzung-Brücke für das Script sein, um den herunterzählenden Wert vom Slider zu beeinflussen) (4)

Im Script:

```
    HealthBar = GameObject.Find("Boss(Clo
(1) maxHealth = enemy.MaxDamageTakes;
(2) HealthBar.maxValue = maxHealth;
(3) HealthBar.value = maxHealth;
(4) currentHealth = maxHealth;
}
```

- Den herunterzählenden Wert vom Slider immer um 1 senken, wenn ein Treffer erzielt wurde (also bei einem Boss mit 8 Leben bei jedem Treffer 1/8tel der maximalen Gesundheit abgezogen wird)

4.2.2.2 Properties

- /

4.2.2.3 Setup

- Liegt auf: GameObject mit dem Namen „Logic“

  Hier bitte beachten!

  - Dieses Script wird nur im zweiten Level gebraucht

- Benötigt: /
- Das Script „Enemy.cs“ auf dem GameObject muss unter „E Bar“ referenziert sein

4.2.3 IconBlink.cs

→ Dieses Script lässt Icons und Images im Script blinken (wird zB. auf der Map verwendet).

4.2.3.1 Funktionen

- Das Auffinden der Image-Komponente auf dem GameObject, worauf das Script liegt, direkt zum Start
- Das Beeinflussen der Farbe vom Image, wo zwei frei im Inspector einstellbare Farben in einer bestimmten Geschwindigkeit hin und herwechseln und damit der Blink-Effekt simuliert wird

4.2.3.2 Properties

- Start Color: Die Farbe, die als erstes in der Routine angezeigt werden soll (default setting: FFFFFF – weiß)
- End Color: Die Farbe, die als zweites in der Routine angezeigt werden soll (default setting: 0BFF00 – grün)
- Speed: Die Geschwindigkeit, mit der die Farben wechseln sollen (default setting: 2)

4.2.3.3 Setup

- Liegt auf: den GameObjects mit dem Namen „Level 1“, „Level 2“ und „Level 3“ (die Markierungen/Symbole für die Level, in der Szene „Map“ zu finden)
- Benötigt: /

4.2.4 JohnsDiary.cs

→ Dieses Script ist für die Tagebucheinträge, die man auf dem Home Screen finden kann, zuständig, insbesondere das Klicken durch die Seiten, ohne für jede einzelne Button-Interaktion eine Funktion schreiben zu müssen.

4.2.4.1 Funktionen

- Das Sammeln aller Seiten mithilfe eines Arrays (wird später im Inspector gefüllt)
- Das Deaktivieren aller Seiten, bis auf die erste
- Eine Funktion, in der die aktuelle Seite deaktiviert und die nächste aktiviert wird (dabei soll das Script erst die nächsthöhere Seite in der Sammlung ausfindig machen)
- Eine Funktion, in der die aktuelle Seite deaktiviert und die vorherige aktiviert wird (dabei soll das Script erst die nächstniedrigere Seite in der Sammlung ausfindig machen)

4.2.4.2 Properties

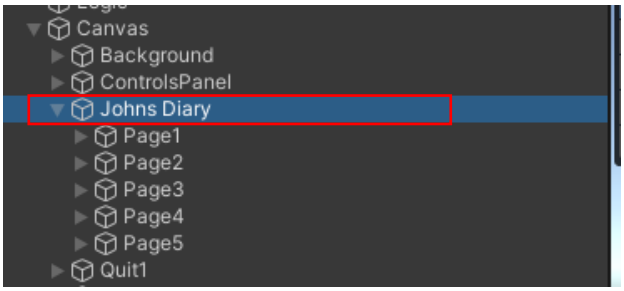
- /

4.2.4.3 Setup

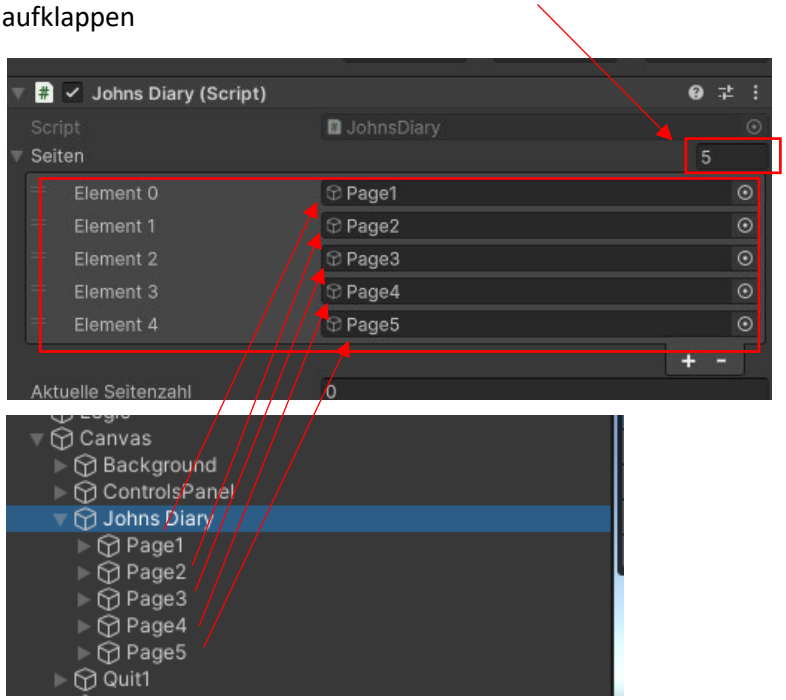
- Liegt auf: GameObject mit dem Namen „Johns Diary“ (in der Szene „HomeScreen“

  Hier bitte beachten!

  - Die UI Elemente sind auf dem ersten Blick in der Hierarchie vllt nicht zu sehen -> dafür das GameObject „Canvas“ aufklappen



- Benötigt: nichts (ist ein Emtpy GameObject nur mit dem Script)
- Alle Seiten-Panels des Tagebuches sollen für die Ordnung unter „Johns Diary“ liegen und wie in dem Bild oben benannt sein
- Zum Füllen der Sammlung im Script die max. Seitenanzahl bei „Seiten“ eintragen und aufklappen



- Danach jedes Seiten-Panel mit Drag’n’Drop in die feien Element-Felder ziehen

Hier bitte beachten!

➢ Die Reihenfolge ist wichtig! Element 0 ist immer Seite 1, das letzte Element die letzte Seite

4.3 Framework und Navigation Management

4.3.1 ScreenLogics.cs

→ Dieses Script stellt die Basis für das Laden von Szenen und Aktivieren bzw. Deaktivieren von Panels dar. Hier werden alle On Click Events geschrieben, die den Buttons im Spiel sagen, was

passieren soll, wenn sie gedrückt werden. Es übernimmt ebenfalls einen Teil der Speicherungs-Logic und schließt das Spiel.

4.3.1.1 Funktionen

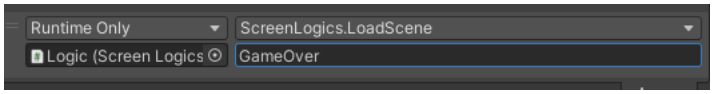
- Deaktivieren aller Panels und GameObjects, die zum Start nicht benötigt werden
- Das Abfragen direkt zum Start, ob eine Speicherung im Hintergrund geschehen ist, als der Spieler ein Level beendet hat (hier die Verwendung von PlayerPrefs) und das Laden der Map-Scene sowie der Homescreen-Scene mit dem jeweiligen Ergebnis (beendete Level werden grün markiert, der „Load Button“ wird nur angezeigt wenn es was zu laden gibt)
- Der IconBlink, je nachdem welches Level bereits beendet wurde und welches als nächstes zu spielen ist, sowie das Einblenden der „Demo-End“ Meldung, wenn beide Level beendet wurden
- Das Hochzählen eines Timers, solange der Spieler nicht das Zielgebäude erreicht hat
- Das Aktvieren des Pause Panels, sobald die Taste „P“ oder „Esc“ gedrückt wurde, sowie das Pausieren von jeglichen Bewegungen in der Szene und das Anpassen von Sound und Cam Effekt
- Eine Methode für das Laden von Szenen, bei der der Name der Szene angegeben werden muss

  Im Script:

```
SceneManager.LoadScene(sceneName);
```

```
SceneManager.LoadScene("GameOver");
```

  In der Engine:



- Eine Methode für das Laden von Level-Scenes, bei der die Nummer des Levels angegeben werden muss, und eine damit verbundene Methode, die die Levelnummer immer um 1 erhöht und das nächste Level lädt (wird zB. verwendet, wenn der Tutorialtext in Level 2 durchgelaufen ist und die Scene „Level2“ geladen werden soll)
- Eine Methode, die im Game Over Screen verwendet wird und das zuvor gespielte Level restartet
- Eine Methode zum Beenden/Schließen des Spiels
- Eine Methode, um alle Speicherungen im Spiel zu reseten und damit ein neues Spiel zu starten
- Das Aktivieren und Deaktivieren von Panels mit den jeweiligen optischen und akustischen Anpassungen
- Die Information abspeichern, wenn der Spieler ein Level beendet hat
- Eine Methode, mit der das Spiel geschlossen werden kann
- Das Abspielen des Klick-Sounds für die Button
- Das Addieren von Essen + das Wechseln der Task im UI Questboard + das Deaktivieren von Collidern bei den Barrieren in den Szenen, sobald der Spieler das Paket abgegeben hat

4.3.1.2 Properties

- Food Reward: Die Essensmenge, die der Spieler erhält wenn er das Zielobjekt erreicht (default setting: 3 (Level1), 5 (Level2))

4.3.1.3 Setup

- Liegt auf: alle GameObjects mit dem Namen „Logic“

- Benötigt: /
- „Button Click“ muss mit der passig benannten Audiosource verbunden werden (diese liegt unter dem GameObject „Other Audios“)
- „Food“ muss mit der passig benannten Audiosource verbunden werden (diese liegt ebenfalls unter dem GameObject „Other Audios“)

4.3.2 NoteLogic.cs

→ Dieses Script ist für das korrekte Anzeigen einer eingesammelten Notiz verantwortlich sowie die interne Informationsabspeicherung, das eine bestimmte Notiz eingesammelt wurde.

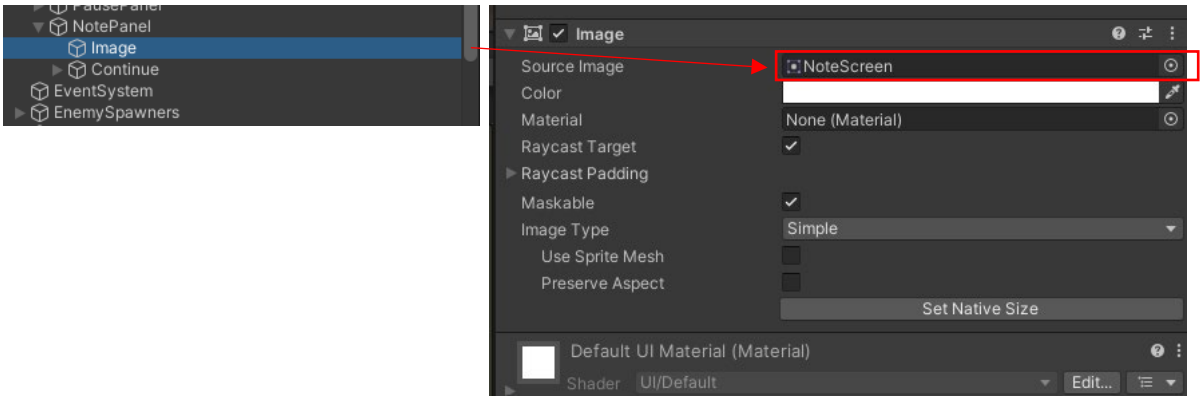
4.3.2.1 Funktionen

- Das Deaktivieren des Note Panels (also das Fenster mit der Notiz) direkt zu Beginn
- Eine Funktion „OpenPanel“, in der gespeichert wird, wenn eine bestimmte Notiz eingesammelt wurde (für die genauere Bestimmung, welche Notiz es war, wird der Name abgefragt -> diesen Namen erhalten wir von „NoteInteraction.cs“)

```
PlayerPrefs.SetString(noteName, " collected");
PlayerPrefs.Save();
```

- Außerdem das Aktivieren des Note Panels und das Laden des korrekten Images (mithilfe der Namensabfrage der Notiz) über den Resources Folder, sobald die „OpenPanel“ Funktion aufgerufen wird

- Erklärung zum Füllen der Image-Komponente mit der korrekten Notiz:



4.3.2.2 Properties

- /

4.3.2.3 Setup

- Liegt auf: GameObject mit dem Namen „Logic“
- Benötigt: /

4.3.3 FileCollectionLogic.cs

→ Dieses Script ist dafür zuständig, auf einfachem Weg die passende File in der File Collection ansehen zu können, wenn man das jeweilige Vorschaubild angewählt hat.

4.3.3.1 Funktionen

- Das Ausfindigmachen aller bisher eingesammelten Notizen/Files in den Leveln
- Das erkenntlich machen, das eine File eingesammelt wurde, indem u.a. die Transparenz vom „Vorschaubild“ genommen wird
- Eine Funktion, in der ein Panel mit der vergrößerten Notiz angezeigt wird und das Bild darauf mit dem Image mit dem selben Namen wie das angeklickte Vorschaubild aus dem Resources Folder geladen wird
- Eine Funktion, die das Panel wieder schließt (diese wird dann aufgerufen, wenn der Spieler „Back“ drückt

4.3.3.2 Properties

- /

4.3.3.3 Setup

- Liegt auf: GameObject mit dem Namen „Logic“
- Benötigt: /

Hier bitte beachten!

- Dieses Script wird ausschließlich in der Szene „FileCollectionScreen“ gebraucht

4.3.4 MapMouseHover.cs

→ Dieses Script zeigt die jeweiligen Auftrags-Fenster neben einem Level auf der Map an, wenn über das Icon des Levels gehovert wurde.

4.3.4.1 Funktionen

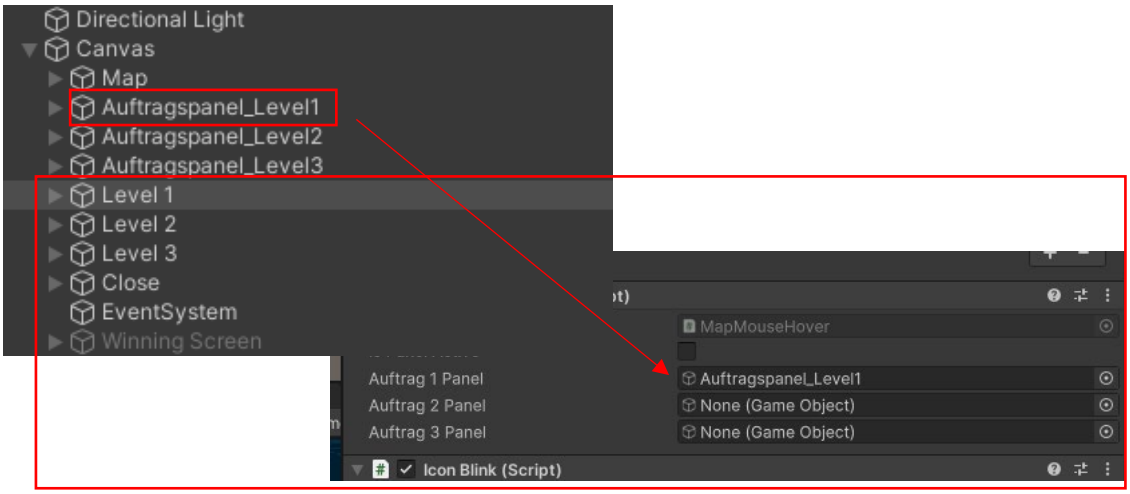
- Das Aktivieren des Level1-Auftrag Fensters, wenn sich der Mauszeiger über dem Icon von Level 1 befindet
- Das Aktivieren des Level2-Auftrag Fensters, wenn sich der Mauszeiger über dem Icon von Level 2 befindet
- Das Aktivieren des Level3-Auftrag Fensters, wenn sich der Mauszeiger über dem Icon von Level 3 befindet
- Das Deaktivieren jeglicher Auftrags-Fenster, sobald der Mauszeiger über dem Hintergrundbild (also der Map) ist -> somit wird automatisch ein Hover Effekt für die Icons nachgestellt

4.3.4.2 Properties

- /

4.3.4.3 Setup

- Liegt auf: GameObjects mit dem Namen „Level 1“, „Level 2“ und „Level 3“
- Benötigt: /
- Je nach Level-Typ muss das richtige Auftrags-Fenster zugewiesen werden (per Drag’n’Drop)



(Beispiel: „Level 1“ hat das Script und benötigt die Referenzierung zu dem „Auftragspanel_Level1“

Hier bitte beachten!

- Bei der Referenzierung NUR das benötigte Panel zuweisen! Es bleiben also immer 2 der 3 Felder leer.

4.3.5 PostProcessingControls.cs

→ Dieses Script ist für das Beeinflussen von Post Processings, oder auch Global Volumes, zuständig. Hier wird vor Allem der Hunger-Effekt, der in der Cam zu sehen ist, gemacht.

4.3.5.1 Funktionen

- Das Ausfindig machen des „Vignette“ Komponenten auf der Volume zum Start
- Das Erhöhen der Intensität von dem Vignette Effekt, je mehr Zeit im Spiel vergeht (mithilfe von Time.deltaTime)
- Eine Funktion, in der die Intensität zurück auf 0 gestellt wird (diese Funktion wird aufgerufen, wenn der Spieler dann mit „F“ die Leiste auffüllt)

4.3.5.2 Properties

- Increase Rate: Der Wert, mit dem die Intensität des Vignette-Effektes jeden Frame erhöht wird (default setting: 0.012)

4.3.5.3 Setup

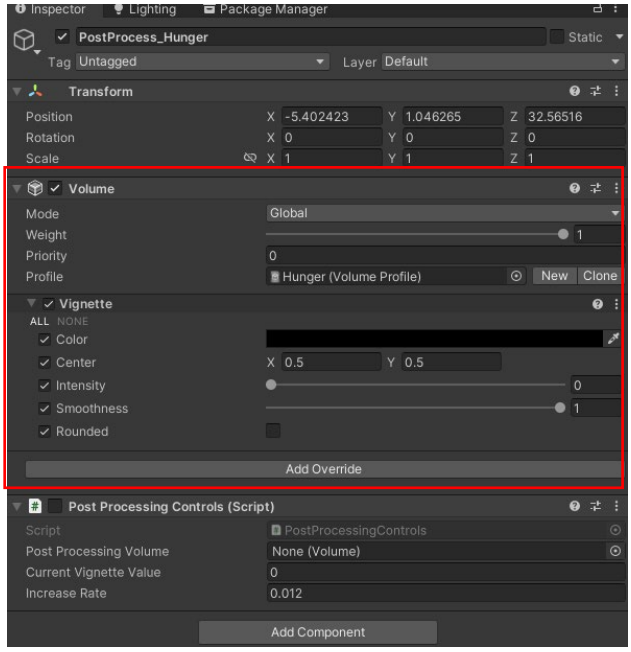
- Liegt auf: GameObject mit dem Namen „PostProcess_Hunger“

  Hier bitte beachten!

  - Es ist wichtig, das dieses Script auf diesem Objekt liegt, da im Script nach der Volume Komponente auf dem Objekt gesucht wird, auf dem das Script liegt.

- Benötigt: /

- Bitte sicherstellen, das auf dem Objekt eine Volume Komponente mit einem Vignette Effekt vorhanden ist:



4.3.6 TutorialManager.cs

→ Dieses Script ist dafür zuständig, das alle Animationen und Tutorial Texte während einer Cutscene an den gebrauchten Stellen abgespielt und aktiviert werden.

4.3.6.1 Funktionen

- Das Deaktivieren von Animations-Stücken, die erst später aktiviert werden sollen
- Das Abspielen des Tutorials nach einem kurzen Zeit-Delay (hier werden die Cinematic Bars in das Bild geschoben)
- Das Deaktivieren des RPG-Talks, wenn das Spiel angehalten wurde (sonst würden diese weiterlaufen)
- Eine Funktion, in der der Spieler in einer nächsten Animation in das Level läuft und ausgeführt wird wenn das Tutorial vorbei ist

4.3.6.2 Properties

- /

4.3.6.3 Setup

- Liegt auf: GameObject mit dem Namen „Tutorial Manager“ (in den Szenen „Cutscene_TutorialLvl1 – Narrativ“ und „Cutscene_TutorialLvl2 – Narrativ“ -> zu finden unter dem Ordner Cutscenes!)
- Benötigt: RPGTalk
- Die Komponente „RPGTalk“ auf dem GameObject „TutorialManager“ muss unter „RPGTALK“ referenziert sein

Hier bitte beachten!

- Es muss das ganze Objekt, auf dem die gesuchte Komponente liegt, in das Feld gezogen werden -> Das GameObject „Tutorial Manager“ auf das Feld ziehen!

- Das GameObject „Timeline2“ muss unter „Cutscene 1 Part 2“ referenziert sein

  Hier bitte beachten!

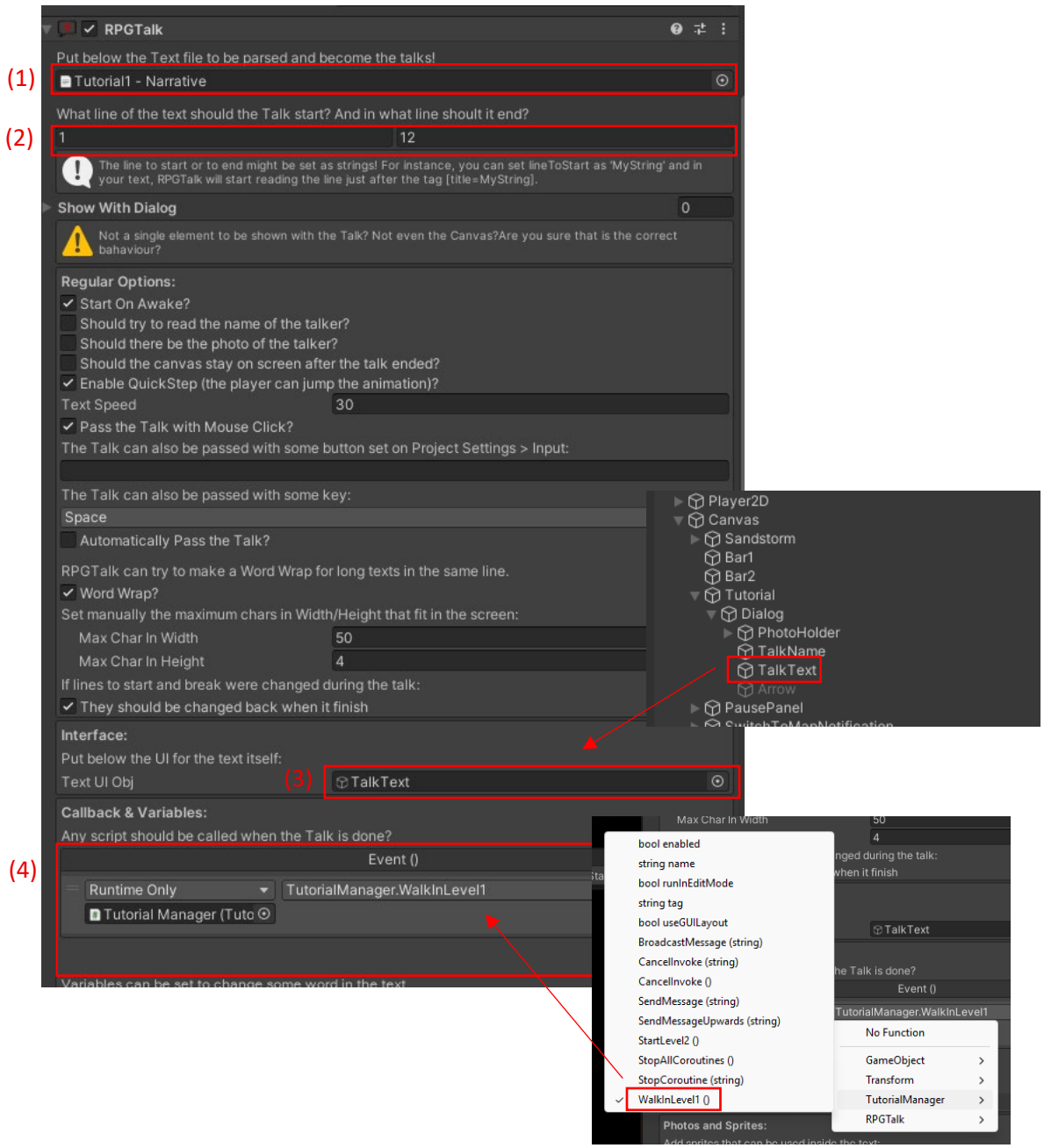
  - Das muss nur in der Szene „Cutscene_TutorialLvl1 – Narrativ“ gemacht werden, woanders existiert dieses GameObject nicht!

- Das GameObject „Tutorial“ muss unter „Tutorial“ referenziert sein

  Hier bitte beachten!

  - Dieses Objekt ist ein UI-Element und deswegen möglicherweise unter „Canvas“ eingeklappt.

- Einrichtung RPGTalk:
  1. Die Text-Datei, die im Tutorial abgespielt werden soll, referenzieren (diese sind unter Assets > Dialogs zu finden). In der Cutscene für Level1 wird „Tutorial1“ und in der Cutscene für Level 2 „Tutorial2“ verwendet.
  2. Die Anfangszeile und die Endzeile, die der Text später in der Speechbubble eintragen soll, angeben. (Level 1 -> 1-12, Level 2 -> 1-2)
  3. Das Textfeld-Objekt zuweisen, das er für das Anzeigen des Textes nutzen soll. Dieses Textfeld ist unter Canvas > Tutorial > Dialog zu finden und nennt sich „TalkText“.
  4. RPGTalk sagen, was nach dem Tutorial passieren soll. Zum hinzufügen eines Events muss auf das „+“ geklickt werden. Die folgende Verknüpfung ist wie bei einem Button aufgebaut:
     a. Cutscene Level 1: Das Script „Tutorial Manager“ von dem Objekt „Tutorial Manager“ anfügen und die Funktion „WalkInLevel1“auswählen.
     b. Cutscene Level 2: Das Script „Tutorial Manager“ von dem Objekt „Tutorial Manager“ anfügen und die Funktion „StartInLevel2“auswählen.



4.3.7 UIButtonClick.cs

→ Dieses Script ist ausschließlich dafür zuständig, den Namen eines angeklickten Objektes bzw. eines Buttons zu registrieren. Es wird in der File Collection verwendet, um das passende Image zum Namen des angeklickten Files anzuzeigen.

4.3.7.1 Funktionen

- Eine Funktion, in der der Name des angeklickten Objektes in der Szene abgefragt wird später unter dem string „ClickedButtonName“ weiterverwendet wird

4.3.7.2 Properties

- /

4.3.7.3 Setup

- Liegt auf: GameObject mit dem Namen „Logic“

  Hier bitte beachten!

  - Dieses Script ist nur in der Szene „FileCollectionScreen“ und wird auch nur dort an das „Logic“ Objekt gehangen

- Benötigt: /

4.4 SONSTIGES

4.4.1 SpriteFacingCam.cs

→ Dieses Script sorgt dafür, das sich die Pappaufsteller-Sprites in der Szene immer in Richtung der Kamera drehen, in diesem Fall nur seitlich (links-rechts). Dieser Effekt wird auch „Billboarding“ genannt.

4.4.1.1 Funktionen

- Das Abspeichern der Position der Kamera in jedem Frame
- Das Überschreiben des Y-Wertes des Objektes, auf dem das Script liegt, mit dem der Kamera
- Das Ausrichten des Sprites in Richtung der Kamera mit einer „LookAt“
- Das sicherstellen, das das Sprite auch korrekt zur Kamera schaut, indem es auf den richtigen Achsen rotiert wird

4.4.1.2 Properties

- /

4.4.1.3 Setup

- Liegt auf: alle GameObjects, die von diesem Effekt betroffen sein sollen (zB. Prefabs „Enemy“, „Boss“, Pakete in Level 1 und 2)
- Benötigt: /

4.4.2 XRayEffect.cs

→ Dieses Script ist dafür zuständig, es optisch angenehmer und ansprechender aussehen zu lassen, wenn die Kamera mal innerhalb eines Objektes kommt und die Spielfigur hinter dem Objekt steht.

Ohne Effekt – alles gleich hell, man sieht normal die Kanten des Objektes in der Luft hängen, wirkt fehlerhaft und wie ein „Glitch“ der Kamera

Mit Effekt – alles ist mehr abgedunkelt, der Sandsturm ist pausiert, der Effekt von „man schaut von dem Inneren eines Objektes auf den Spieler“ kommt besser zur Geltung

4.4.2.1 Funktionen

- Das Auffinden von dem Sandsturm-Layern zum Anfang
- Das Abfragen des Materials von einem Objekt, wenn dieses von der Kamera berührt wird und den Tag „NeedsXRay“ trägt, und „Zwischenspeichern“ des Materials, um später wieder verwenden zu können
- Das Austauschen des Materials von dem Objekt, das von der Kamera berührt wurde, mit einem Vorgefertigtem, das den Effekt wie auf dem Bild oben auslöst und das Deaktivieren des Sandsturms
- Das Zurücksetzen des Materials auf das zwischengespeicherte Ursprungsmaterial, wenn die Kamera keine Berührung mehr mit dem Objekt hat und das wieder Aktivieren des Sandsturms

4.4.2.2 Properties

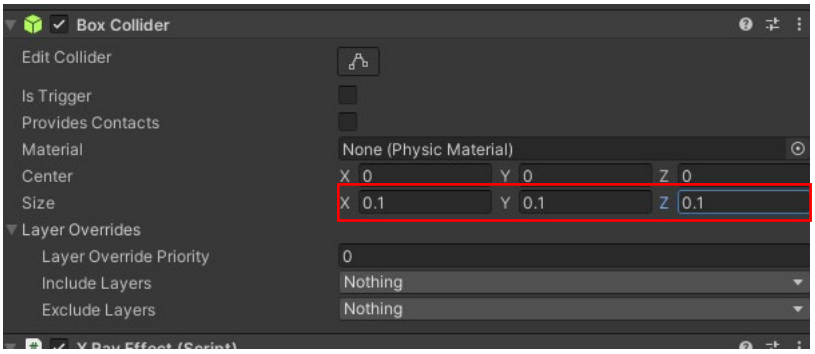
- /

4.4.2.3 Setup

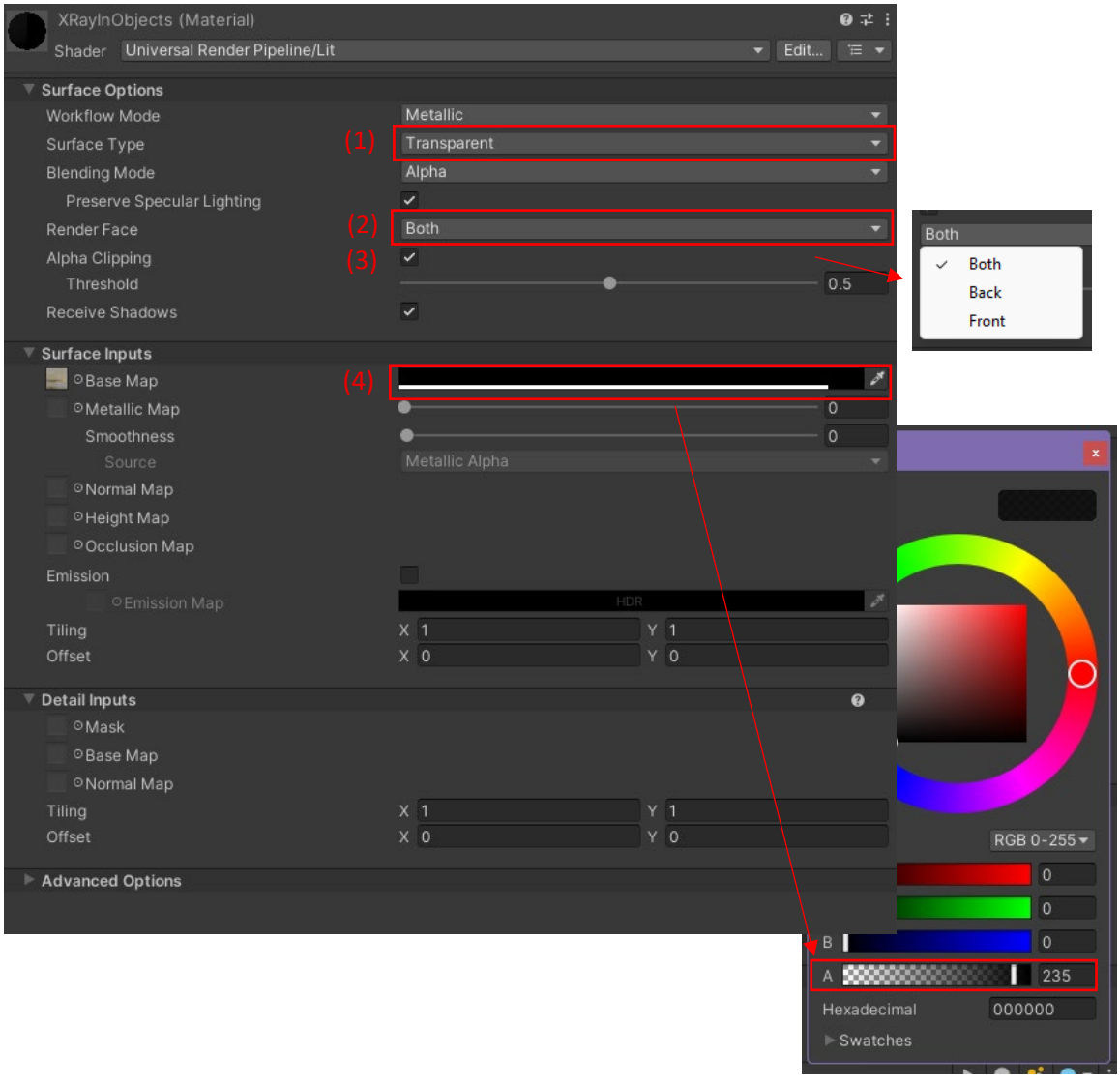
- Liegt auf: Child-GameObject „Collider“ von GameObject „Main Collider“
- Benötigt: Box Collider, Rigidbody

Hier bitte beachten!

- Der Box Collider sollte nicht größer als 0.1 auf allen Achsen sein



- Das Vorgefertigte Material:
  1. Das Surface Type muss von „Opaque“ auf „Transparent“ gewechselt werden
  2. Bei „Render Face“ muss „Both“ ausgewählt sein
  3. Der Haken bei „Alpha Clipping“ muss aktiv sein
  4. Die Transparenz der Farbe des Materials muss nach Wünschen angepasst werden (für dieses Material wird eine Transparenz von 285 benutzt (Stand: 17.06))



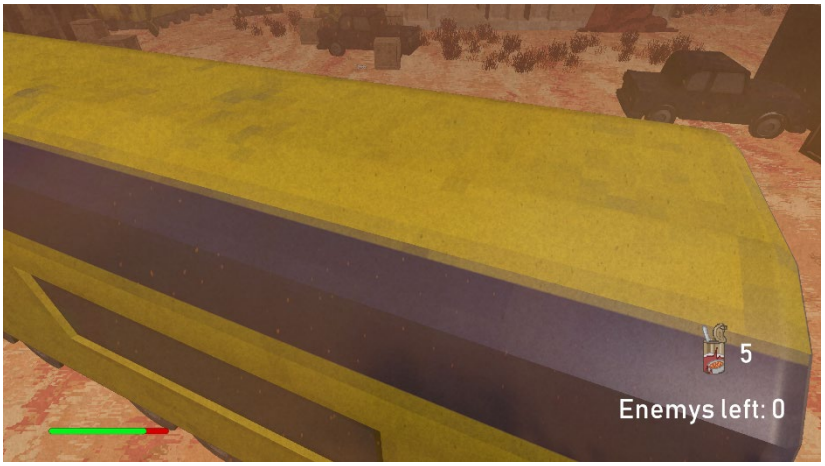
Hier bitte beachten!

- ➢ Der Name des Materials muss „XRayInObjects“ lauten und bei dem Ordner „Resources“ untergeordnet sein!

- Nochmals kontrollieren, das alle Objekte, die von diesem Effekt beeinflusst werden sollen, den Tag „NeedsXRay“ tragen

4.4.3 XRayWithRaycast.cs

→ Dieses Script ist ähnlich zu dem „XRayEffect.cs“ und soll Objekte, die die Sicht auf den Spieler und seine Umgebung versperren, leicht durchsichtig machen.



Ohne Effekt – der Spieler ist hinter dem Objekt, was vor ihm ist, nicht mehr zu sehen, was die Spielqualität einschränkt

Mit Effekt – das Objekt vor ihm wird leicht transparent gemacht, der Spieler fühlt sich nicht „blind“ und kann alles, was hinter dem Objekt passiert, weiterhin sehen ohne das das Objekt komplett „verschwindet“

4.4.3.1 Funktionen

- Das Erstellen eines Raycasts auf der Kamera in Richtung, in die die Kamera schaut
- Die Abfrage, ob ein Objekt in der Szene von dem Raycast getroffen wurde und dieses Objekt den Tag „NeedsXRay“ trägt
- Das Abfragen des Materials, welches das Objekt in Moment hat und das „Zwischenspeichern“ des Materials

- Das Auswechseln des momentanen Materials mit dem Vorgefertigtem, wodurch der Effekt, wie oben im Bild gezeigt, erzielt wird
- Das Zurücksetzen des Materials zu dem „zwischengespeicherten“ Original und den „Zwischenspeicher“ leeren, wenn das Objekt nicht mehr von dem Raycast registriert wird

4.4.3.2 Properties

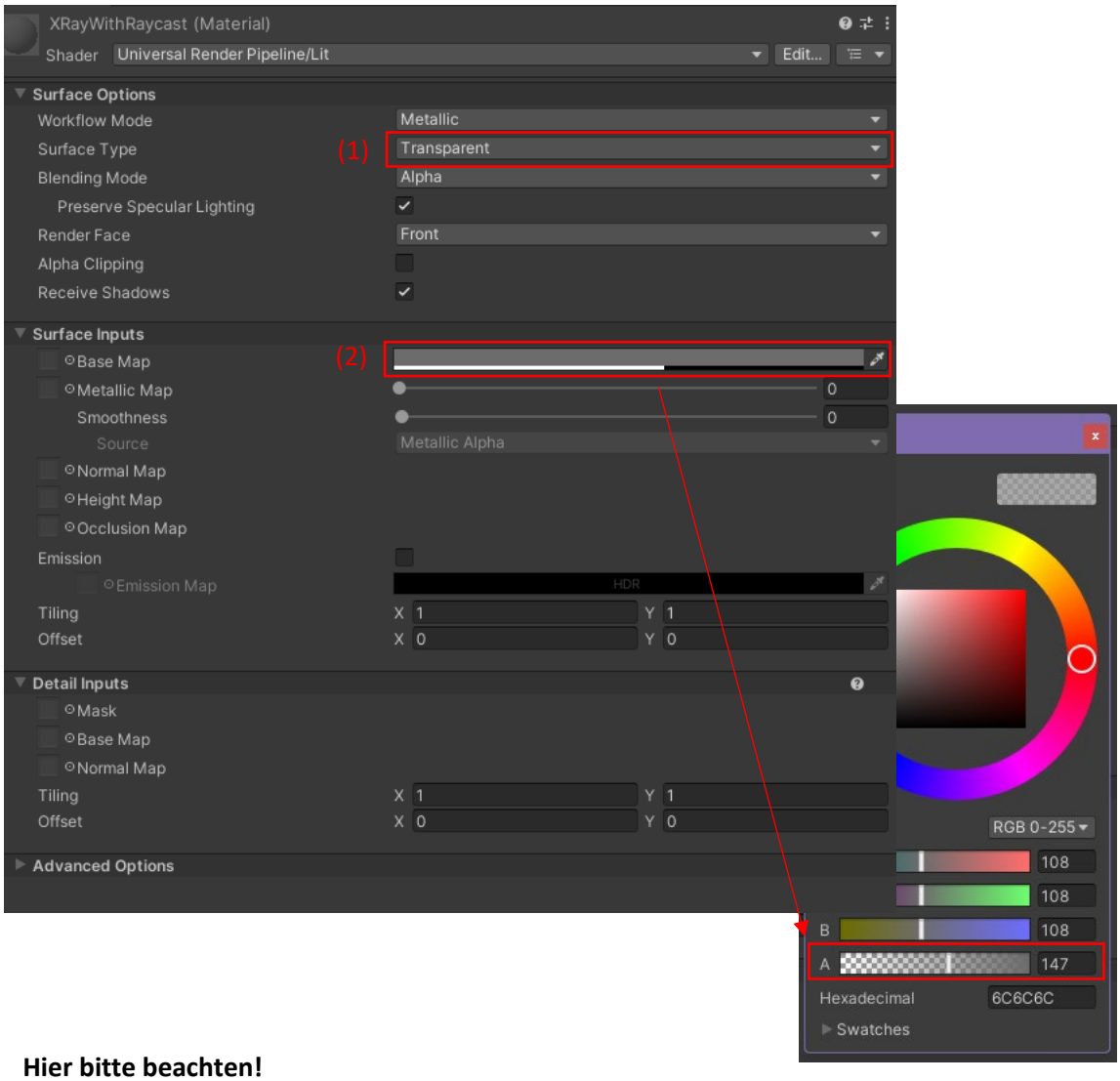
- /

4.4.3.3 Setup

- Liegt auf: GameObject mit dem Namen „Main Camera“
- Benötigt: /

- Das Vorgefertigte Material:

1. Das Surface Type muss von „Opaque“ auf „Transparent“ gewechselt werden
2. Die Transparenz der Farbe des Materials muss nach Wünschen angepasst werden (für dieses Material wird eine Transparenz von 147 benutzt (Stand: 17.06))



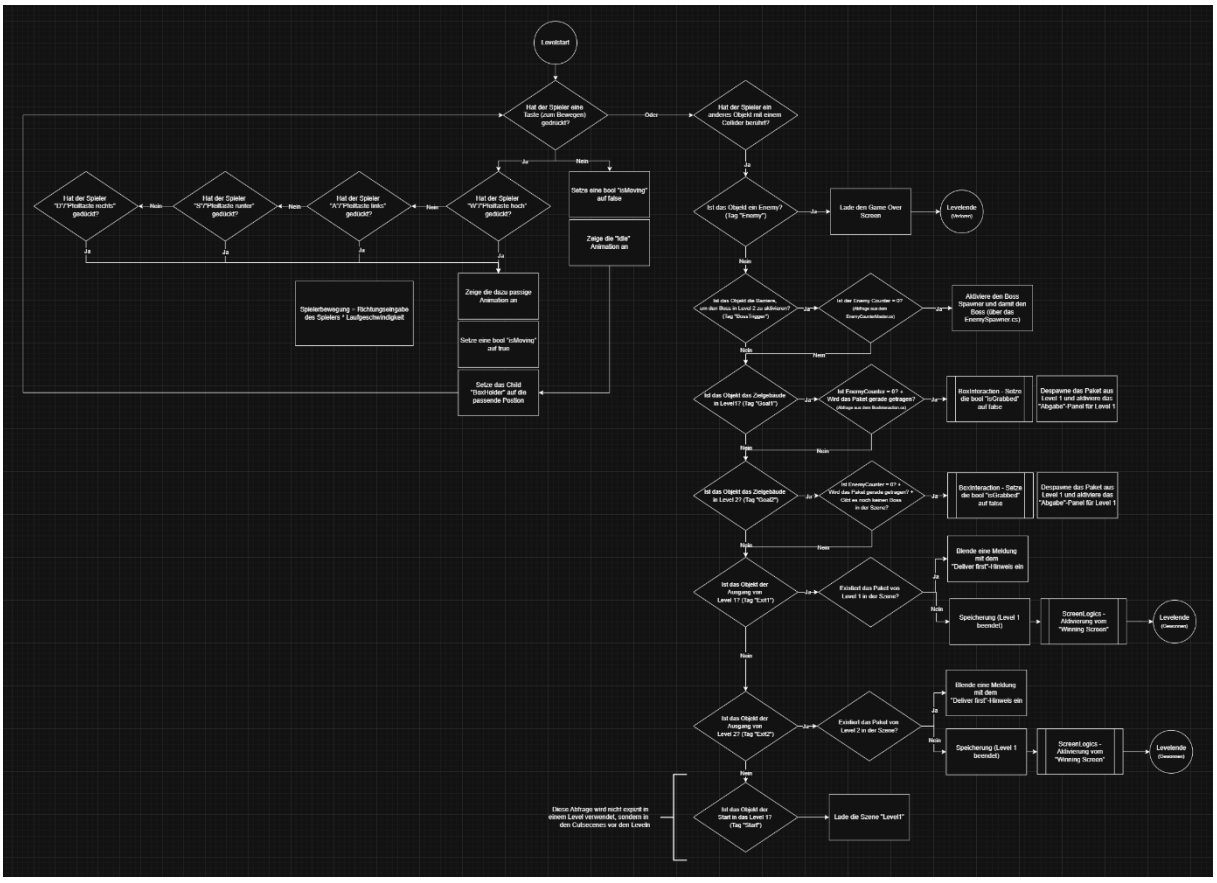
Hier bitte beachten!

➢ Der Name des Materials muss „XRayWithRaycast“ lauten und bei dem Ordner „Resources“ untergeordnet sein!

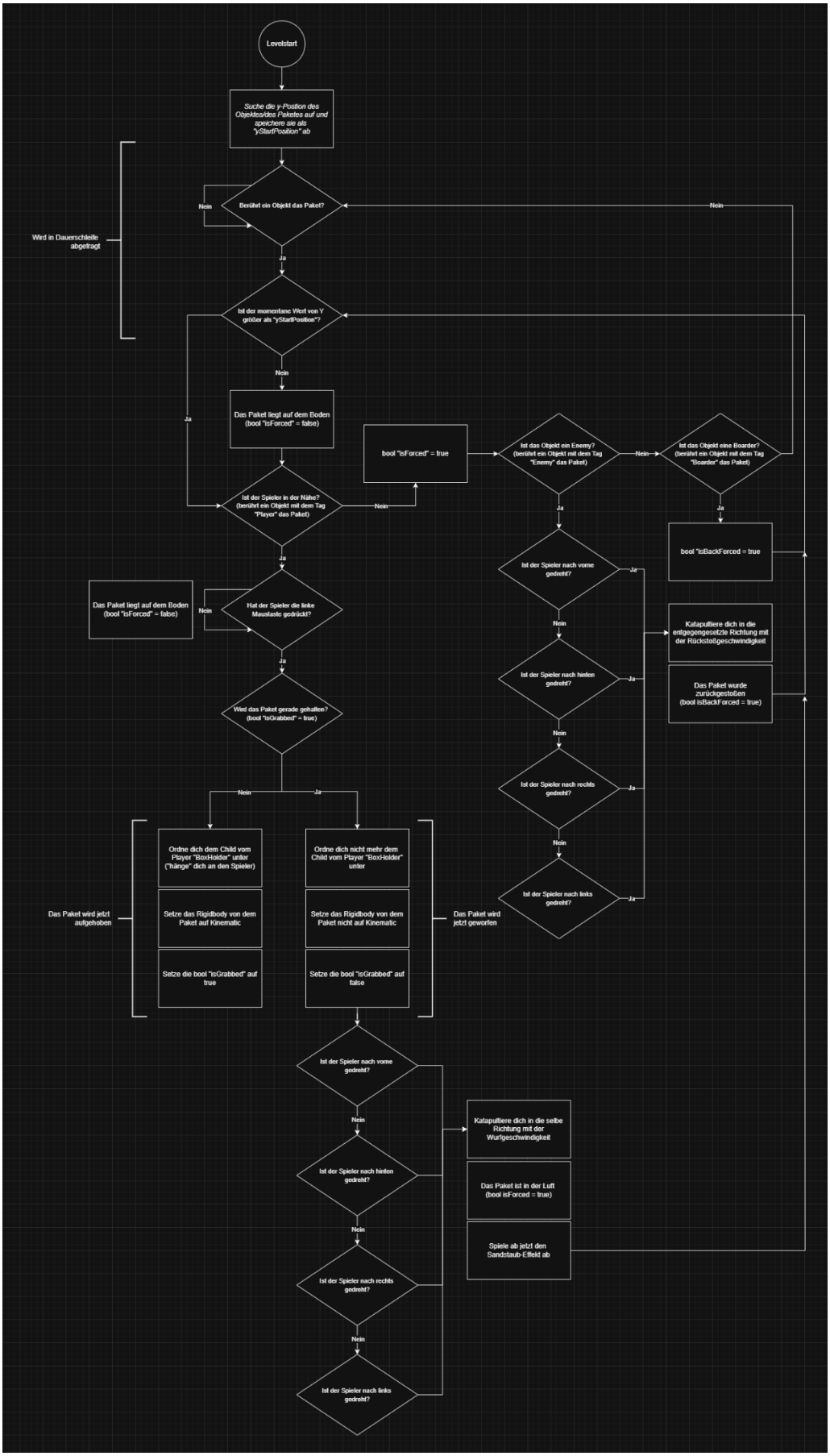
- Nochmals kontrollieren, das alle Objekte, die von diesem Effekt beeinflusst werden sollen, den Tag „NeedsXRay“ tragen

5 FLOWCHARTS

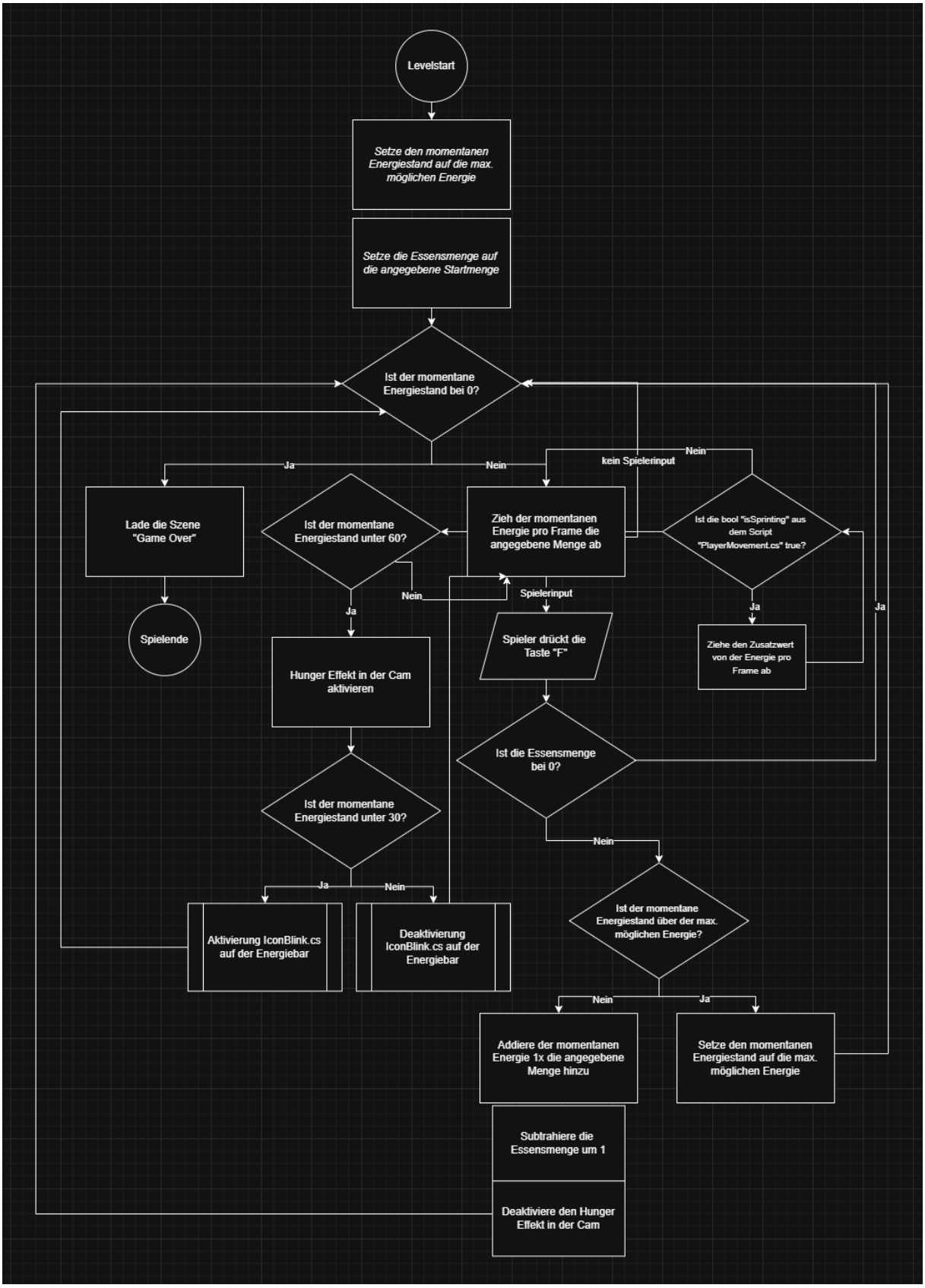
5.1 PLAYERMOVEMENT.CS



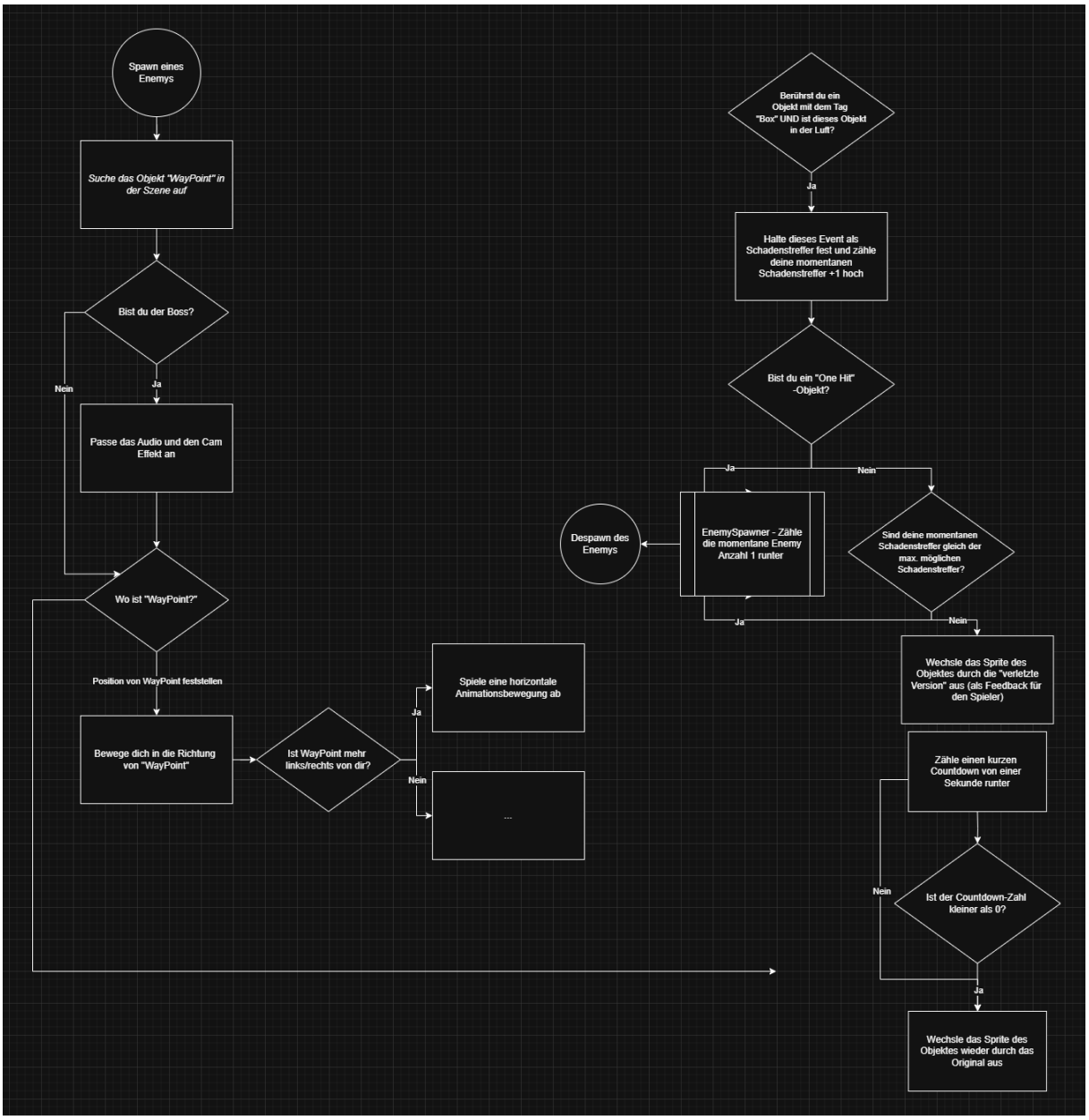
5.2 BoxInteraction.cs



5.3 ENERGYBAR.CS



5.4 ENEMY.CS


Spawn eines Enemys
Suche das Objekt "WayPoint" in der Szene auf
Bist du der Boss?
Nein
Ja
Passe das Audio und den Cam Effekt an
Wo ist "WayPoint?"
Position von WayPoint feststellen
Bewege dich in die Richtung von "WayPoint"
Ist WayPoint mehr links/rechts von dir?
Ja
Spiele eine horizontale Animationsbewegung ab
Nein
...
Berührst du ein Objekt mit dem Tag "Box" UND ist dieses Objekt in der Luft?
Ja
Halte dieses Event als Schadenstreffer fest und zähle deine momentanen Schadenstreffer +1 hoch
Bist du ein "One Hit" -Objekt?
Ja
Nein
EnemySpawner - Zähle die momentane Enemy Anzahl 1 runter
Despawn des Enemys
Sind deine momentanen Schadenstreffer gleich der max. möglichen Schadenstreffer?
Ja
Nein
Wechsle das Sprite des Objektes durch die "verletzte Version" aus (als Feedback für den Spieler)
Zähle einen kurzen Countdown von einer Sekunde runter
Ist der Countdown-Zahl kleiner als 0?
Nein
Ja
Wechsle das Sprite des Objektes wieder durch das Original aus