TDD und Hybrid Document

Knucklebones Remake mit Erweiterung

(basierend auf dem Minigame aus “Cult of the Lamb”)

Semesterabschlussprojekt 5

GDS23 – Macromedia Leipzig, von Linnea Hoppe

Letzter Eintrag: 25.01.2025

INHALT

1 VORWORT

1.1 WAS IST DIESES DOKUMENT?

Die hier festgehaltenen Informationen dienen der Dokumentation und Prozesserklärung meines Einzelprojektes für das 5. Semester meiner Ausbildung. Folgende Thematik wählte ich als Prüfung aus:

> „Ziel meines Semesterabschlussprojekts ist die Erweiterung des Minispiels „Knucklebones“ aus „Cult of the Lamb“. Dabei soll zunächst das Kernsystem des Originals funktional rekonstruiert werden, inklusive Würfelmechanik, Spielfeldlogik und Punkteberechnung. Den KI-Gegner ersetze ich durch einen lokalen Multiplayer-Modus.
>
> Anschließend soll das Spiel durch neue Mechaniken gezielt erweitert und strategisch vertieft werden. Darunter zählen verschiedene Spezialwürfel und einsetzbare Aktionen, von denen ich jeweils mind. 2 zusätzlich sinnvoll einbauen möchte.
>
> Ziel ist es, die einfache Grundidee in ein eigenständiges, taktisch anspruchsvolles Würfelspiel mit ca. 15 Minuten Spielzeit zu transformieren. Technisch liegt der Fokus auf einer sauberen Umsetzung in Unity (C#), mit klarer Struktur und funktionalem Whitebox-Design.“

Meine Spezialisierung liegt in der Game Programmierung, weshalb der Hauptteil in diesem Dokument als TDD (Technical Design Document) festgehalten wird. Alle restlichen hier eingetragen Informationen dienen nur der vollständiger Halber, da ein kleiner Teil des Projektes auch GDD-relevant und für das Verständnis wichtig zu wissen und zu verstehen ist.

1.2 TECHNISCHE ECKDATEN

Unity Version:

Code: Visual Studio & C#

1.3 PLAYTESTING DOCUMENTATION

Sowohl die Ergebnisse von Testungen von mir als auch von Externen Playtestern wurden während des Prozesses grob festgehalten und später in eigenständige Aufgaben im Taskboard auf meinem Teams Kanal dokumentiert und bearbeitet. Diese Aufgaben tragen den Namen „Bugfixing und Anpassungen“ mit einer abgekürzten Zahlenfolge danach, die das Datum darstellen (zB. Test am 17.11.2025 -> 1711)

2 Game Design Document

2.1 Overview Knucklebones Original

2.1.1 Eckdaten

Genre: Würfel Spiel, Glücks- und Strategiespiel, Minispiel

Setting: düster-mystische Fantasiewelt (spielt in „Cult of the Lamb“)

Art Style: 2D-Comicstil mit düster-niedlicher Ästhetik, handgezeichnete Grafik (-> typisch für das Originalspiel)

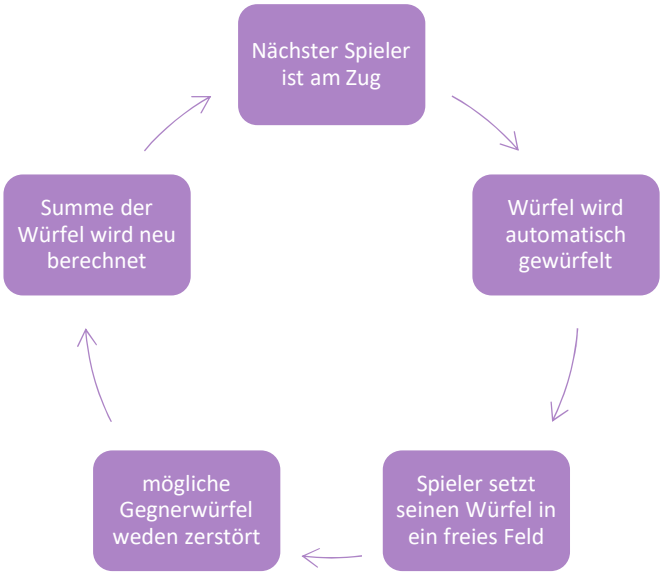
USK: 12 (abgeleitet vom Hauptspiel)

Spielmodi: ursprünglich Singleplayer mit KI-Gegner, neueres Update ermöglicht auch Coop-Gameplay

Plattform: PC, Controller (Playstation, Xbox, Nintendo)

Ansicht: Top Down (Draufsicht auf das Spielfeld)

2.1.2 Core Loop



2.1.3 Features

- Hoher Zufallsfaktor
- Mischung aus Glück und strategisches Platzieren
- (In Cult of the Lamb) mehrere Gegner mit steigendem Schwierigkeitsgrad
- (In Cult of the Lamb) Goldgewinne bei Runden-Sieg
- Im typischen Cult oft he Lamb-Stil (niedlich-makaber, mit handgezeichneter Grafik und humorvoller Stimmung)
- Schnelle Spielrunden
- Würfel schieben sich in Richtung Mitte auf
- Jede Spalte hat seinen eigenen summierten Wert der darauf liegenden Würfel

2.1.4 Key Features

- Rundenbasiertes Würfel platzieren
- Punktesystem (der Spieler mit den meisten Punkten gewinnt)
- 3 Spalten mit jeweils 3 Feldern zum Platzieren auf jeder Spielerseite
- Gleiche Würfelwerte verstärken sich gegenseitig (mehr Punkte in einer Spalte)
    - Doppelte Würfel = (x + x) * 2
    - Dreifache Würfel = (x + x + x) * 3
- Gegnerwürfel können mit gleichem Würfelwert zerstört werden
- Spielende wenn alle Felder voll auf einer Seite

2.1.5 USP

Ein taktisches, süchtig machendes Würfelspiel im niedlich-okkulten Stil, das Glück, Strategie und schwarze Magie charmant miteinander verbindet.

2.2 NEW GAMEPLAY

2.2.1 Feste Komponenten für das Rework

- KI-Gegner wird durch einen local Coop-Modus ersetzt
- Plattform: (vorerst) Controller (Xbox)

2.2.2 Neue Spiel Elemente

Alle hier aufgeführten „neuen Spielelemente“ sind im Spiel als „Ability Cards“ dargestellt, also Karten in einem Deck, die dann die folgenden Fähigkeiten ausführen:

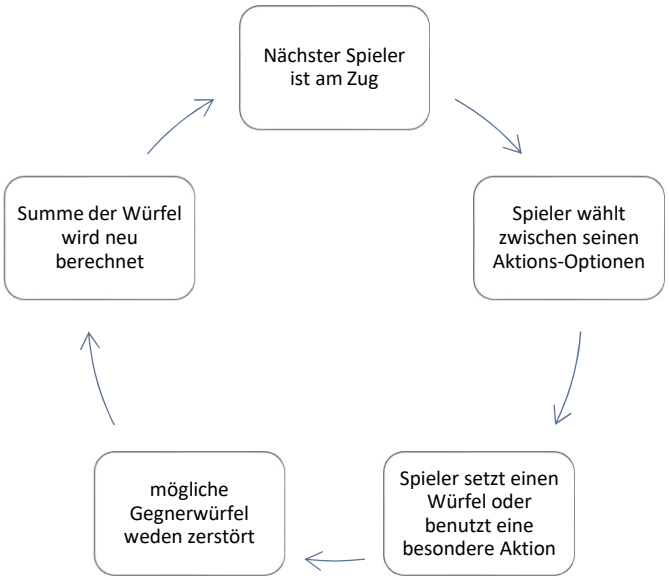
2.2.2.1 Spezialwürfel

- "Der Schicksalswürfel"
  Der Spieler kann auswählen, dass der nächste Würfel nur gerade oder nur ungerade Zahlen zeigen darf.
    - Wird der Würfel ausgewählt, öffnet sich ein kleineres Fenster wo der Spieler die Wahl zwischen geraden Zahlen oder ungeraden Zahlen hat. Sobald eine Entscheidung getroffen wurde, wird "normal gewürfelt, aber mit den ausgewählten Zahlen
- "Der Langfinger"
  Man wählt einen gegnerischen Würfel, dieser wird dann auf die eigene gegenüberliegende Reihe verschoben.
    - Wenn man diesen Würfel auswählt, kann der Spieler einen Würfel des gegnerischen Feldes auswählen. Sobald einer angeklickt wurde, wird ein gleichwertiger Würfel auf die gegenüberliegende Reihe (vom aktiven Spieler) platziert. Damit wird der Würfel des Gegners gelöscht.

2.2.2.2 Aktionen

- "Das Schild"
  Eine ausgewählte Würfelreihe kann 3 Runden lang nicht vom Gegner beeinflusst werden.
  - Wenn diese Aktion ausgewählt wird, kann der Spieler eine von seinen drei Reihen auswählen. Nach dem Anklicken wird diese Reihe symbolisch als "geschützt" markiert und eine kleine "Healthbar" unterhalb angezeigt. Die Healthbar ist beim Setzen der Aktion voll und verliert nach einem gegnerischen Zug immer etwas Health (geplant: 1/3). Solange das Schild aktiv ist, kann der Gegner die Reihe nicht beeinflussen. Ist die Healthbar auf 0 verschwindet die "Schutzmarkierung" und der Effekt.
- „Der Twister“
  Tauscht die Position zwei eigener ausgewählter Würfel.
  - Wenn diese Aktion ausgewählt wird, kann der Spieler zwei Würfel auf seinem Feld auswählen. Sobald zwei Würfel angewählt wurden, werden die Werte der Würfel miteinander getauscht, was den Effekt eines Würfeltausches impliziert. Später kann dieser Effekt auch visuell durch einen richtigen "Würfeltausch" erfolgen.

2.2.3 Core Loop



2.2.4 Eingebaute NTHs (Nice To Haves)

2.2.4.1 Mechaniken

- "Diagonale Bonus"
  Rechnet einen extra Bonus auf die Gesamtpunktzahl eines Spielers.
  - Hat der Spieler eine Diagonale mit gleichwertigen Würfeln auf seinem Spielfeld gebildet wird ein bestimmter Bonus für den Spieler berechnet, der (nur!) neben der Gesamtpunktzahl angezeigt wird. Die Rechnung funktioniert hierbei wie bei den bereits im Original-Spiel integrierten Punkteerhöhungs-Szenarien. Da sie aber wegen ihrer Schwierigkeit stärker belohnend sein soll, ist die Rechnung folgende: (x + x + x) * 4.
- „Exodia"
  Lässt einen Spieler sofortig gewinnen, sobald eine X-förmige Würfel-Anordnung gebildet wurde.
  - Schafft der Spieler gleich zwei Diagonalen in seinem Feld zu bilden und damit eine X-förmige Würfel-Anordnung zu legen, hat dieser sofort gewonnen. Diese Mechanik ist angelehnt an das bekannte Spiel „Yu-Gi-Oh!", wo es ebenfalls einen Effekt namens Exodia existiert und einen Spieler sofort gewinnen lässt. Wegen ihrer großen Schwierigkeit und der hohen Gefahr, vom Gegner zerstört zu werden, ist die Wahrscheinlichkeit zwar ziemlich gering, wird aber dafür umso mehr belohnt.

2.2.4.2 Visuals

Das geplante Whitebox wurde optisch mit neuen Sprites und Effekten aufgewertet. Das Thema des Designs: Japanische Neon Ski-Fi Stadt im Retro Pixel Art Style. Bei der Umsetzung achtete ich vor Allem auf Neon-farbene Highlights und Pixel-Grafiken. Effekte wurden ebenfalls in Pixel-Optik gestaltet.

- Camera Effects und Post Processing mit Global Volumes
- Auswechseln der relevantesten Whitebox Elemente durch optisch ansprechendere Sprites
- Standard Schrift Art durch eine Pixel Font ersetzt
- Effekte für UI, Würfel-Kombis und Abilitys
  - Würfelfeld -> leichter sanfter Particle Effect (für den aktiven Spieler)
  - Hintergrund -> Pulsierender Effekt mit Farbregler
  - Triple -> Pixel Fire auf der Reihe
  - Dioagonale -> Blink Effect auf den betroffenen Feldern
  - Exodia -> Rainbow Effect auf den betroffen Feldern
  - Schild -> Pulsierender Particle Effect
- Bewegendes UI
  - Karten hochziehen, sobald sie angewählt werden
  - Karten umdrehen, wenn sie nicht verwendet dürfen
  - Würfel Shake, was das Würfeln darstellen soll

2.2.5 Entstehungsphase

2.2.5.1 13.10.25

Ideensammlung für neue Spielmechaniken mit Fokus auf Strategie und Variation.

Ergebnisse:

2.2.5.2 14.10.25

Testen der neuen Spielideen auf einem „Papierprototypen“ auf Miro. Jeder Spieler hat 3 Spezialwürfel und 3 Aktionen zur Verfügung, sie sind vor einem Würfelwurf zu verwenden. Werden sie benutzt, sind sie verbraucht.

Ergebnisse:

- „Der Verdoppler“ noch sehr undurchdacht, benötigt mehr Testings und Überlegung
- „Der Manipulator“ schnell geeinigt auf die zweite Variante (Wahl zwischen geraden/ungeraden Zahlen im nächsten Wurf)
- „Der Spiegel“ ist sehr stark
- „Der Joker“ ist zu undurchdacht, schnell verworfen
- „Das Schild“ sollte über mehrere Runden effektiv sein, getestet mit 3 Runden Effekt
- „Der Twister“ ebenfalls sehr stark
- „Bodenfalle“ hat wenig Effekt und ist zum Teil sinnlos, verworfen
- „Neblige Sicht“ ist sehr glücksbasiert, möglicherweise Frustration

Test 2, wo die Spezialwürfel und Aktionen mit verdienten Punkten freigekauft werden können. Wie bei Test 1 hat man die Auswahl zwischen Spezialwürfeln und Aktionen. Wird ein Effekt gekauft, wird der Zahlungsbetrag von der Gesamtpunktzahl des Spielers abgezogen. Effekte können unendlich oft gekauft werden.

Ergebnisse:

- Kaum eine Gewinnchance, wenn man etwas versucht zu kaufen, stattdessen man neigt man sogar eher von den Effekten ab
- Schlechte Design Choice, da damit ein relevantes Spielelement (das Punktesystem zum Gewinnen) zerstört wurde
- Idee verworfen

2.2.5.3 16.10.25

Weiteres Testen der übrigen neuen Spielmechaniken, nach dem System von Test 1. Außerdem ausbauen und genaueres Formulieren der Effekte.

Ergebnisse:

- „Der Verdoppler" gliederte sich nach längerem Brainstorming in 3 verschiedene Würfeleffekte
    - „Der Verdoppler" – Man erhält den gewürfelten Würfel doppelt.
    - „Der Zwillingswürfel" – Man würfelt 2 Würfel, die aber beide den gleichen Wert erhalten werden.
    - „Der Multiplikator" – Der Würfelwert des Würfels wird verdoppelt.
- „Der Spiegel" änderte sich zu „Der Langfinger", da diese Bezeichnung wegen der Würfelentfernungs-Mechanik mehr Sinn ergab - Man wählt einen gegnerischen Würfel, dieser wird dann auf die eigene gegenüberliegende Reihe verschoben.
- Diagonale Zahlenreihe bringt ebenfalls Punkte (Effekt wie doppelte/dreifache Zahlen)
- Kreuzförmige Zahlenkombi (doppelte Diagonale) führt zum direkten Sieg! (Anlehnung an Exodia aus Yu-Gi-Oh)
- Festlegung der Effekte, die planmäßig auf jeden Fall umgesetzt werden sollen: „Der Schicksalswürfel", „Der Langfinger", „Das Schild", „Der Twister"

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.
- Wird genutzt zB. zum Initialisieren eines neuen Schildes

3.1.2 Hierarchie Verständnis

- Parent -> Ein Parent ist ein übergeordnetes GameObject in der Hierarchie.
- Child -> Ein Child ist ein GameObject, untergeordnet unter einem Parent-Object

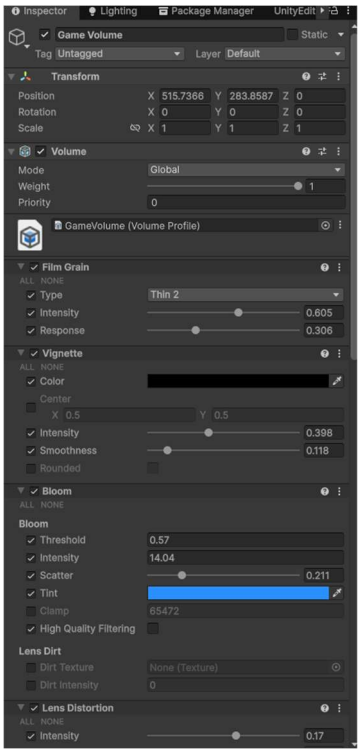
3.1.3 Global Volume und Post Processing

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

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

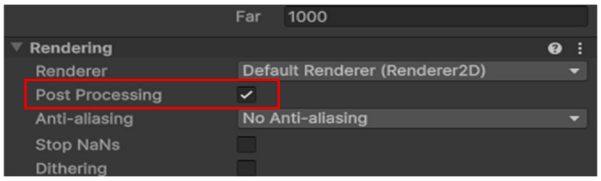
3.1.3.1 Post Processing

- 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.
- WICHTIG! Um die eingestellten Effekte auf einem Global Volume auch auf der Kamera sehen zu können, muss auf dieser das Häkchen bei „Post Processing“ gesetzt sein!

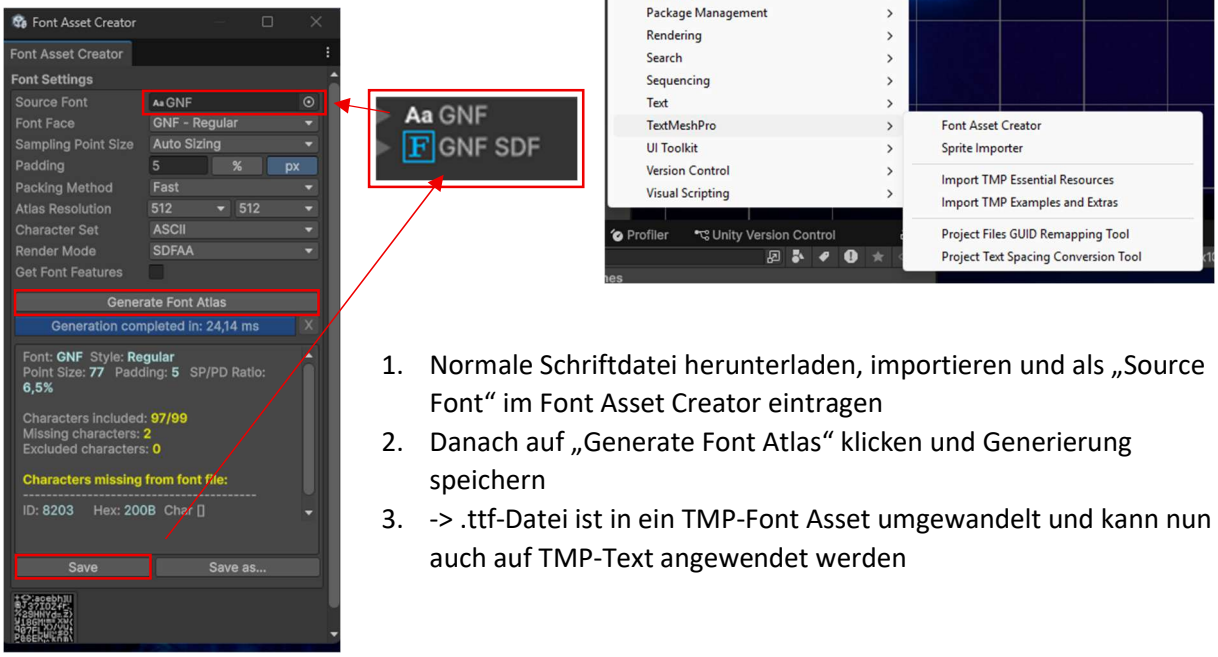


3.1.4 Text Mesh Pro

→ TMP (TestMesh Pro) ist das moderne Text-System von Unity, was im Gegensatz zum alten „UI Text“ deutlich leistungsfähiger und besser funktioniert. Es bietet mehr Qualität, Schärfe und Auflösung, aber auch mehr Styling Möglichkeiten.

3.1.5 Font Asset Creator

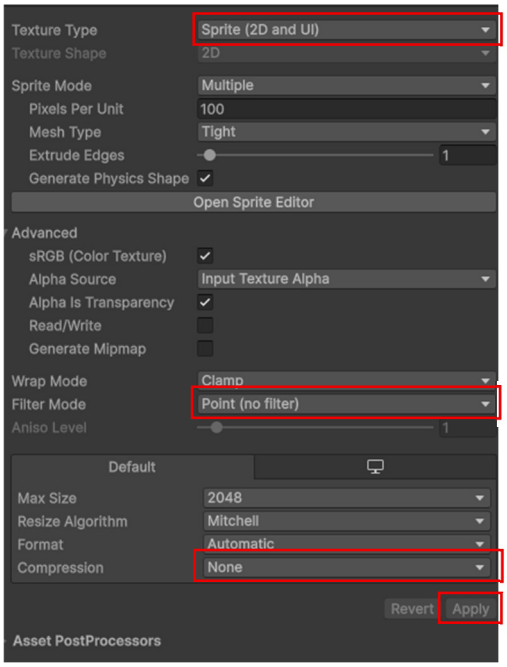
- Der „Font Asset Creator“ wandelt normale Schriftart (z. B. .ttf oder .otf) in ein TextMesh-Pro-Font-Asset um, das TMP im Spiel effizient verwenden kann
- Wichtig hierbei ist, den Font Asset Creator für „TextMeshPro“ zu verwenden!



1. Normale Schriftdatei herunterladen, importieren und als „Source Font“ im Font Asset Creator eintragen
2. Danach auf „Generate Font Atlas“ klicken und Generierung speichern
3. -> .ttf-Datei ist in ein TMP-Font Asset umgewandelt und kann nun auch auf TMP-Text angewendet werden

3.1.6 Einrichten von Sprites

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



3.1.7 KI-generierte Images in Photoshop anpassen

Alle Karten wurden mit dem Image-Generator Tool von ChatGPT erstellt. Um sie passig für die Optik des Games zu machen, habe ich folgende Schritte durchgeführt:

1. Alle Karten-Bilder an die Auflösung des Karten Rahmes angepasst (121x84 Pixel)
2. Entstehende Unschärfe mit dem Scharfzeichnugsfilter verbessert, bis die Pixel wieder scharf genug zu erkennen sind

3. Danach fehlerhafte Berechnungen von vereinzelnden Pixeln durch die Verschärfung händisch umgefärbt
4. Zuletzt das Image noch optisch teilweise angepasst und verändert, damit es „richtig“ aussieht
    a. Schicksalswürfel: Die Augenzahlen auf den Würfeln wurden nicht wie gewollt in gerade und ungerade Zahlen getrennt, aus diesem Grund musste ich diese händisch anpassen
    b. Langfinger Würfel: Der Dieb auf dem generierten Bild hatte unpassende Utensilien bei sich, aus diesem Grund habe ich nur seinen Kopf herausgeschnitten und den blauen Magie-Effekt händisch (mit dem Stempel-Pinsel) erweitert

3.2 SCRIPT

3.2.1 Basics

3.2.1.1 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

3.2.1.2 Void(), Start(), Update():

- Eine Void ist wie ein eigener abgekapselter Teil eines Scripts, das einen bestimmten Ablauf oder eine Funktion beschreibt, die dann an passenden Stellen im selben oder in externen Scripts aufgerufen werden kann und dann ausgeführt wird.
- Die Start() ist eine Void, die einmal zu Beginn aufgerufen wird. Hier bietet es sich zB. an, Werte auf ihren gewollten Ursprungswert zurückzusetzen oder gebrauchte GameObjects zu laden oder zu suchen.
- Die Update() ist eine Void, die zu jedem Frame aufgerufen wird. Hier bietet es sich zB. an, einen laufenden Timer zu managen.

3.2.2 Festgelegte Game Modes

→ „Normal“ – Standard Rundenablauf, ist automatisch am Anfang eingestellt
→ „Destiny“, „Thief“, „Shield“, „Twister“ – verhilft dem System im Hintergrund zu verstehen, welche Art von Runde der aktive Spieler spielt und managt mit dem Wissen dann das UI und Spielmechaniken, wird jeweils aktiv gesetzt sobald die jeweilige Ability Card angeklickt wurde

3.2.3 Benennung im Script

→ Ability Cards werden im Script wie folgt genannt:
- Schicksalswürfel -> Destiny
- „Langfingerwürfel“ -> Thief
- Schild Effekt -> Shield
- Twister Effekt -> Twister

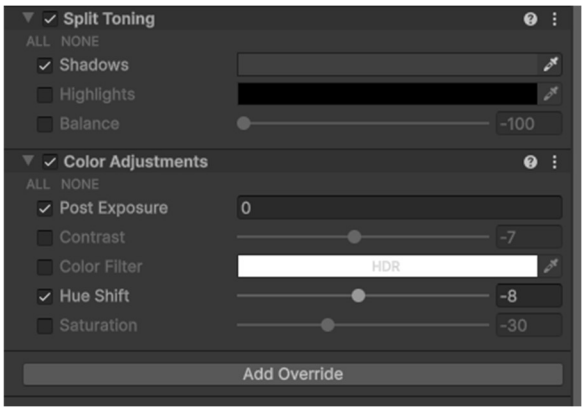
4 GAME SETUP (ONLY FUNCTIONAL RELEVANT)

4.1 ALLGEMEIN

4.1.1 Volume Settings

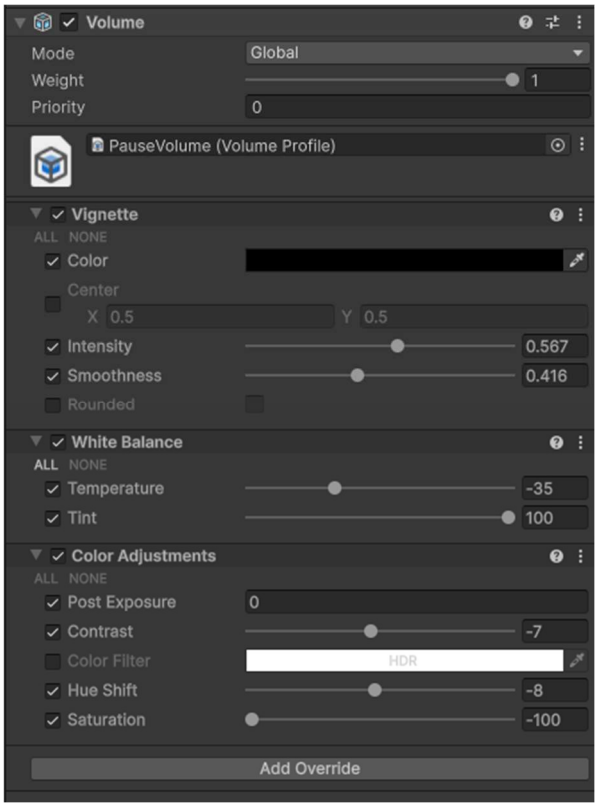
4.1.1.1 „Game Volume“

Dieses Volume ist für den Screen Effekt des Spiels verantwortlich. Durch ihn erscheinen Schrift, Game UI und Effekte Neon leuchtend, was gut zum Gesamterscheinungsbild und zum Vibe des Spiels passt.



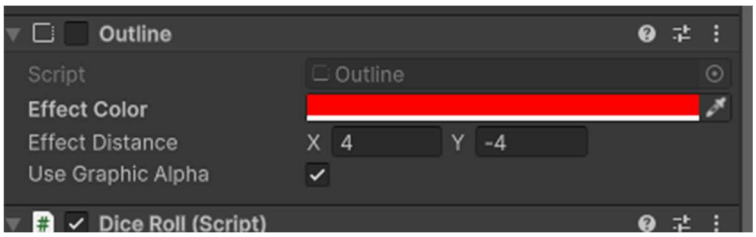
4.1.1.2 („Pause Volume“)

Dieses Volume ist für den gräulichen Effekt beim pausieren des Spiel verantwortlich. Intention, ein Volume nur für diesen Zweck zu bauen war, einen klareren visuellen Unterschied gegenüber dem farbenfrohen Game zu bekommen.



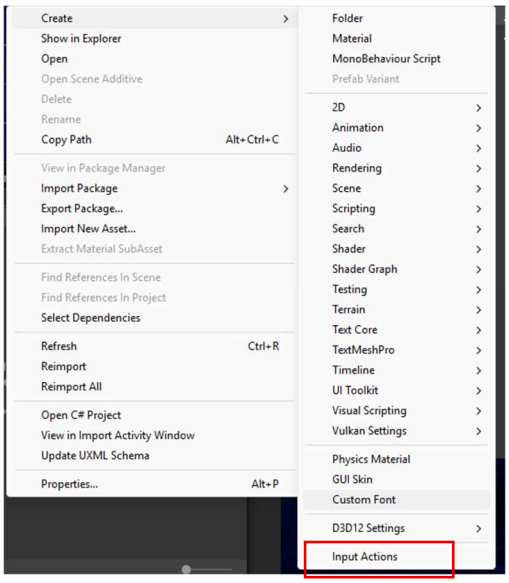
4.1.2 Outline Component

Diese Komponente muss auf jedes UI Element, was interagierbar sein soll. Die Einstellungen sind bei jedem gleich und sollen so eingerichtet werden:

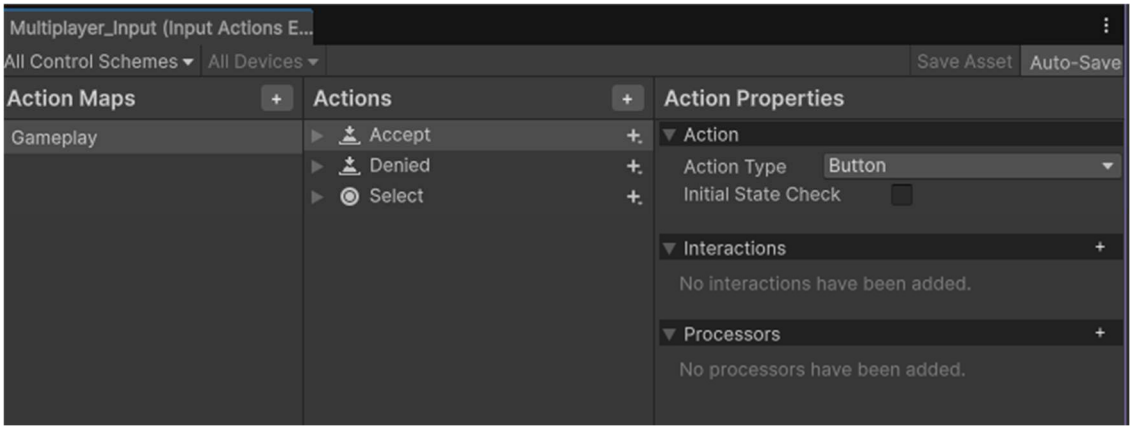


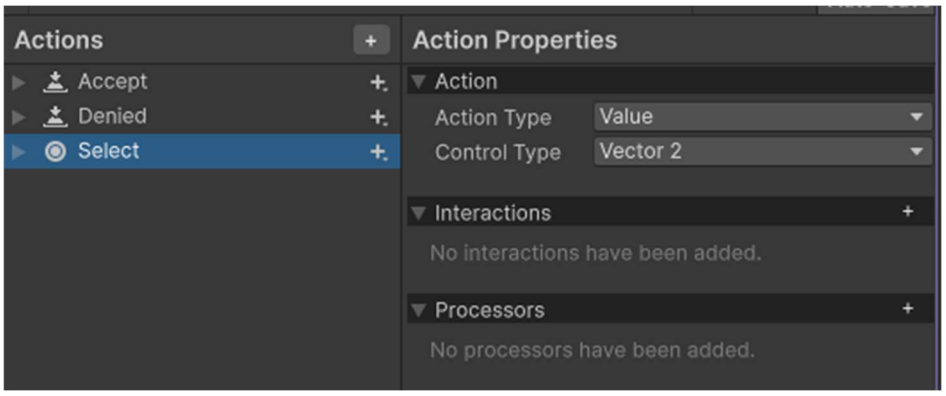
4.1.3 Controller Input Action

„Input Actions“ dienen als Leitfaden für Input Systeme wie Controller oder Keyboard, und wie sie bei bestimmten Eingaben reagieren sollen. Das in diesem Projekt verwendete und eingerichtete Input Action System nennt sich „Multiplayer_Input“.



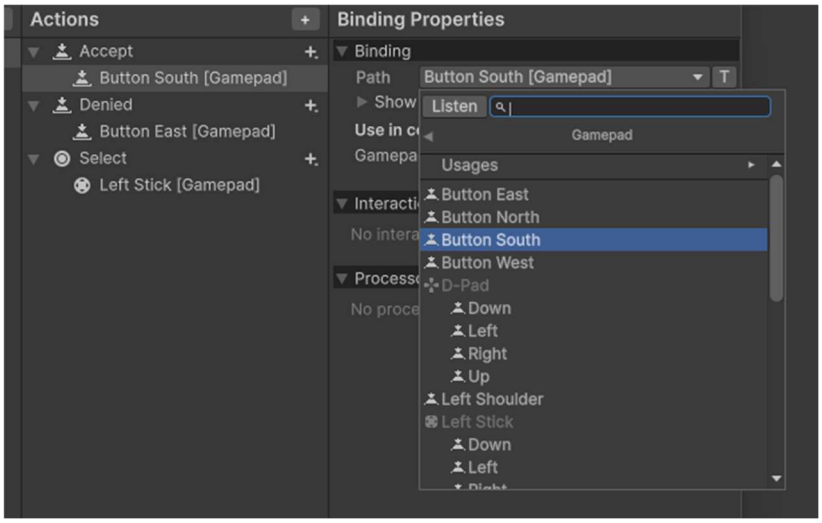
→ Ein neues Input Action wird über Create > Input Action erstellt





1. Neue Action Map erstellen (auf das Plus) -> „Gameplay“
2. 3 neue Actions erstellen -> „Accept“, „Denied“, „Select“
    a. Accept ist zum Bestätigen und als Action Type ein „Button“

b. Denied ist zum Schließen oder Zurückwählen und als Action Type ebenfalls ein „Button“
c. Select ist zum einfachen Anwählen/Hovern über Buttons und deswegen vom Action Type eine „Value“, und wegen einer möglichen Bewegung auf 2 Achsen vom Control Type „Vector 2“



3. Jede erstellte Action mit einem Binding genauer Definieren
    a. Accept wird auf dem Gamepad mit dem „South Button“ bestätigt
    b. Denied wird auf dem Controller mit dem „East Button“ bestätigt
    c. Select wird auf dem Controller mit dem „Left Stick“ gesteuert

Nun können die Controller Eingaben im Spiel gemanagt werden, u.a. über das Script „Input Manager.cs“.

4.2 „Home Screen“

4.2.1 Buttons/Interactables

- Start Game (Startet das Spiel)
- Quit Game (Öffnet ein Fenster, wo man das Beenden des Spiels bestätigen muss)
- (Tutorial (Öffnet ein Panel, das die Spielregeln erklärt))

4.2.2 Canvas Einstellungen

4.2.2.1 Canvas 1

Name: „Canvas_NoBloom“

Einstellung: benötigt keine Extra Einstellungen, Original-Canvas Settings übernehmen

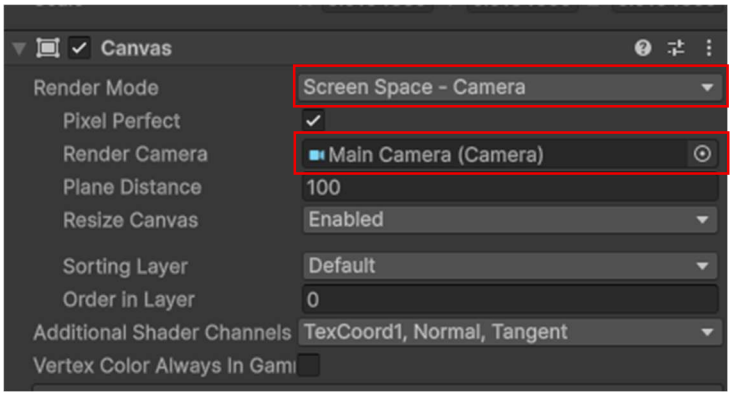
Darunterliegende Elemente: Hintergrund

Funktion: Dieses Canvas dient ausschließlich UI Elementen, die nicht von dem Global Volume Effekt beeinflusst werden sollen.

4.2.2.2 Canvas 2

Name: „Canvas“ (Haupt-Canvas)

Einstellungen: Render Mode muss auf „Screen Space – Camera“ gestellt sein, Render Camera muss mit der Szenen Kamera („Main Camera“) gefüllt werden



Außerdem bitte daran denken, den Haken bei „Post Processing“ auf der Kamera zu setzen!
(Siehe Coding und Engine Lexika – Global Volume)

Darunterliegende Elemente: alles andere

Funktion: Dieses Canvas dient ausschließlich UI Elementen, die von dem Global Volume Effekt beeinflusst werden sollen.

4.3 „Game Screen“

4.3.1 Buttons/Interactables

- Auswählen und Bestätigen ([A] / [X])
  - Würfel (Würfelt eine beliebige Zahl)
  - Fähigkeits-Karten (Ändern den Game Mode und lösen ihre Ability aus)
  - Reihen (Setzen einen Würfel oder wählen die darin liegenden Würfel aus)
  - Destiny-Dice Auswahl (zum Entscheiden, welche Art Zahl gewürfelt werden soll)
- Schließen oder Zurück ([B] / [O])
  - Öffnet im Spiel automatisch ein Fenster zum Beenden des Spiels

4.3.2 Canvas Einstellungen

4.3.2.1 Canvas 1

Name: „Canvas_NoBloom“

Einstellung: benötigt keine Extra Einstellungen, Original-Canvas Settings übernehmen

Darunterliegende Elemente: Hintergrund, Spieler Sprites

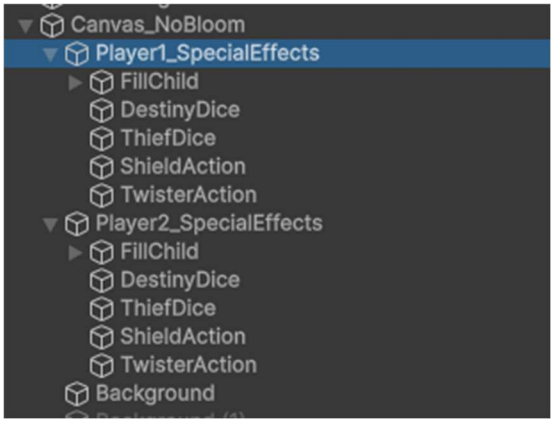
Funktion: Dieses Canvas dient ausschließlich UI Elementen, die nicht von dem Global Volume Effekt beeinflusst werden sollen.

4.3.2.2 Canvas 2

(siehe „Home Screen“)

4.3.3 Wichtige Hierarchie-Struktur!

Alle hier vermerkten Benennungen sind unbedingt einzuhalten, da sie relevant sind, um die dazugehörigen GameObjects für Scripte zu finden!

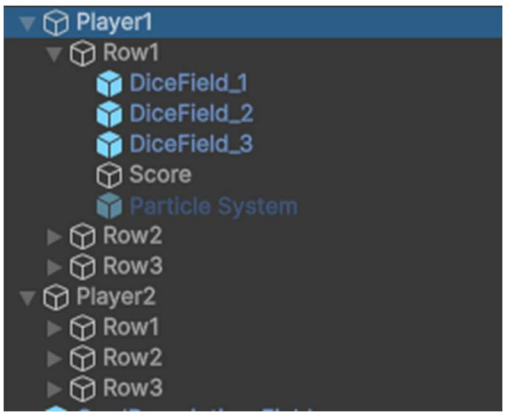


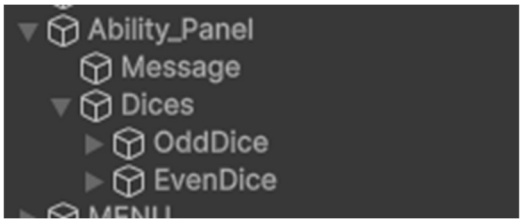
4.3.3.1 Ability Cards

Alle Fähigkeitskarten liegen in einem Parent mit dem Namen „(Player)_SpecialEffects“ und haben die Namen im Bild. Über den Karten liegt (vorrübergehend) ein Empty GameObject (mit Child!) als Platzhalter, damit keine Fehler im Code entstehen.

4.3.3.2 Reihen, Felder und Würfel

Unter einem Parent-Object „(Player)“ liegen alle dem zugehörigen Reihen mit der Benennung „Row(Nummer)“. Darunter liegen dann jeweils 3 Würfelfelder „DiceField_(Nummer), auf die später die Würfel als Childs gesetzt werden können. Außerdem relevant für das Game sind die einzeln zählenden Punktestände für jede Reihe. Dafür wird unter den Würfelfeldern ein TMP_Text-Object gelegt, was von den Scripten angesprochen und gefüllt wird.



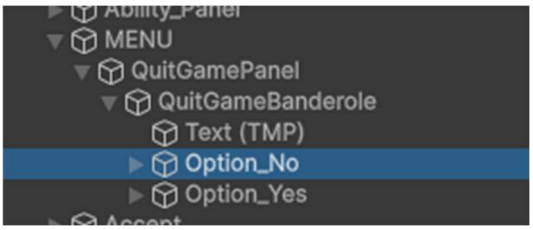


4.3.3.3 Ability Panel

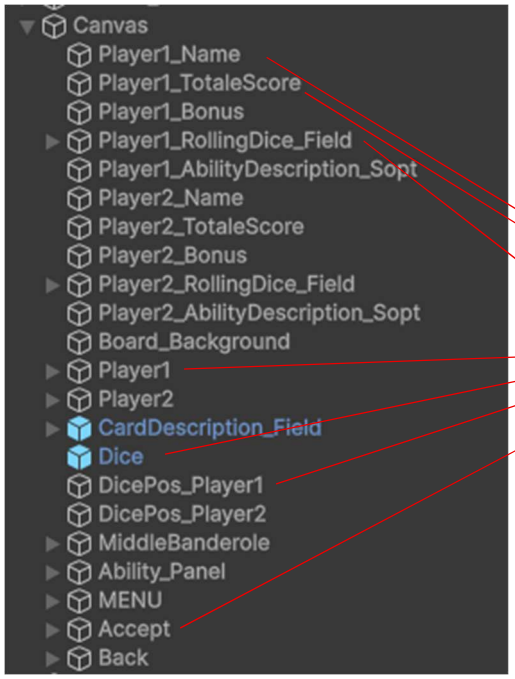
Auf dem „Ability Panel“ liegen ein TMP_Text-GameObject und darunter, eingekapselt in ein weiteres Parent, die Auswahl-Objekte für die Destiny Dice Fähigkeit. Diese tragen ebenfalls die Namen ihrer Darstellung „OddDice“ und „EvenDice“.

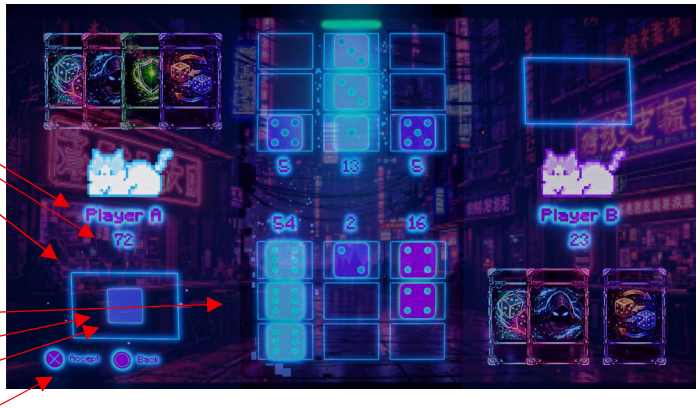
4.3.3.4 Quit Menu Panel

Hier muss die Benennung bei den Parent Objekten nicht die gleiche wie in dem Bild sein. Dennoch sollten sowohl die Rangordnung der letzten Child Objekte sowie ihre Benennung eingehalten werden.



4.3.3.5 Gesamte Hierarchie





5 SCRIPTS

5.1 WICHTIGE ANMERKUNGEN!

Alle Script Verweise werden automatisch in der Start() gefüllt und müssen nicht per Hand zugewiesen werden. Erklärungen zu Benennungen im Script werden im Kapitel „Engine und Coding Lexika“ aufgelistet.

5.2 OVER ALL SCRIPTS

5.2.1 GameManager.cs

→ Dieses Script steuert die Kernlogik des Spiels: Es verwaltet den aktuellen Spielmodus, die aktiven Spieler, die Würfel- und Spezialaktionen, die Punktevergabe pro Reihe, den Spielerwechsel inklusive Aktivieren/Deaktivieren von Buttons und Reduzierung von Schildlebenspunkten und prüft, ob ein Spieler gewonnen hat.
Außerdem sorgt es dafür, dass die Benutzeroberfläche (Buttons, Interaktionen, Spezialeffekte) immer korrekt auf den aktuellen Spielzustand reagiert.

5.2.1.1 Funktionen

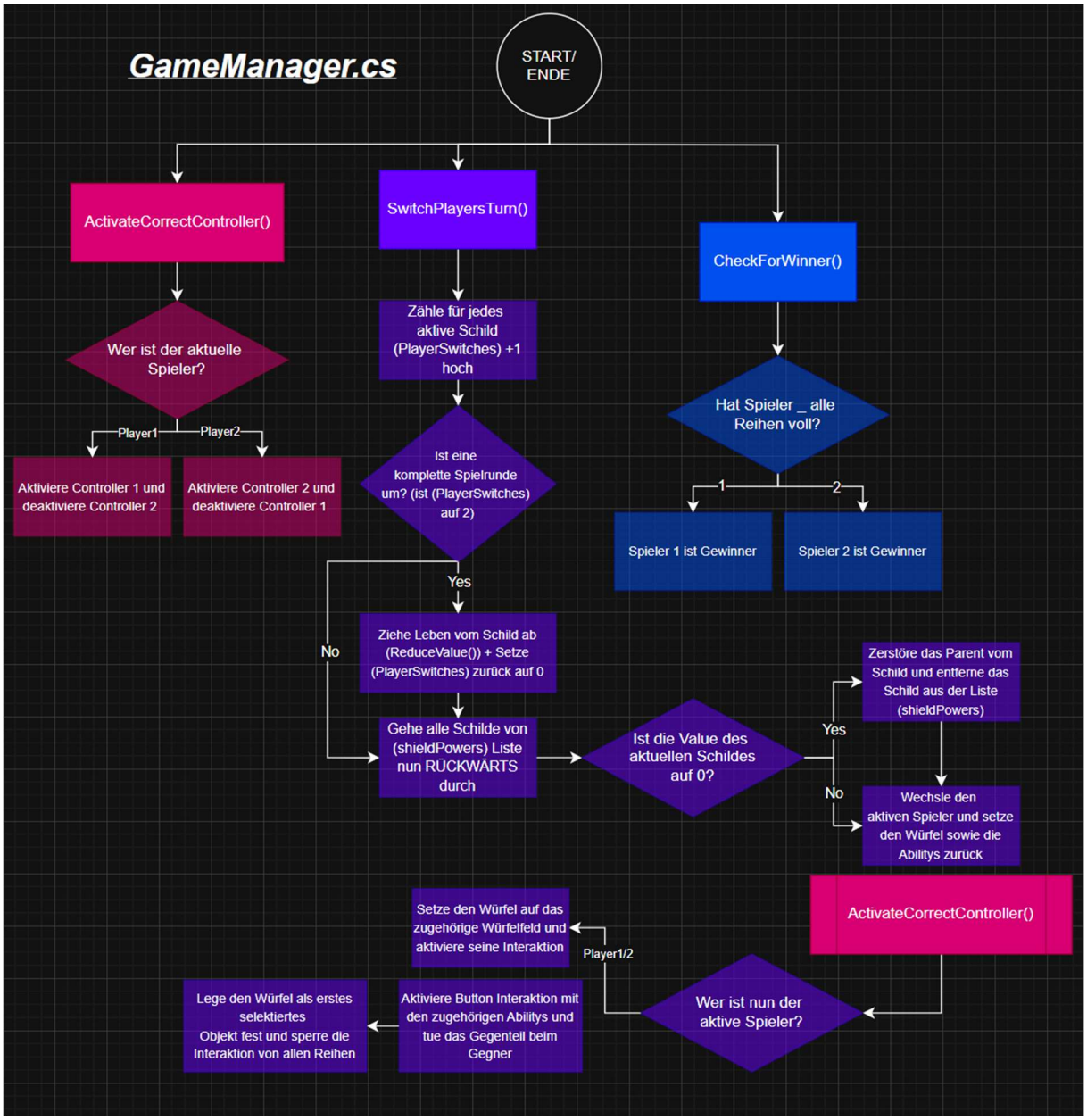
- Das Initialisieren des Spiels in der Szene GameScreen, inklusive zufälliger Startspielerwahl, Setzen des aktiven Spielers und Positionieren des Würfels beim jeweiligen Spieler (über ScreenLogic.cs)
- Der Player Switch (SwitchPlayersTurn()) -> Verwalten von aktiven Schilden (Health-Abzug, Entfernen wenn aufgebraucht), Spielerwechsel (+ Aktivierung der jeweiligen Abilitys), das Sperren und Aktivieren der jeweiligen Controller (ActivateCorrectController()) und Set-Up vor nächstem Zug (Würfel selektiert, zur Sicherheit nochmal alle Reihen und darin liegende Würfel nicht interaktiv setzen)
- Das Aktivieren bzw. Deaktivieren der Interaktionsmöglichkeiten von Controllern, sodass immer nur ein Spieler das Spielgestehen kontrollieren
- Das Überprüfen der Siegbedingung anhand vollständig gefüllter Reihen und das Festlegen des Gewinners für den Spielabschluss

5.2.1.2 Properties

- Player Name 1: Der Name, der für den linken Spieler angezeigt werden soll (default setting: „PlayerA“)
- Player Name 2: Der Name, der für den rechten Spieler angezeigt werden soll (default setting: „PlayerB“)
- Default Color_Cards: Die Farbe, die die Fähigkeitskarten tragen sollen wenn sie normal selektiert werden dürfen (default setting: DAC0FF)
- Double Color: Die Farbe, die die Würfel tragen sollen wenn sie zu Doppel-Würfeln werden (default setting: EDCD5D)
- Triple Color: Die Farbe, die die Würfel tragen sollen wenn sie zu Dreier-Würfeln werden (default setting: 5D59F1)
- Diagonale Bonus Color/Exodia Color: Die Farbe, die die Würfel tragen sollen wenn sie zu Dreier-Würfeln werden (default setting: CFCFCF)
- Twisting Color: Die Farbe, die die Würfel tragen sollen wenn sie bei der Twister Fähigkeit ausgewählt werden (default setting: FF0C00)

5.2.1.3 Setup

- Liegt auf: GameObject „Logic“
- Benötigt: /



5.2.2 HighlightButton.cs

→ Dieses Script sorgt dafür, dass der Button, der gerade im Event System ausgewählt ist, visuell hervorgehoben wird: Es aktiviert eine Outline für den aktuell ausgewählten Button und deaktiviert sie für alle anderen.

5.2.2.1 Funktionen

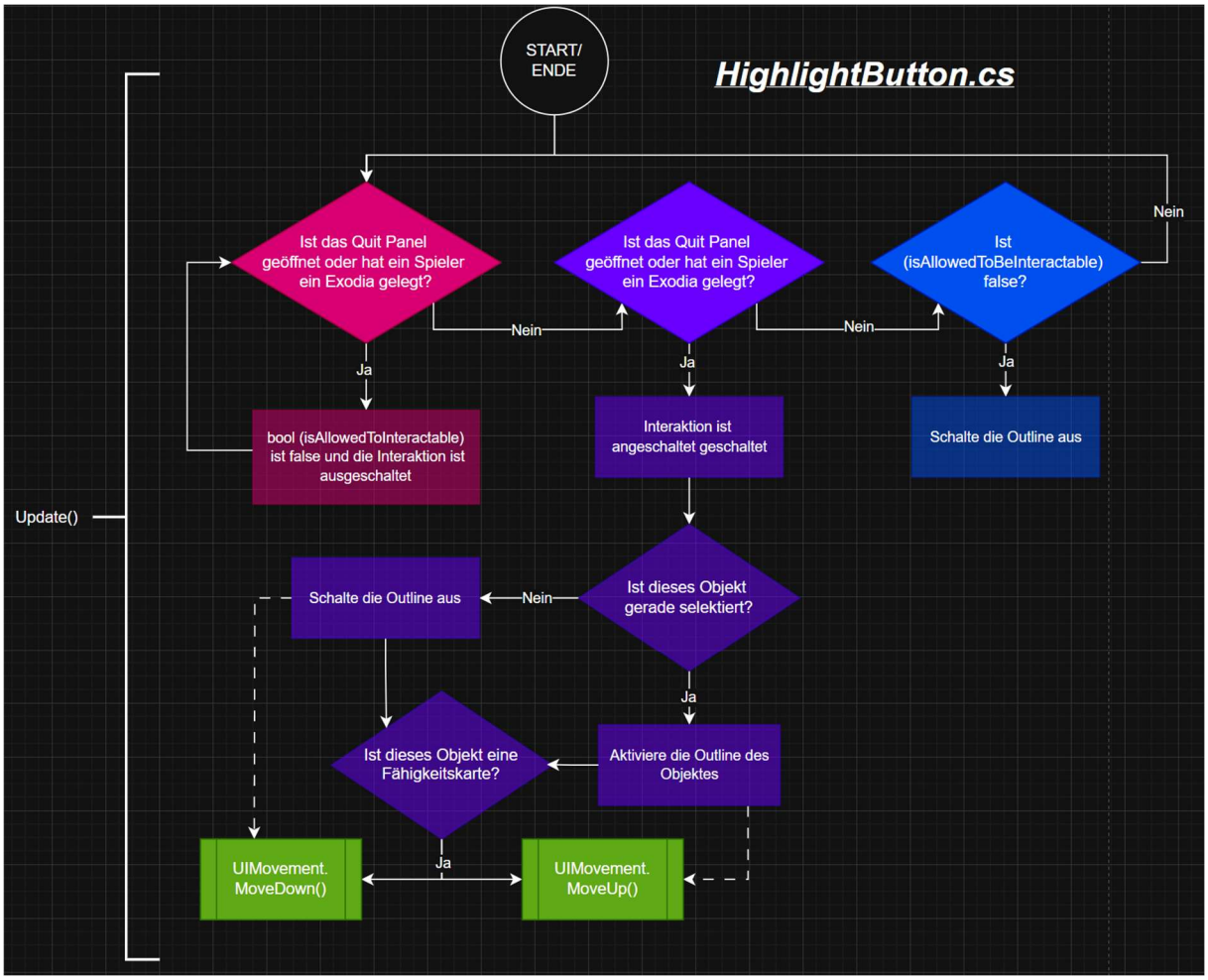
- Das Aktivieren und Deaktivieren von Outlines auf Buttons, wenn sie selektiert werden bzw. nicht mehr selektiert sind

5.2.2.2 Properties

- /

5.2.2.3 Setup

- Liegt auf: alle interagierbaren UI Elemente
- Benötigt: /



5.2.3 InputMananger.cs

→ Dieses Script sorgt dafür, dass die Eingabe des Spielers für das Öffnen des Quit-Panels erkannt wird. Es überprüft in jedem Frame, ob die definierte Quit-Action ausgelöst wurde, und speichert das Ergebnis in einer boolschen Variable (QuitInput), auf die andere Scripts zugreifen können.

5.2.3.1 Funktionen

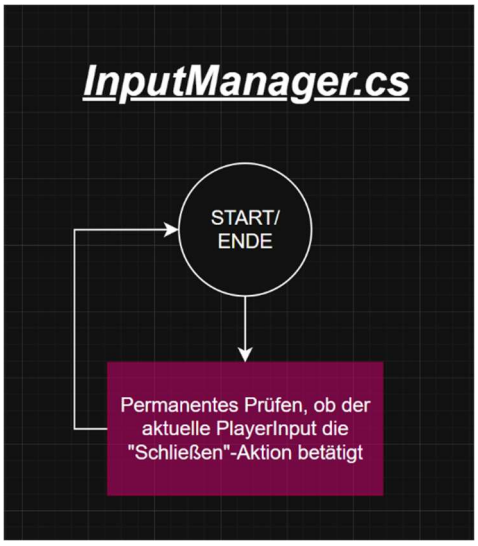
- Das Erstellen eines Singletons, um sicherzustellen, dass nur eine Instanz von InputManager existiert
- Das Abfangen der Spieler-Eingabe für das Öffnen des Quit-Panels über die definierte Action „Denied“
- Das Aktualisieren eines boolschen Werts (QuitInput) in jedem Frame, der angibt, ob die Quit-Action gerade ausgelöst wurde

5.2.3.2 Properties

- /

5.2.3.3 Setup

- Liegt auf: GameObject „PlayerInputPrefab“ (Mehr zum erstellenden Prefab bei PlayerJoinManager.cs)
- Benötigt: /



5.2.4 MenuManager.cs

→ Dieses Script überwacht, ob die „Quit“-Taste gedrückt wurde (über InputManager).
Je nachdem, ob das Quit-Menü bereits geöffnet ist oder nicht, öffnet es das Quit-Panel oder schließt es wieder, und merkt sich den aktuellen Zustand.

5.2.4.1 Funktionen

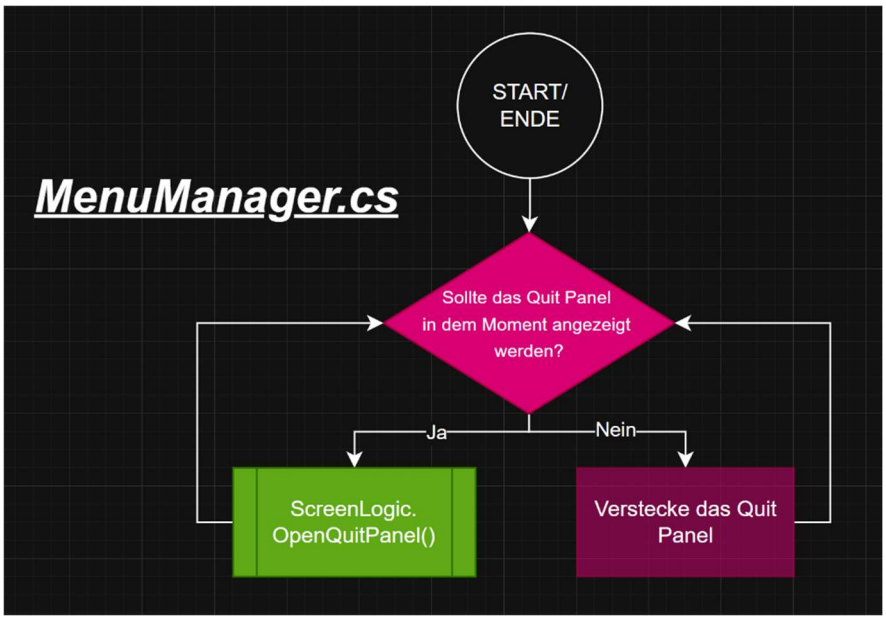
- Das Öffnen des Quit-Game-Panels, wenn der Spieler die Quit-Eingabe auslöst und das Panel aktuell geschlossen ist
- Das Schließen des Quit-Game-Panels, wenn der Spieler die Quit-Eingabe erneut auslöst und das Panel aktuell geöffnet ist
- Das Zwischenspeichern des aktuellen Panel-Zustands über die boolesche Variable isQuitPanelOpen, um zwischen Öffnen und Schließen zu unterscheiden

5.2.4.2 Properties

- /

5.2.4.3 Setup

- Liegt auf: GameObject „PlayerInputPrefab“ (Mehr zum erstellenden Prefab bei PlayerJoinManager.cs)
- Benötigt: /



5.2.5 ScreenLogic.cs

→ Dieses Script kümmert sich primär um die Benutzeroberfläche, Navigation zwischen Szenen, Anzeige von Panels und das Starten/Beenden des Spiels. Es ist der zentrale Controller für UI-Interaktionen.

5.2.5.1 Funktionen

- Das Initialisieren wichtiger UI-Elemente (QuitGamePanel, Continue-Button, MiddleBanderole + Name des Anfang-Spielers, AbilityPanel) beim Spielstart
- Das ständige Überprüfen und Vermerken des zuletzt selektierten Buttons (wichtig, um nach dem Schließen eines Fensters wieder auf den davor angewählten Button zurückzukehren)
- Das Erfassen der zuletzt ausgewählten Buttons und Weiterleiten der Spielerinteraktion an AbilityManager, abhängig vom aktuellen Spielmodus (Destiny, Thief, Twister, Shield)
- Das Anzeigen des Gewinners am Spielende im MiddleBanderole-Panel an und aktivieren des Continue-Buttons
- Das Öffnen des Ability-Panels, Anzeigen der passenden Nachricht zur gewählten Spezialaktion und Deaktivieren der Würfel-Buttons, um Spielerinteraktionen zu steuern
- Das Managen der Anzeige auf dem Controller Setup Panel am Anfang des Spiels
- Die Steuerung des Quit-Panels und eventuelle Schließung des Spiels
- Das Laden der Home-Screen-Szene nach dem Ende des Spiels oder beim Abbrechen

5.2.5.2 Properties

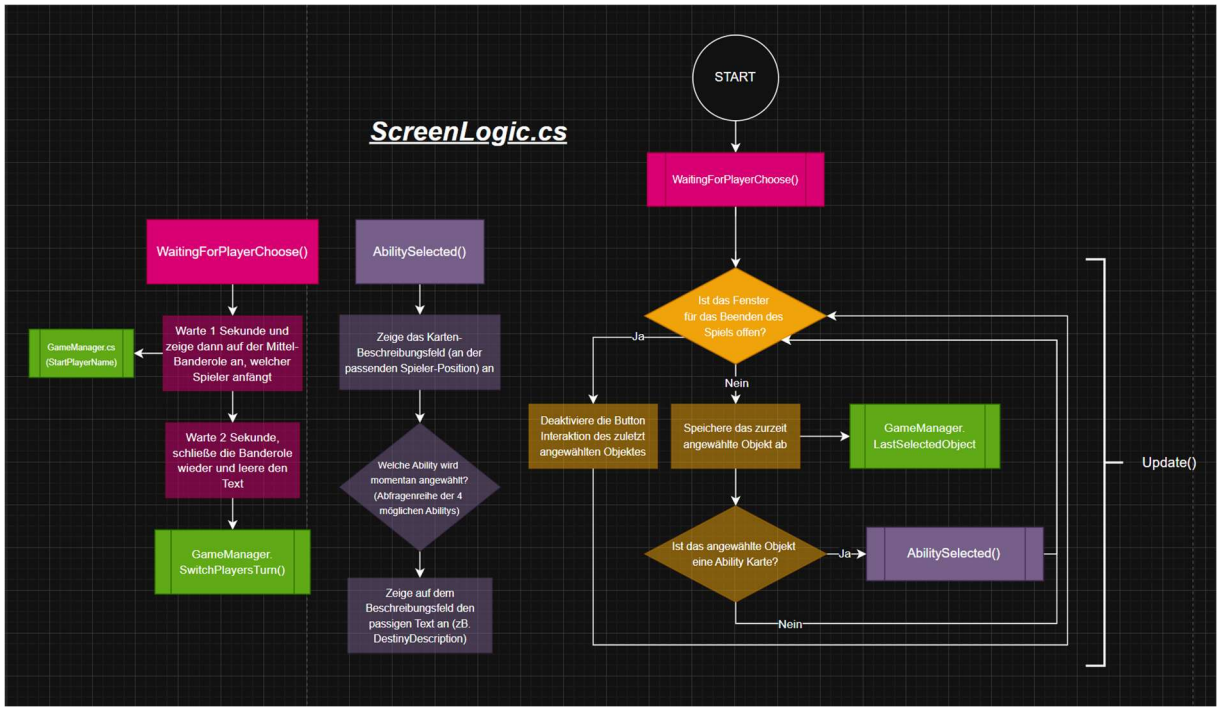
- Destiny Message: Der Text, der angezeigt werden soll, sobald ein Spieler die Destiny-Ability auswählt (default setting: „Choose your Dice Numbers!“)
- Thief Message: Der Text, der angezeigt werden soll, sobald ein Spieler die Thief-Ability auswählt (default setting: „Choose an Opponent Dice!“)
- Shield Message: Der Text, der angezeigt werden soll, sobald ein Spieler die Shield-Ability auswählt (default setting: „Choose a Row to protect!“)
- Twister Message: Der Text, der angezeigt werden soll, sobald ein Spieler die Twister-Ability auswählt (default setting: „Choose 2 Dices!“)
- Destiny Description: Der Text, der angezeigt werden soll, wenn ein Spieler über die Destiny-Ability hovert (default setting: „Choose between Even or Odd Numbers and roll only those next!“)
- Thief Description: Der Text, der angezeigt werden soll, wenn ein Spieler über die Thief-Ability hovert (default setting: „Steal a Dice from your Oponent! It will be placed in your own Row across.“)
- Shield Description: Der Text, der angezeigt werden soll, wenn ein Spieler über die Shield-Ability hovert (default setting: „Protect one of your Rows for 3 Rounds! This way you won't lose any dice and you'll destroy opponent one's.“)
- Twister Description: Der Text, der angezeigt werden soll, wenn ein Spieler über die Twister-Ability hovert (default setting: „Swap the positions of two of your dice that you choose first!“)

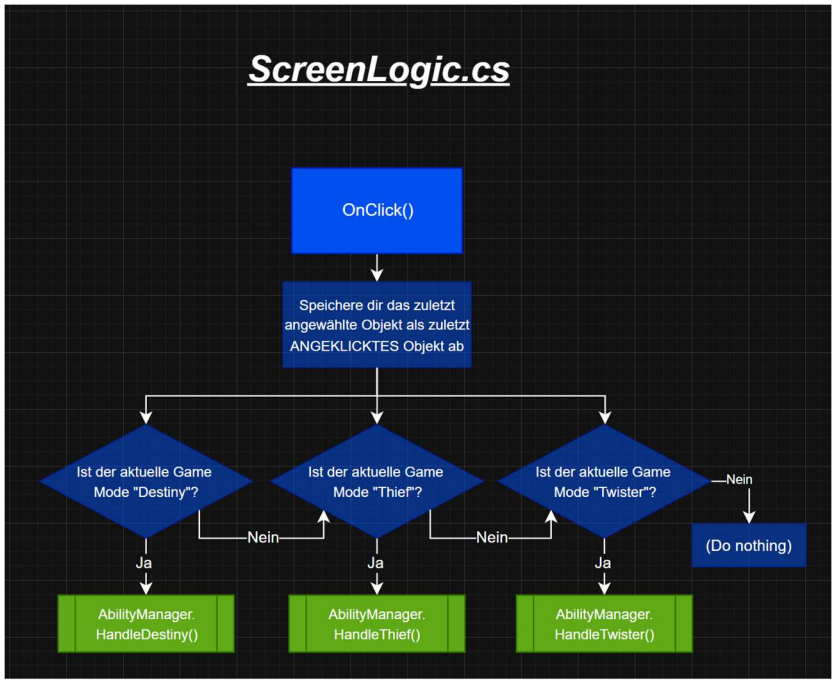
5.2.5.3 Setup

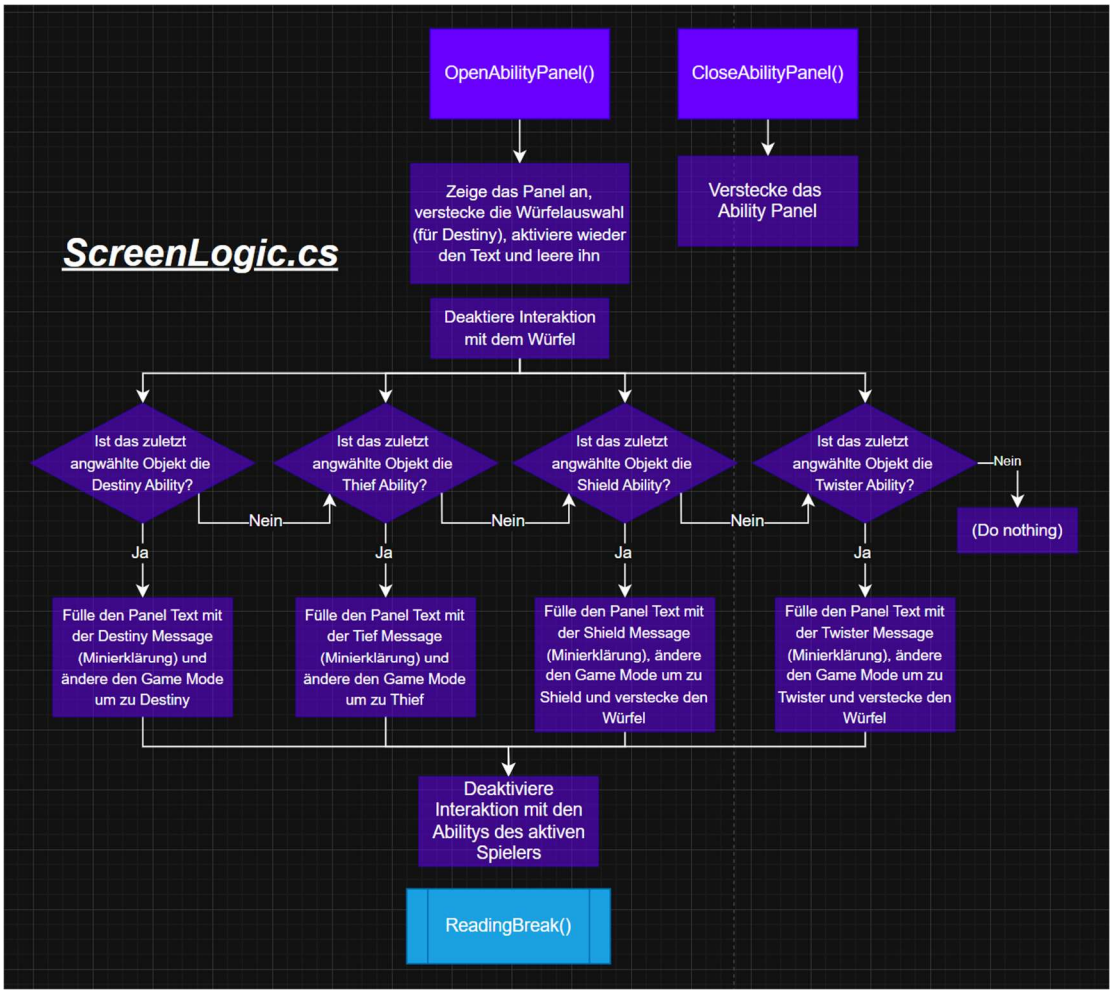
- Liegt auf: GameObject „Logic“
- Benötigt: /

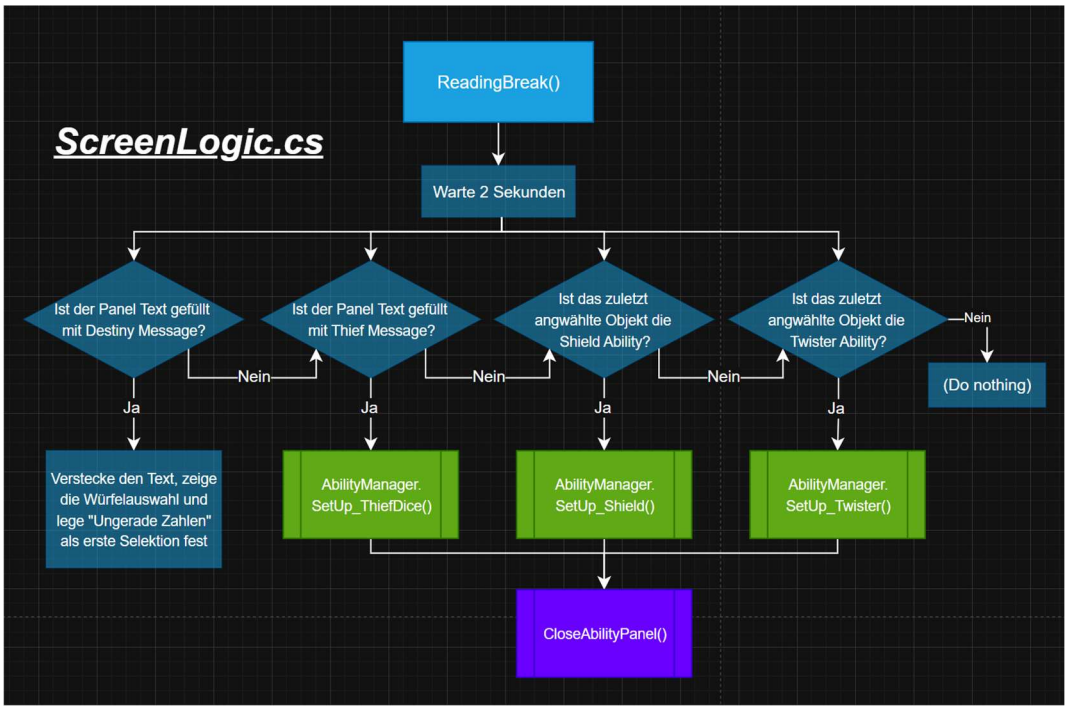
Hier bitte beachten!

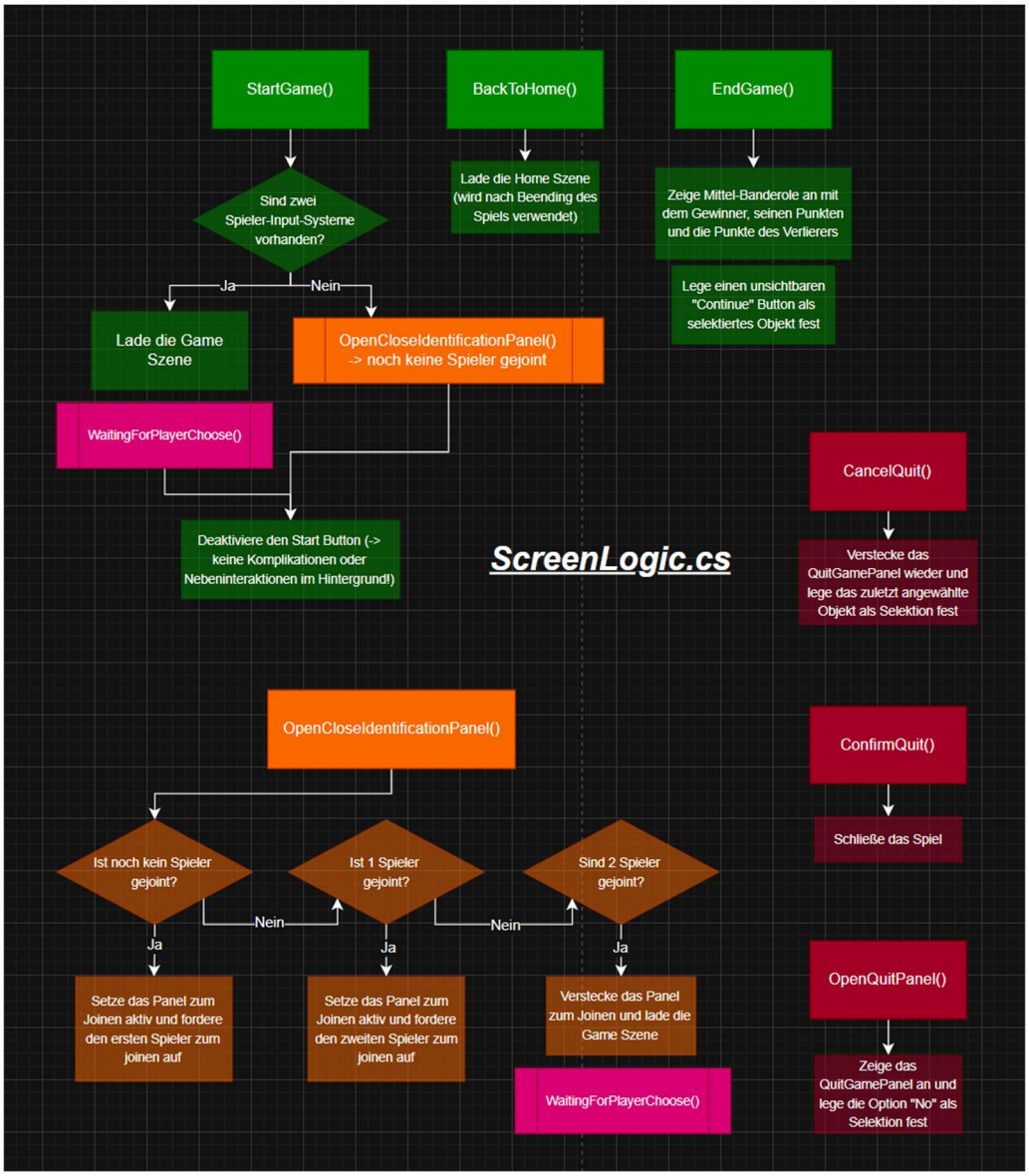
- On Click Event:
  -> Jeder Button bekommt dieses Event zugewiesen, BIS AUF DIE BUTTONS IM QUIT PANEL!
- OnClick() sorgt dafür, dass die Interaktion mit einem Button im GameMananger.cs gespeichert wird. Diese Prüfung braucht das System aber, um nach der Schließung des Quit Panels wieder auf den zuletzt angewählten Button zurückzukehren. Somit darf diese Überprüfung nicht von Interaktionen aus dem Quit Panel überschrieben werden!






ScreenLogic.cs
OpenAbilityPanel()
CloseAbilityPanel()
Zeige das Panel an, verstecke die Würfelauswahl (für Destiny), aktiviere wieder den Text und leere ihn
Verstecke das Ability Panel
Deaktiere Interaktion mit dem Würfel
Ist das zuletzt angwählte Objekt die Destiny Ability?
Ist das zuletzt angwählte Objekt die Thief Ability?
Ist das zuletzt angwählte Objekt die Shield Ability?
Ist das zuletzt angwählte Objekt die Twister Ability?
Nein
Ja
(Do nothing)
Fülle den Panel Text mit der Destiny Message (Minierklärung) und ändere den Game Mode um zu Destiny
Fülle den Panel Text mit der Tief Message (Minierklärung) und ändere den Game Mode um zu Thief
Fülle den Panel Text mit der Shield Message (Minierklärung), ändere den Game Mode um zu Shield und verstecke den Würfel
Fülle den Panel Text mit der Twister Message (Minierklärung), ändere den Game Mode um zu Twister und verstecke den Würfel
Deaktiviere Interaktion mit den Abilitys des aktiven Spielers
ReadingBreak()


ScreenLogic.cs
ReadingBreak()
Warte 2 Sekunden
Ist der Panel Text gefüllt mit Destiny Message?
Ist der Panel Text gefüllt mit Thief Message?
Ist das zuletzt angwählte Objekt die Shield Ability?
Ist das zuletzt angwählte Objekt die Twister Ability?
Nein
Ja
(Do nothing)
Verstecke den Text, zeige die Würfelauswahl und lege "Ungerade Zahlen" als erste Selektion fest
AbilityManager. SetUp_ThiefDice()
AbilityManager. SetUp_Shield()
AbilityManager. SetUp_Twister()
CloseAbilityPanel()


StartGame()
BackToHome()
EndGame()
Sind zwei Spieler-Input-Systeme vorhanden?
Lade die Home Szene (wird nach Beending des Spiels verwendet)
Zeige Mittel-Banderole an mit dem Gewinner, seinen Punkten und die Punkte des Verlierers
Lege einen unsichtbaren "Continue" Button als selektiertes Objekt fest
Ja
Nein
Lade die Game Szene
OpenCloseIdentificationPanel() -> noch keine Spieler gejoint
WaitingForPlayerChoose()
Deaktiviere den Start Button (-> keine Komplikationen oder Nebeninteraktionen im Hintergrund!)
ScreenLogic.cs
CancelQuit()
Verstecke das QuitGamePanel wieder und lege das zuletzt angewählte Objekt als Selektion fest
OpenCloseIdentificationPanel()
ConfirmQuit()
Schließe das Spiel
Ist noch kein Spieler gejoint?
Ist 1 Spieler gejoint?
Sind 2 Spieler gejoint?
Nein
Nein
Ja
Ja
Ja
Setze das Panel zum Joinen aktiv und fordere den ersten Spieler zum joinen auf
Setze das Panel zum Joinen aktiv und fordere den zweiten Spieler zum joinen auf
Verstecke das Panel zum Joinen und lade die Game Szene
WaitingForPlayerChoose()
OpenQuitPanel()
Zeige das QuitGamePanel an und lege die Option "No" als Selektion fest

5.3 ONLY HOME RELEVANT

5.3.1 PlayerJoinManager.cs

→ Dieses Script ist dafür zuständig, den nötigen Player Join am Anfang vom Spiel zu regeln und beide PlayerInputs in die Gane Szene weiterzugeben.

5.3.1.1 Funktionen

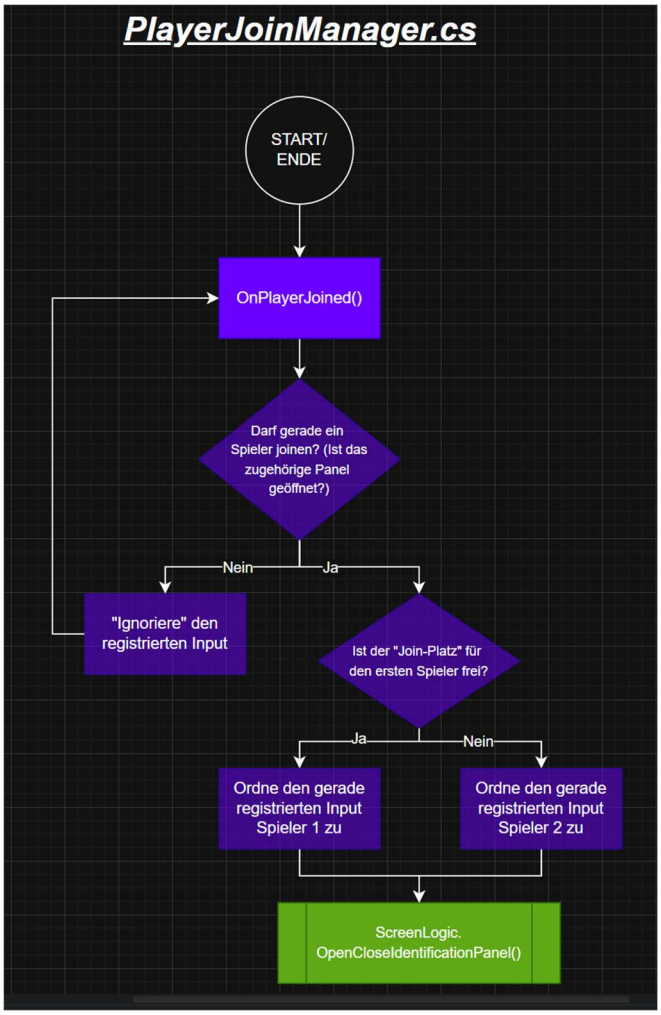
- Das Speichern vom Player Input vom ersten Spieler, sobald dieser eine Taste auf dem Controller klickt (Bestätigungs-Bedingung) -> OnPlayerJoined()
- Danach das Speichern vom Player Input vom zweiten Spieler, sobald dieser eine Taste auf dem Controller klickt (Bestätigungs-Bedingung) -> OnPlayerJoined()

5.3.1.2 Properties

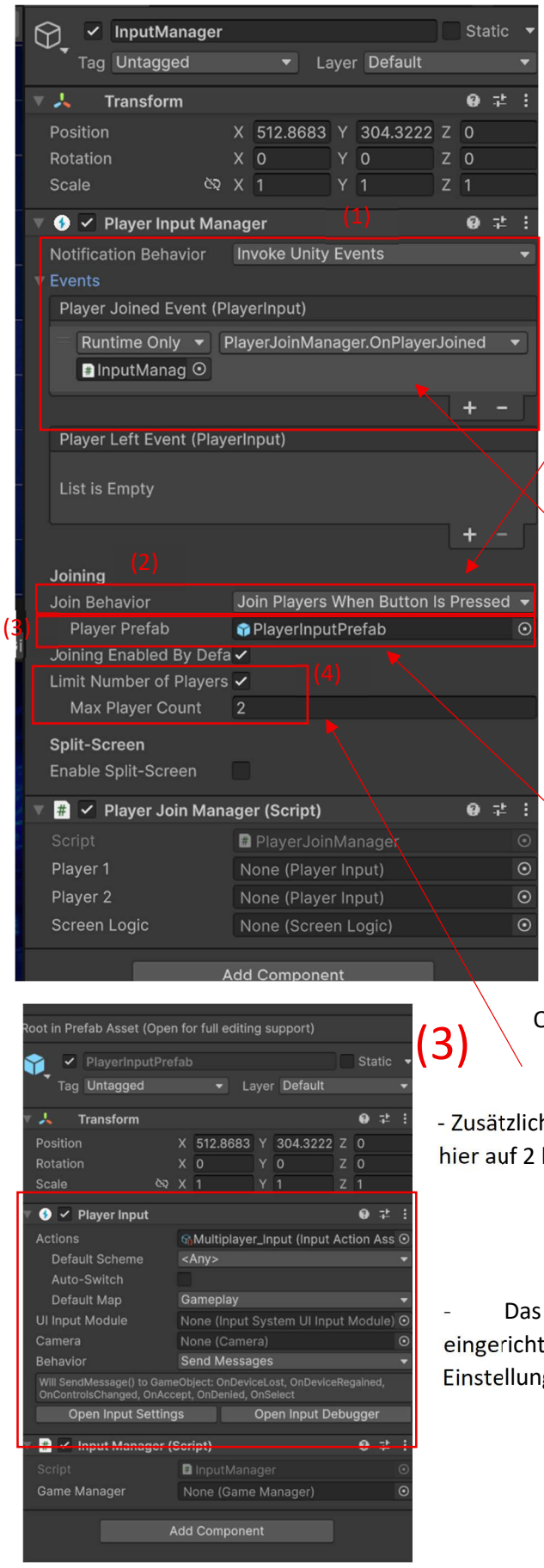
- /

5.3.1.3 Setup

- Liegt auf: GameObject „InputManager“
- Benötigt: Player Input Manager



- Erklärung der Verknüpfung:



- Der **Player Input Manager** ist eine Unity Input System Komponente, die dafür da ist, einen local-Multiplayer zu verwalten und automatisch Controller oder Keyboards Spielern zuzuordnen, damit mehr Spieler joinen können

- Kann über einstellbare Wege neue Input Systeme erkennen -> Hier verwenden wir „Join Players When Button Is Pressed“, was nichts anderes bedeutet als das die Spieler eine Taste auf ihrem Controller klicken müsssen, um dem Spiel als eigener Spieler beizutreten (siehe (2))

- Wie genau nach einem Input eines Spielers reagiert werden soll wird bei „Notification Behavior“ ausgewählt -> Wir verwenden „Invoke Unity Events“, da diese Behavior sehr üblich und vorteilhafter für Turn-based Controller Sperrung, UI Navigation und mehrere Player Inputs ist (siehe (1))
- Als auszuführendes Event geben wir ihm unsere geschriebene „OnPlayerJoined()“

- Bei einem Join Versuch erstellt der Player Input Manager automatisch ein PlayerInput-GameObject -> Zum Erstellen geben wir ihm ein davor eingerichtetes „InputPlayerPrefab“, damit erhält jeder Spieler sein eigenes Input-Objekt (siehe (3))

- Zusätzlich kann man ein Limit an Spielern eintragen, was hier auf 2 beschränkt ist (siehe (4))

- Das Prefab trägt als Input Action Asset den vorher eingerichteten „Multiplayer_Input“ (Erklärung zur Einstellung unter „Game Setup -> Allgemein“)

5.4 ONLY GAME RELEVANT

5.4.1 AbilityManager.cs

→ Dieses Script verwaltet die Spezialaktionen (Destiny, Thief, Twister, Shield) der Spieler. Es arbeitet mit ScreenLogic.sc zusammen, kapselt aber die Logik für Interaktionen mit Würfeln, Reihen und Spezialfähigkeiten ab.

5.4.1.1 Funktionen

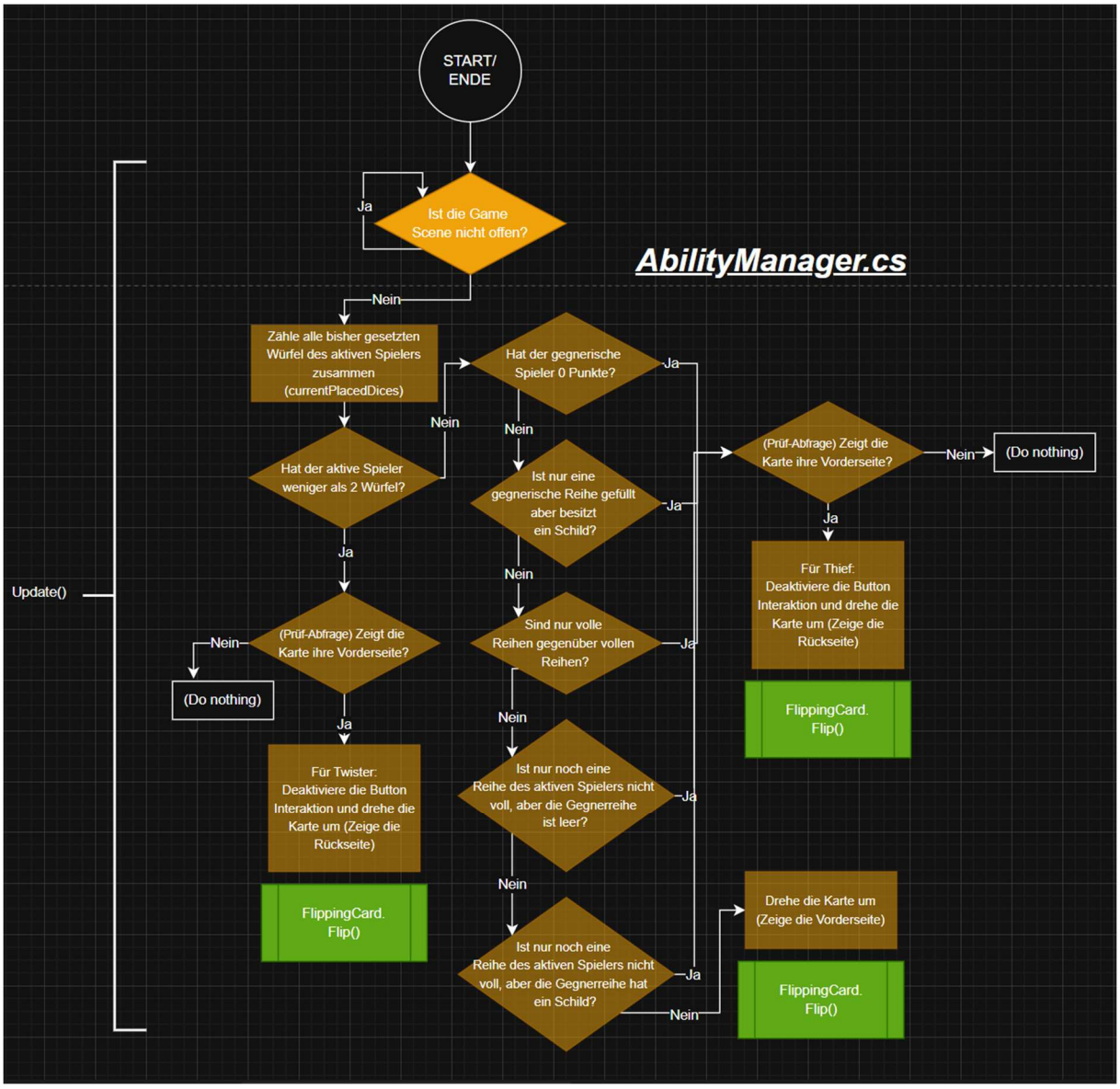
- Das Verwalten der Spezialaktionen Destiny, Thief, Twister und Shield, inklusive Aktivierung und Deaktivierung der jeweiligen Buttons abhängig vom Spielzustand
- Destiny Dice
  - Handling: Abfrage, ob der Spieler bereits sein Schicksal gewählt hat (gerade oder ungerade Zahl), worauf dann der spezielle Würfelwurf folgt (DiceRoll.RollDestinyDice()) und Spielmodus auf Normal gestellt wird (da von da an der Ablauf normal weitergeht)
- Thief Dice
  - Set Up: Die Interaktion mit den gegnerischen Reihen aktivieren, unter Beachtung verschiedener Bedingungen bzw. Szenarien wo die Fähigkeit nicht eingesetzt werden kann
  - Handling: Nach Reihenauswahl Interaktion mit den darin liegenden Würfeln aktivieren + nach Würfelauswahl Interaktionen deaktivieren, Befehl zum Würfeln einer Kopie (DiceRoll.RollThiefDice()) abgeben und die eigene gegenüberliegende Reihe automatisch selektieren, danach Game Mode auf Normal zurücksetzen
- Twister
  - Set Up: Die Interaktion mit den eigenen Reihen aktivieren
  - Handling: Nach Reihenauswahl Interaktion mit den darin liegenden Würfeln aktivieren + nach Würfelauswahl Zurücksetzen der Interaktionsmöglichkeit, Anzahl der ausgewählten Würfel vermerken und die Positionen tauschen, sobald 2 Würfel ausgewählt wurden, danach Punkteverrechnung und Spielerwechsel
- Schild
  - Set Up: Das Erzeugen eines Shield-Objektes, Ausrichtung der Health Bar entsprechend des aktuellen Spielers (da die Bar immer außen positioniert ist) und Aktivieren der Interaktion mit den Spielerreihen (vorerst ist das Schild nicht sichtbar, wird aber Aktiviert, sobald eine Reihe angewählt wurde)
- Das Vorbereiten der ersten auswählbaren Reihe oder des Würfels für jede Spezialaktion (SetUp_ThiefDice, SetUp_Twister, SetUp_Shield), um die Spielerinteraktion sauber zu steuern
- Das kontinuierliche Prüfen der Bedingungen für Twister und Thief, um die Verfügbarkeit der Spezialfähigkeiten im UI korrekt anzuzeigen (z. B. Twister nur bei mehr als 2 Dices, Thief nur bei mind. einem existierenden Gegnerwürfel)

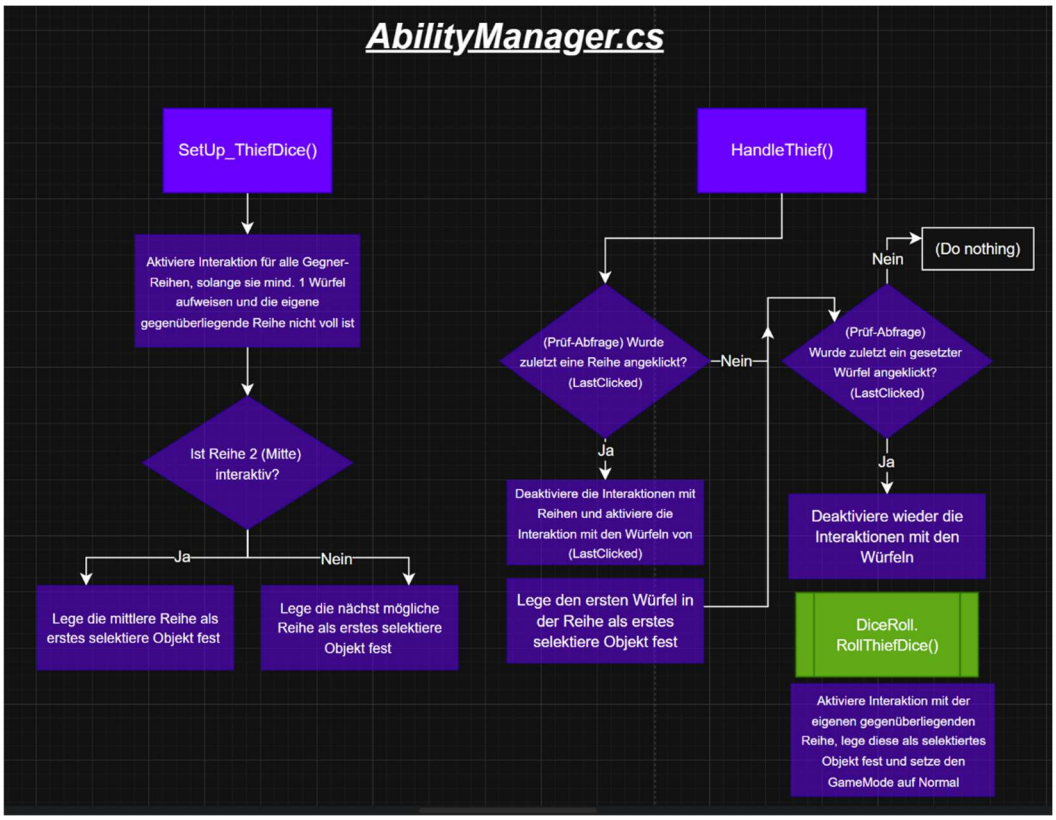
5.4.1.2 Properties

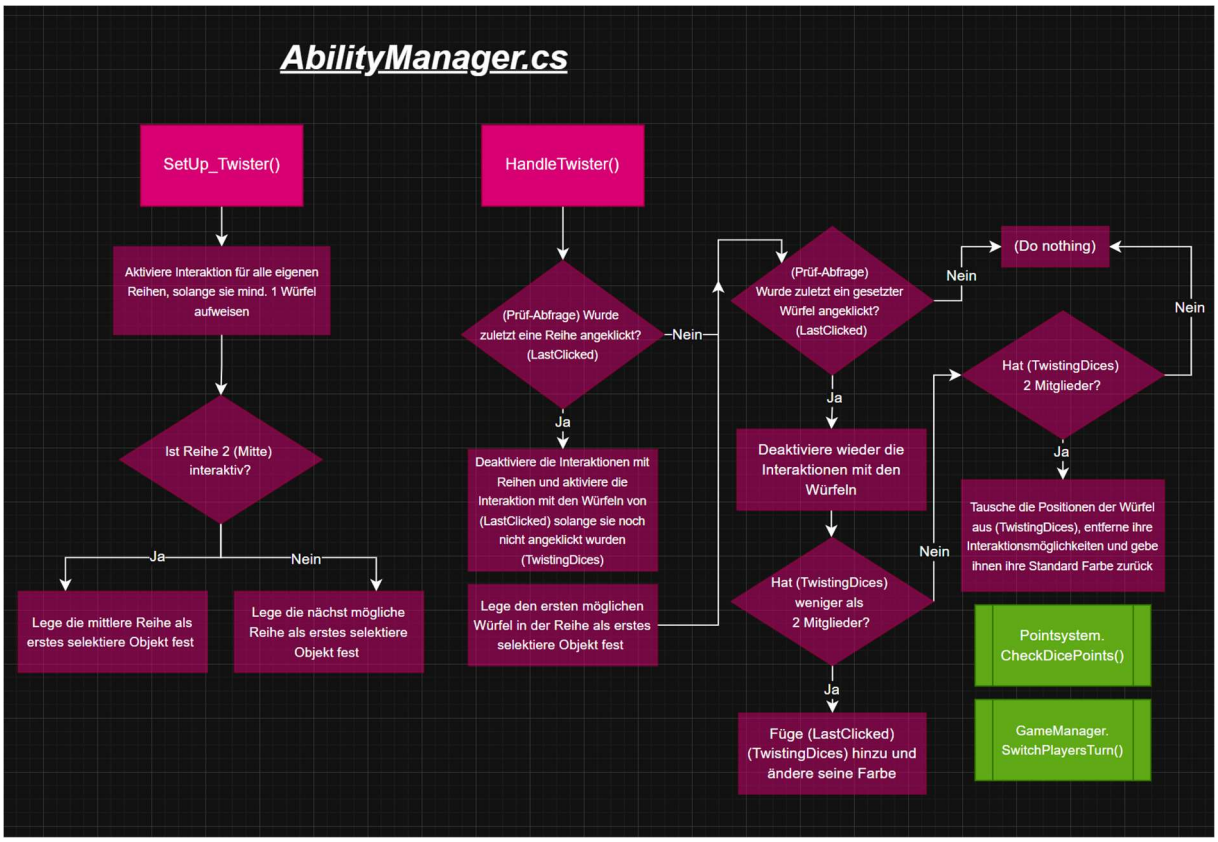
- /

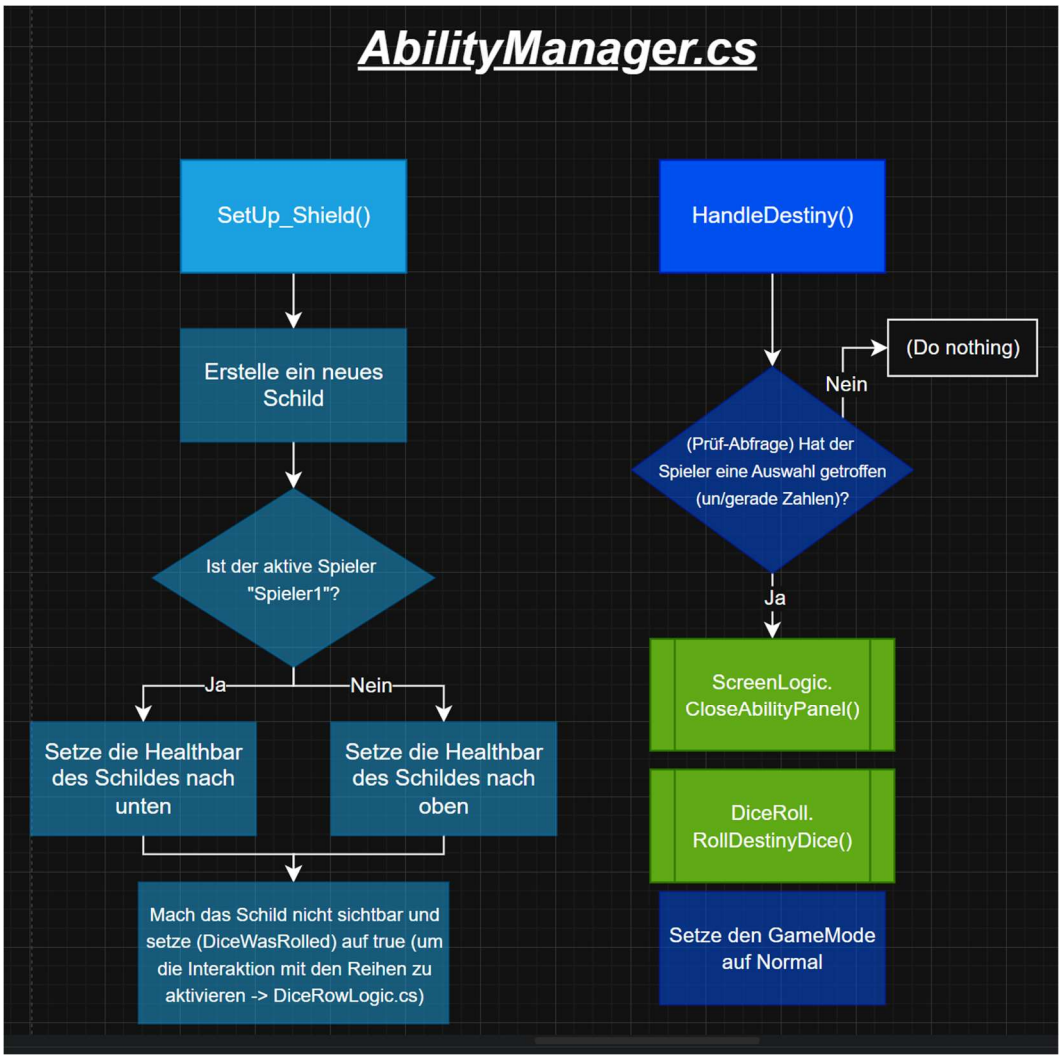
5.4.1.3 Setup

- Liegt auf: dem GameObject („Logic“)
- Benötigt: /




AbilityManager.cs
SetUp_ThiefDice()
Aktiviere Interaktion für alle Gegner-Reihen, solange sie mind. 1 Würfel aufweisen und die eigene gegenüberliegende Reihe nicht voll ist
Ist Reihe 2 (Mitte) interaktiv?
Ja
Nein
Lege die mittlere Reihe als erstes selektiere Objekt fest
Lege die nächst mögliche Reihe als erstes selektiere Objekt fest
HandleThief()
(Prüf-Abfrage) Wurde zuletzt eine Reihe angeklickt? (LastClicked)
Nein
Ja
Deaktiviere die Interaktionen mit Reihen und aktiviere die Interaktion mit den Würfeln von (LastClicked)
Lege den ersten Würfel in der Reihe als erstes selektiere Objekt fest
(Prüf-Abfrage) Wurde zuletzt ein gesetzter Würfel angeklickt? (LastClicked)
Nein
(Do nothing)
Ja
Deaktiviere wieder die Interaktionen mit den Würfeln
DiceRoll. RollThiefDice()
Aktiviere Interaktion mit der eigenen gegenüberliegenden Reihe, lege diese als selektiertes Objekt fest und setze den GameMode auf Normal


AbilityManager.cs
SetUp_Twister()
Aktiviere Interaktion für alle eigenen Reihen, solange sie mind. 1 Würfel aufweisen
Ist Reihe 2 (Mitte) interaktiv?
Ja
Nein
Lege die mittlere Reihe als erstes selektiere Objekt fest
Lege die nächst mögliche Reihe als erstes selektiere Objekt fest
HandleTwister()
(Prüf-Abfrage) Wurde zuletzt eine Reihe angeklickt? (LastClicked)
Nein
Ja
Deaktiviere die Interaktionen mit Reihen und aktiviere die Interaktion mit den Würfeln von (LastClicked) solange sie noch nicht angeklickt wurden (TwistingDices)
Lege den ersten möglichen Würfel in der Reihe als erstes selektiere Objekt fest
(Prüf-Abfrage) Wurde zuletzt ein gesetzter Würfel angeklickt? (LastClicked)
Nein
(Do nothing)
Ja
Deaktiviere wieder die Interaktionen mit den Würfeln
Hat (TwistingDices) weniger als 2 Mitglieder?
Nein
Ja
Füge (LastClicked) (TwistingDices) hinzu und ändere seine Farbe
Hat (TwistingDices) 2 Mitglieder?
Nein
Ja
Tausche die Positionen der Würfel aus (TwistingDices), entferne ihre Interaktionsmöglichkeiten und gebe ihnen ihre Standard Farbe zurück
Pointsystem. CheckDicePoints()
GameManager. SwitchPlayersTurn()


AbilityManager.cs
SetUp_Shield()
Erstelle ein neues Schild
Ist der aktive Spieler "Spieler1"?
Ja
Nein
Setze die Healthbar des Schildes nach unten
Setze die Healthbar des Schildes nach oben
Mach das Schild nicht sichtbar und setze (DiceWasRolled) auf true (um die Interaktion mit den Reihen zu aktivieren -> DiceRowLogic.cs)
HandleDestiny()
(Do nothing)
Nein
(Prüf-Abfrage) Hat der Spieler eine Auswahl getroffen (un/gerade Zahlen)?
Ja
ScreenLogic. CloseAbilityPanel()
DiceRoll. RollDestinyDice()
Setze den GameMode auf Normal

5.4.2 AbilityUsedUpCheck.cs

→ Dieses Script überwacht, ob eine Fähigkeit vom aktuellen Spieler benutzt wurde. Dabei wird der Fähigkeits-Button deaktiviert und das Karten-Image versteckt, sobald der Spielerwechsel stattfindet.

5.4.2.1 Funktionen

- Das Zwischenspeichern des aktuellen Spielers, wenn eine Fähigkeit angeklickt wurde (On Click Event -> „AbilityClicked“
- Das Registrieren mithilfe einer bool, wenn der zwischengespeicherte Spieler nicht mehr der jetzt aktuelle Spieler ist (-> also, ob ein Spielerwechsel stattgefunden hat)
- Nach dem Spielerwechsel wird die Interaktion mit der angeklickten Fähigkeit dauerhaft ausgeschaltet (ist nicht mehr für den Spieler verwendbar) und wird zusätzlich unsichtbar gemacht (-> Fähigkeit wurde aufgebraucht)

5.4.2.2 Properties

- /

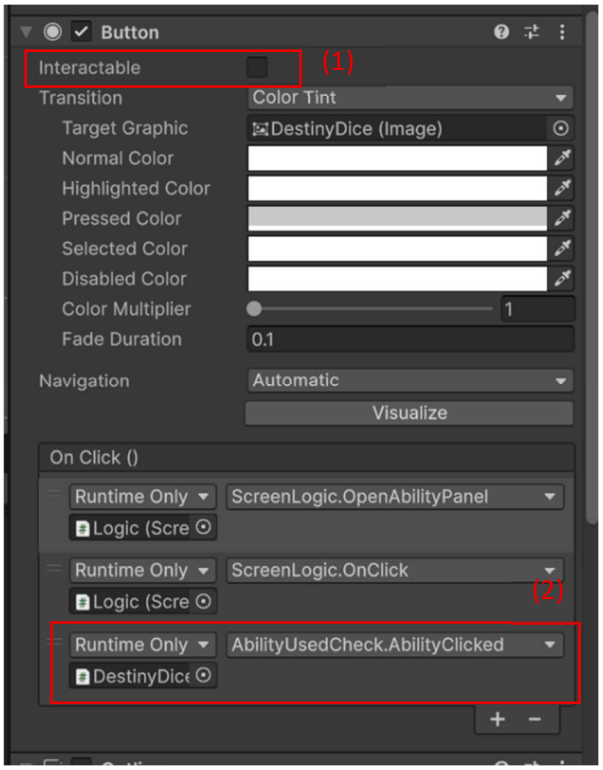
5.4.2.3 Setup

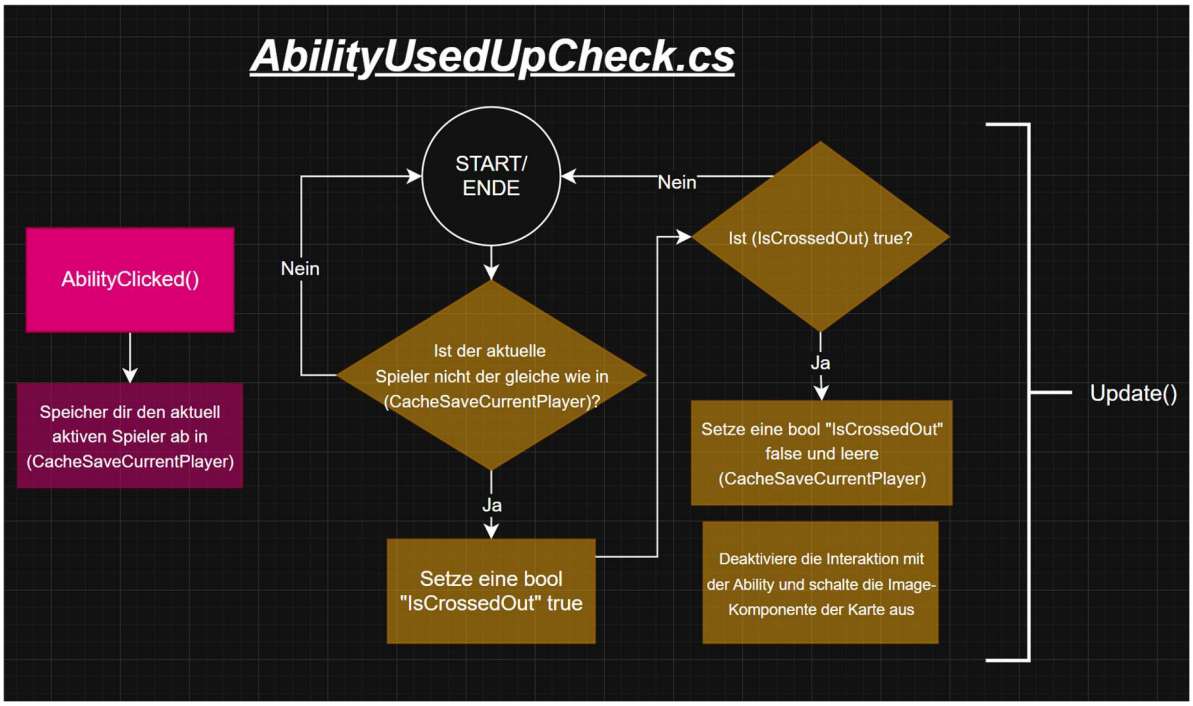
- Liegt auf: allen Fähigkeits-Buttons
- Benötigt: Button, Image

Hier bitte beachten!

➢ Button Interaktion ist ausgeschaltet (1)

➢ On Click Event einrichten:
-> Das Script „AbilityUsedUpCheck“ von dem Objekt in den linken freien Slot drag’n’droppen und danach die Funktion „AbilityClicked“ im rechten Slot auswählen (2)





5.4.3 DiceRoll.cs

→ Dieses Script steuert das Würfeln eines UI-Würfels, setzt zufällig (oder regelbasiert) den Würfelwert, aktualisiert das Sprite (Würfelbild) und erlaubt danach die Interaktion mit den Reihen.
Zusätzlich berücksichtigt es Sondermodi wie Destiny- und Thief-Würfel und speichert den gewürfelten Wert im DiceValueSaver Script.

5.4.3.1 Funktionen

- Das normale „Würfeln“ einer Zahl zwischen 1 und 6 und das Anzeigen des passenden Sprites auf dem Würfel mit RollDice()
- Das Aktivieren des Würfel-Shake Effektes (-> DiceShake.cs)
- Das Würfeln mit den ausgewählten Zahlen der Schicksal Würfel-Fähigkeit, wobei abgefragt wird welche Entscheidung der Spieler getroffen hat und die dementsprechenden Zahlen beim Würfelwurf berücksichtigt werden
- Das Deaktivieren der Button Interaktion mit den Fähigkeitskarten, sobald gewürfelt wurde
- Das Kopieren eines Würfelwertes, wenn der Spieler den Langfinger verwendet und einen gegnerischen Würfel zum Stehlen ausgewählt hat
- Das Speichern des gewürfelten Wertes auf einem „Zwischenspeicher-Script“, was ebenfalls auf dem Würfel liegt (DiceValueSaver.cs)

5.4.3.2 Properties

- /

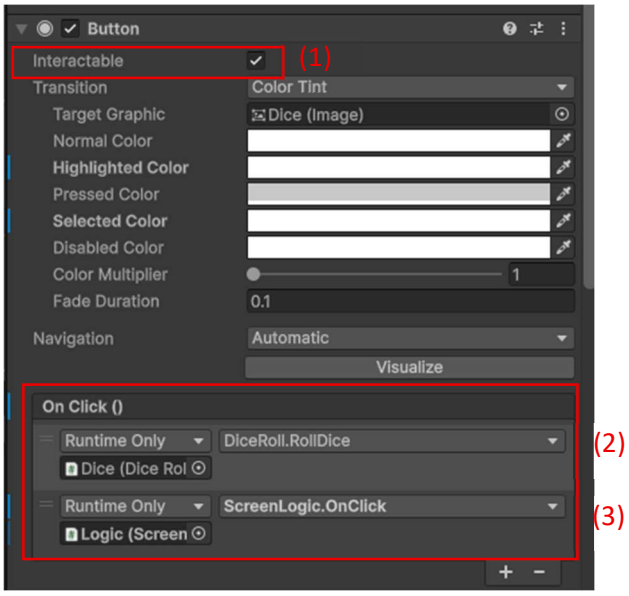
5.4.3.3 Setup

- Liegt auf: dem zu würfelnden Würfel („Dice“)
- Benötigt: Image (UI Component), Button

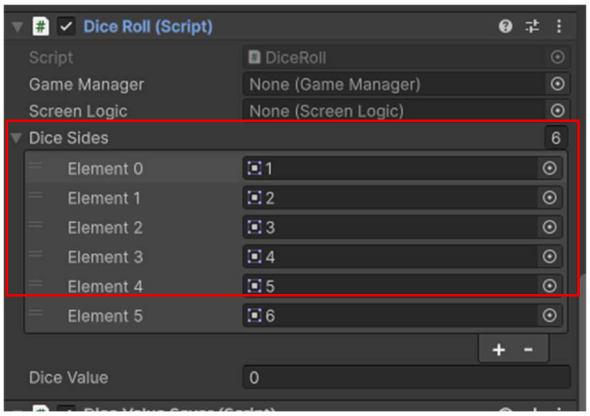
Hier bitte beachten!

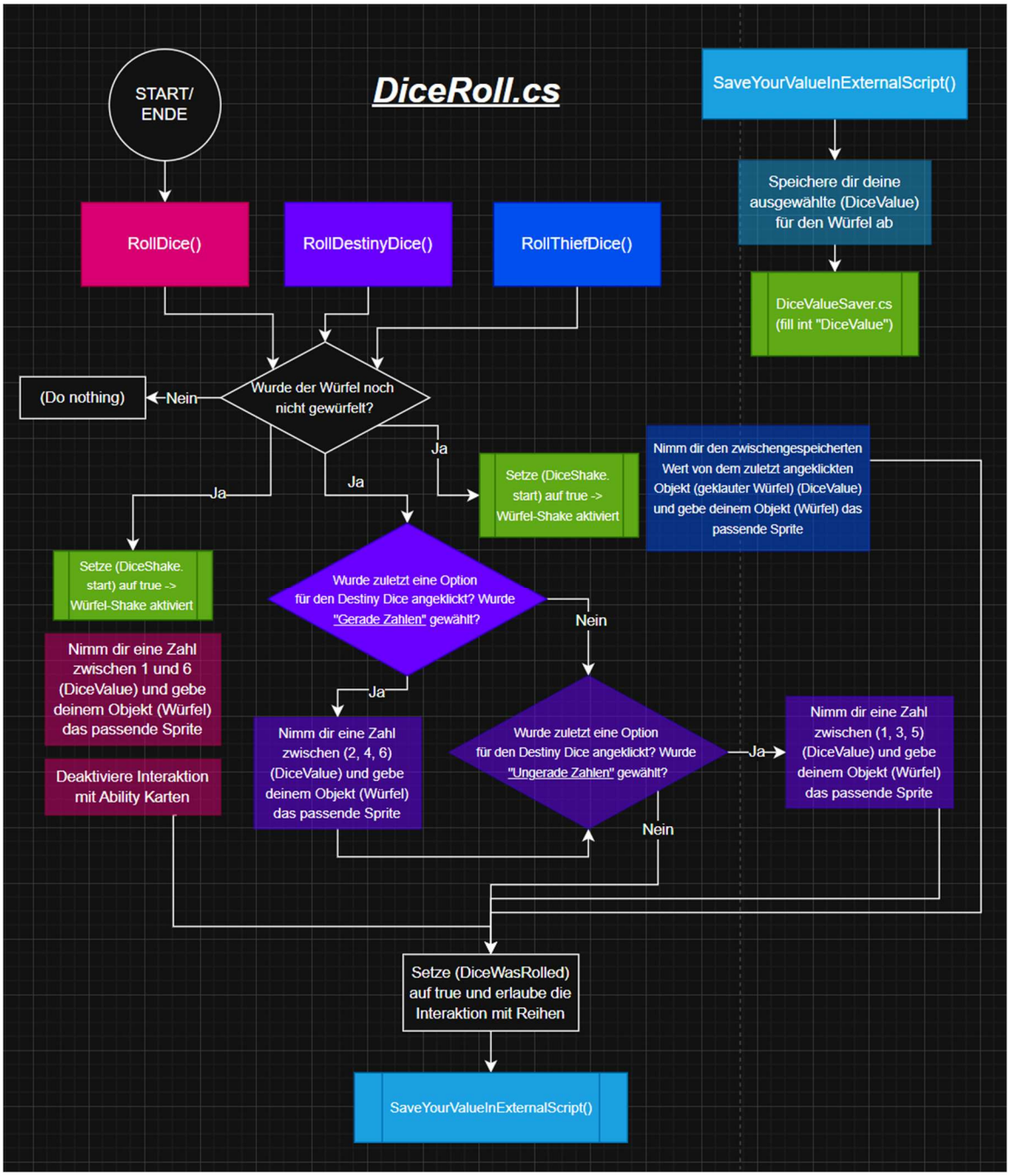
➢ Button Interaktion ist angeschaltet (1)

➢ On Click Event einrichten:
-> Das Script „DiceRoll“ von dem Objekt in den linken freien Slot drag’n’droppen und danach die Funktion „RollDice“ im rechten Slot auswählen (2)
-> Das Script „ScreenLogics“ von dem GameObject „Logic“ in den linken freien Slot drag’n’droppen und danach die Funktion „OnClick“ im rechten Slot auswählen (3)



- Auf dem Script das Array „Dice Sides“ mit den Sprite-Nummern 1-6 füllen (zu finden unter dem Ordner Assets > Sprites)





5.4.4 DiceRowsLogic.cs

→ Das Script verwaltet die Logik für eine einzelne Würfelreihe im Spiel: Es überprüft ständig, ob die Reihe voll ist, ermöglicht das Platzieren von Würfeln oder Schilden, aktualisiert die Punkte über das PointSystem Script und steuert die Interaktion mit der Reihe selbst, basierend auf Spielmodus, aktuellem Spieler und dem Status der Reihe (gefüllte Felder). Zusätzlich sorgt es dafür, dass beim Platzieren von Würfeln die Reihen korrekt aktualisiert,

die Reihenauswahl an den richtigen Stellen deaktiviert und die Würfel visuell im UI richtig positioniert werden.

5.4.4.1 Funktionen

- Keine Interaktion mit Reihen wenn das Quit Panel geöffnet
- Die Normale Reihen Interaktion wenn der Spieler gewürfelt oder die Schild-Ability ausgewählt hat (dabei wird die mittlere Reihe immer automatisch als Erst-Auswahl festgelegt)
- Das Überprüfen ob die Reihe vollständig besetzt ist um eine Würfelüberschreibung zu verhindern und um rechtzeitig zu erkennen, wann ein Spieler keinen Würfel mehr setzen kann (-> Siegerkührung)
- Das Setzen eines Schildes auf die angeklickte Reihe, wenn die Schild-Aktion verwendet wird und den darauf folgenden Spielerwechsel
- Das Setzen eines Duplikats von dem gewürfelten Würfel in die angeklickte Reihe in das nächst freie Feld, danach wird dem Würfel-Duplikat die Button-Interaktion und das DiceRoll.cs entzogen, Punkte neu verrechnet (Pointsystem.CheckDicePoints()) und der Spieler gewechselt

5.4.4.2 Properties

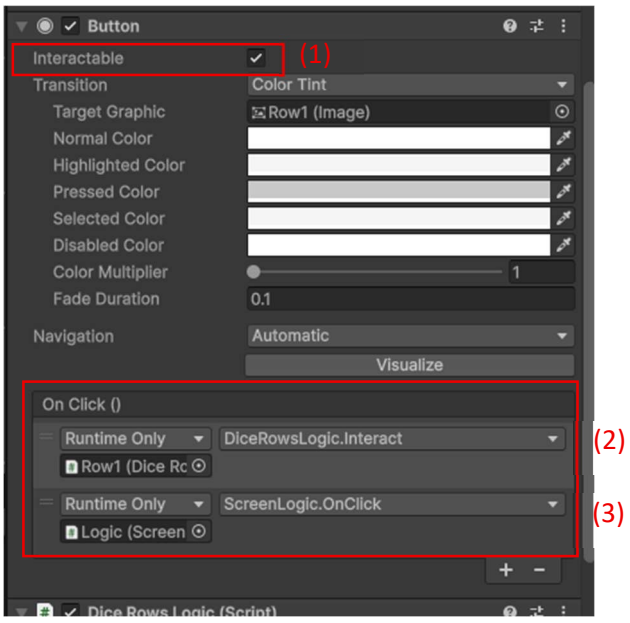
- /

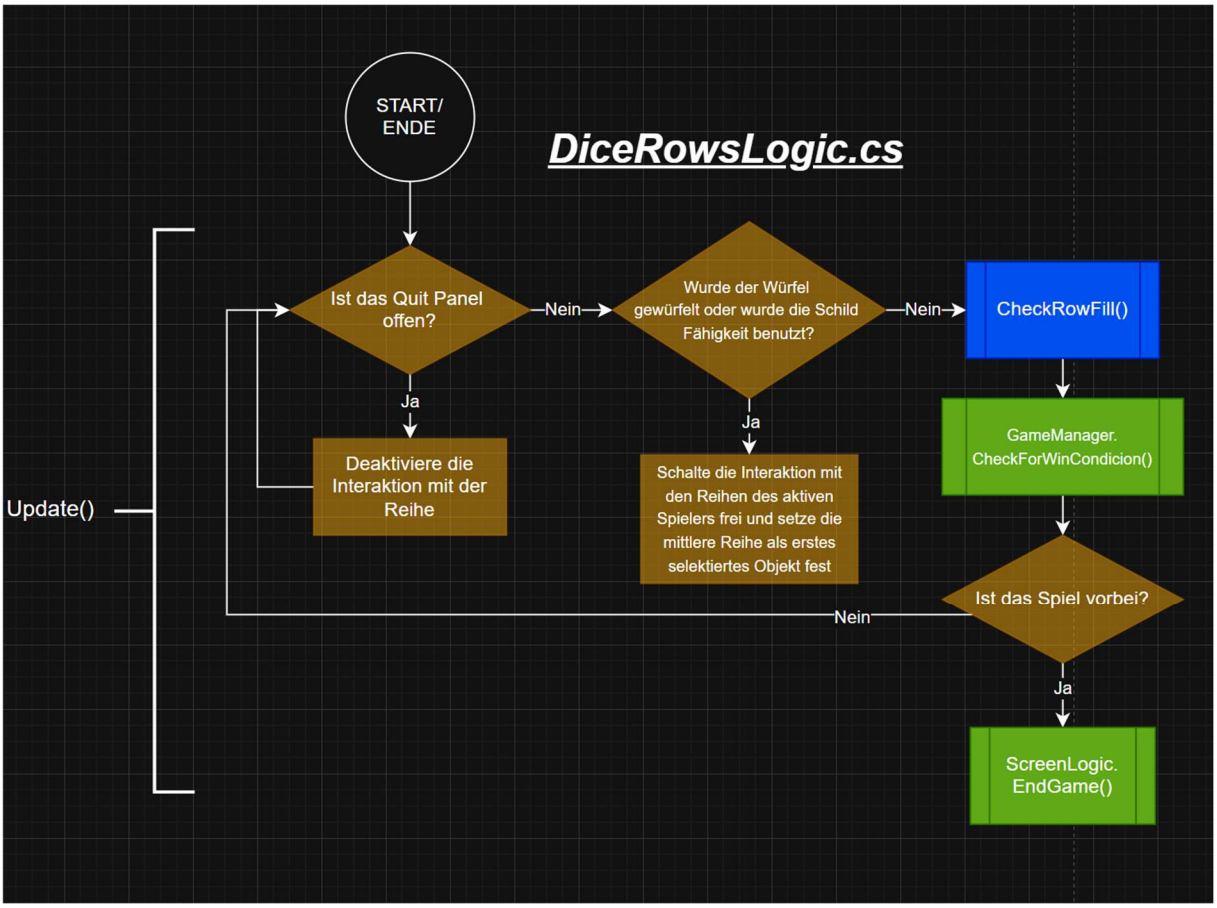
5.4.4.3 Setup

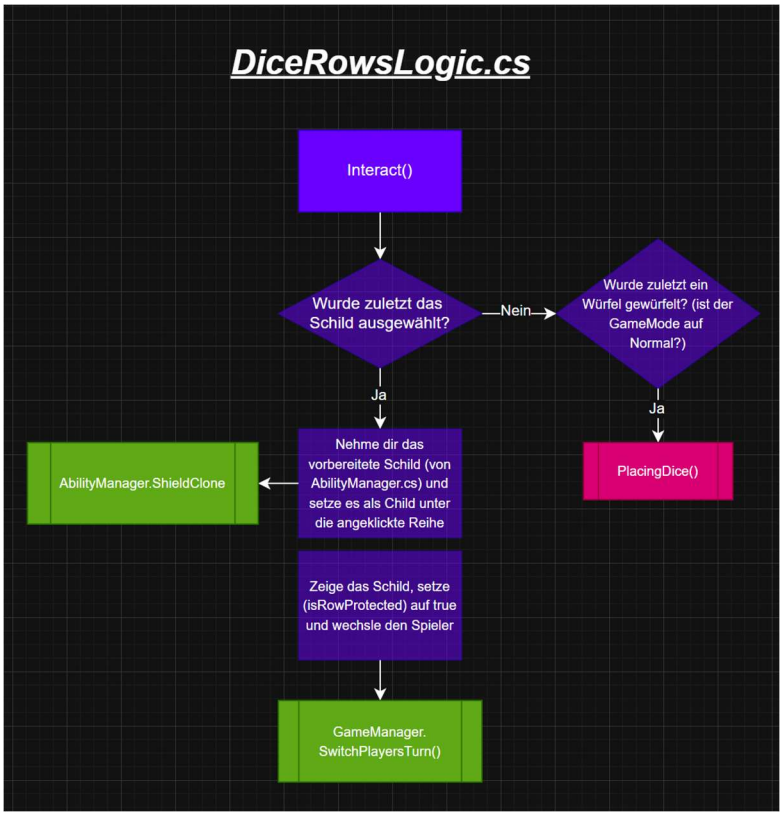
- Liegt auf: allen GameObjects mit dem Namen „Row“ (unter „Player1“ und „Player2“)
- Benötigt: Button

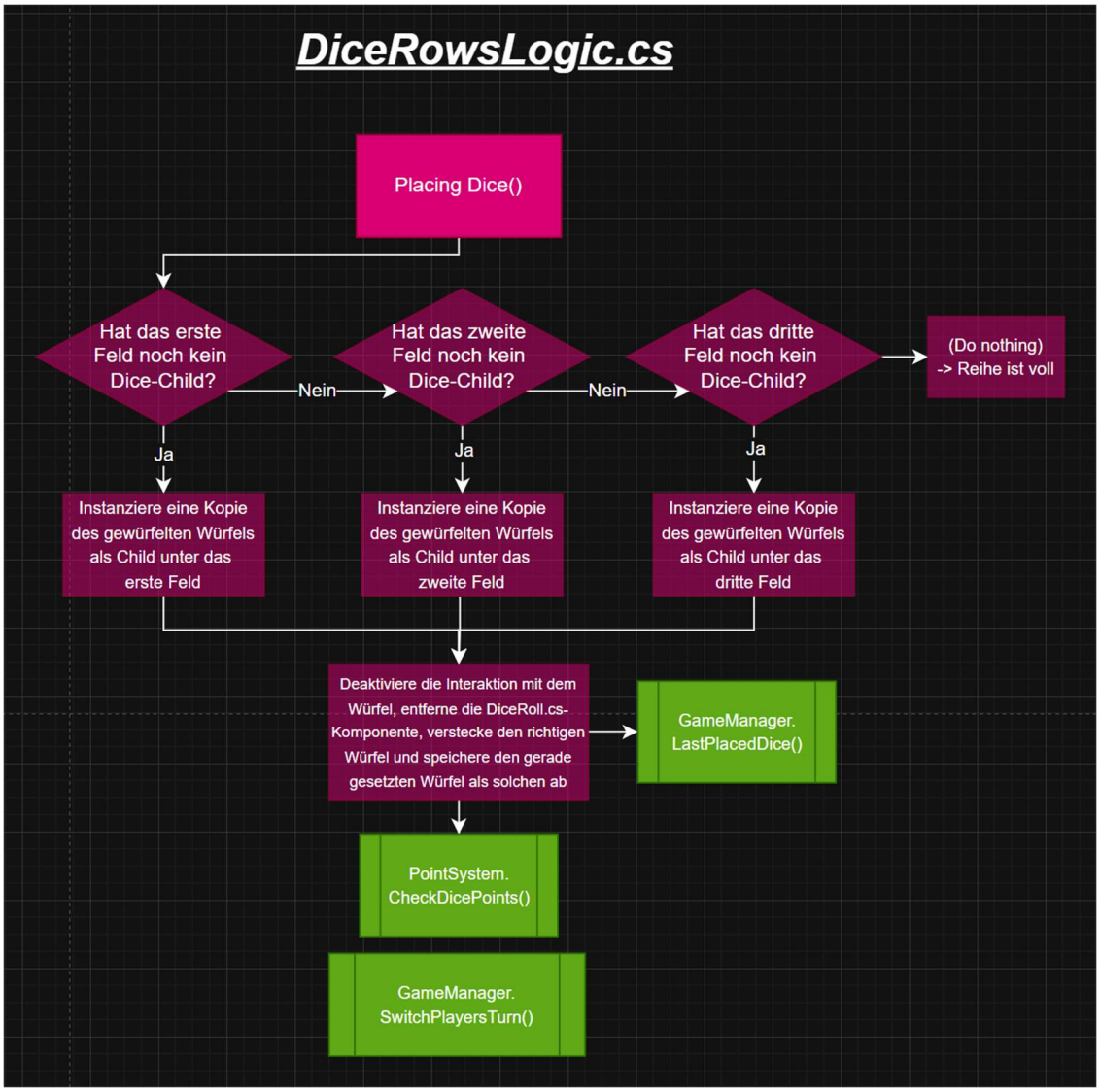
Hier bitte beachten!

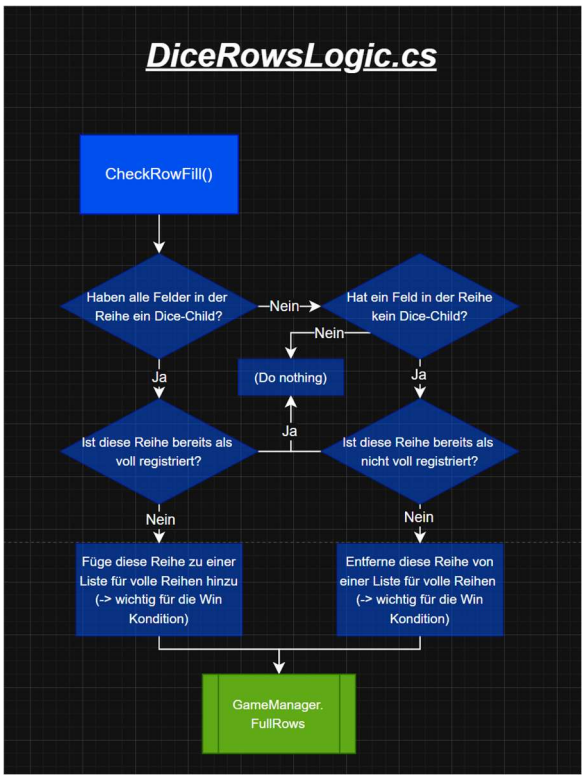
- Button Interaktion ist angeschaltet (1)
- On Click Event einrichten:
  -> Das Script „DiceRow“ <u>von dem jeweiligen Objekt</u>! („Row1“ benötigt das Script von diesem „Row1“) in den linken freien Slot drag’n’droppen und danach die Funktion „Interact“ im rechten Slot auswählen (2)
  -> Das Script „ScreenLogics“ von dem GameObject „Logic“ in den linken freien Slot drag’n’droppen und danach die Funktion „OnClick“ im rechten Slot auswählen (3)




START/
ENDE
DiceRowsLogic.cs
Ist das Quit Panel offen?
Nein
Wurde der Würfel gewürfelt oder wurde die Schild Fähigkeit benutzt?
Nein
CheckRowFill()
Ja
Deaktiviere die Interaktion mit der Reihe
Ja
Schalte die Interaktion mit den Reihen des aktiven Spielers frei und setze die mittlere Reihe als erstes selektiertes Objekt fest
GameManager.
CheckForWinCondicion()
Update()
Ist das Spiel vorbei?
Nein
Ja
ScreenLogic.
EndGame()


DiceRowsLogic.cs
Interact()
Wurde zuletzt das Schild ausgewählt?
Nein
Wurde zuletzt ein Würfel gewürfelt? (ist der GameMode auf Normal?)
Ja
Ja
AbilityManager.ShieldClone
Nehme dir das vorbereitete Schild (von AbilityManager.cs) und setze es als Child unter die angeklickte Reihe
PlacingDice()
Zeige das Schild, setze (isRowProtected) auf true und wechsle den Spieler
GameManager.
SwitchPlayersTurn()


DiceRowsLogic.cs
Placing Dice()
Hat das erste Feld noch kein Dice-Child?
Nein
Hat das zweite Feld noch kein Dice-Child?
Nein
Hat das dritte Feld noch kein Dice-Child?
(Do nothing) -> Reihe ist voll
Ja
Ja
Ja
Instanziere eine Kopie des gewürfelten Würfels als Child unter das erste Feld
Instanziere eine Kopie des gewürfelten Würfels als Child unter das zweite Feld
Instanziere eine Kopie des gewürfelten Würfels als Child unter das dritte Feld
Deaktiviere die Interaktion mit dem Würfel, entferne die DiceRoll.cs-Komponente, verstecke den richtigen Würfel und speichere den gerade gesetzten Würfel als solchen ab
GameManager. LastPlacedDice()
PointSystem. CheckDicePoints()
GameManager. SwitchPlayersTurn()


DiceRowsLogic.cs
CheckRowFill()
Haben alle Felder in der Reihe ein Dice-Child?
Nein
Hat ein Feld in der Reihe kein Dice-Child?
Nein
Ja
(Do nothing)
Ja
Ja
Ist diese Reihe bereits als voll registriert?
Ist diese Reihe bereits als nicht voll registriert?
Nein
Nein
Füge diese Reihe zu einer Liste für volle Reihen hinzu (-> wichtig für die Win Kondition)
Entferne diese Reihe von einer Liste für volle Reihen (-> wichtig für die Win Kondition)
GameManager. FullRows

5.4.5 DiceValueSaver.cs

→ Dieses Script dient als einfacher Speicher für den Wert eines Würfels; jede Würfel-Instanz hält hier ihre aktuelle Zahl.

5.4.5.1 Funktionen

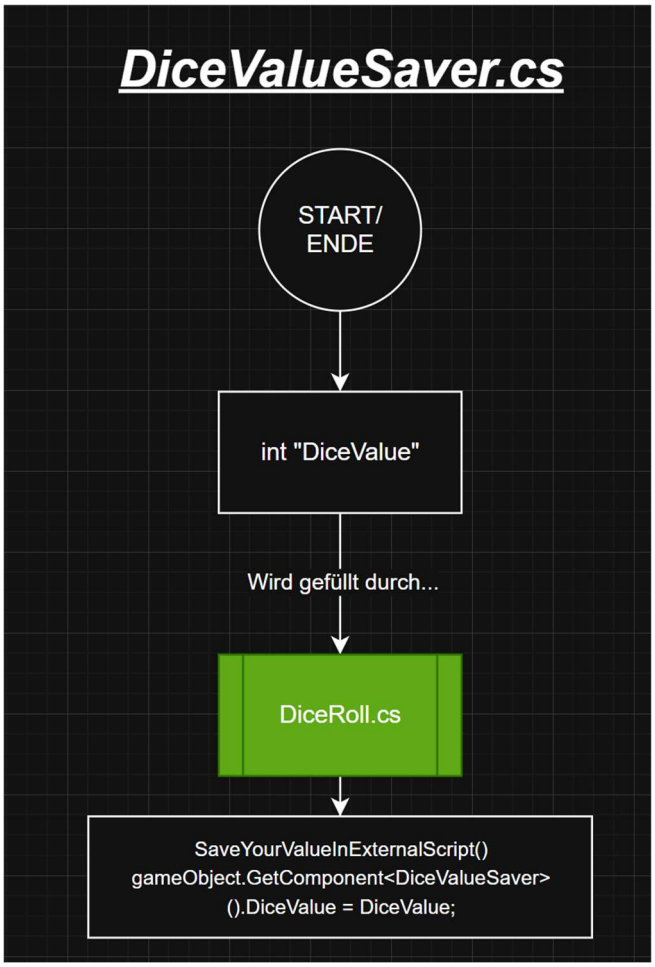
- /
- (-> Dieses Script enthält ausschließlich eine leere Value vom Typ int, die später gefüllt wird)

5.4.5.2 Properties

- /

5.4.5.3 Setup

- Liegt auf: GameObject „Dice“
- Benötigt: (nichts, bis auf DiceRoll.cs)



5.4.6 PointSystem.cs

→ Dieses Script ist für die Berechnung und Aktualisierung der Punkte der letzten beeinflussten Reihe und der Gegenüberliegenden zuständig. Dabei werden Doppel-, Dreifach-, Diagonal- und Exodia Kombinationen erkannt und farblich markiert sowie Interaktion zwischen eigenen und gegnerischen Würfeln berücksichtigt, inklusive Schutzmechanismus durch Schilde.

5.4.6.1 Funktionen

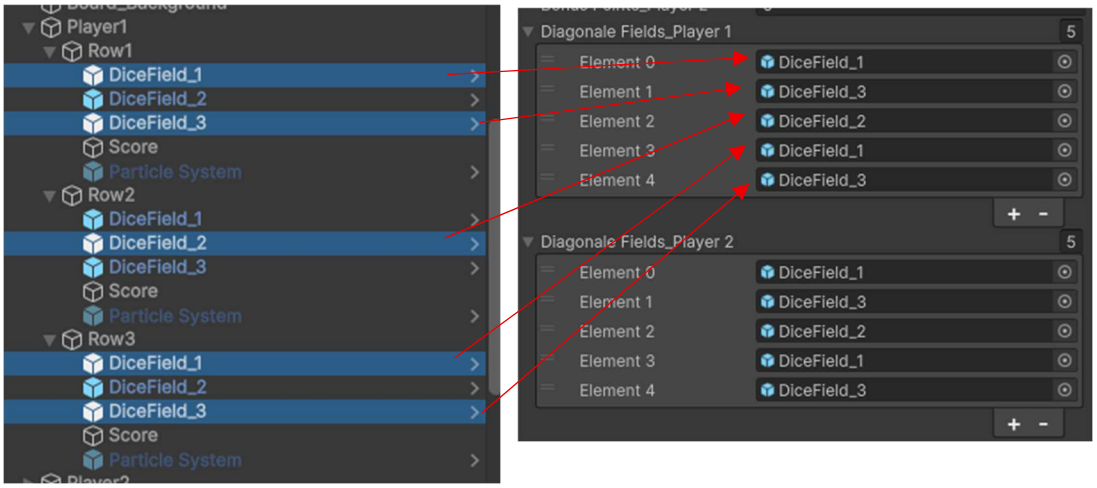
- Das Zurücksetzen aller farblichen Markierungen und Effekte auf den Würfeln, worauf dann eine neue Prüfung nach Triples (drei gleiche), Doubles (zwei gleiche) und Diagonales (drei gleiche in einer Diagonalen) folgt (-> weniger Fehler lastig)
- Danach Punkteverrechnung der eigenen sowie der gegnerischen Reihe mit Beachtung von speziellen Kombinationen + farbliche Hervorhebung (Normaler Modus -> nur die letzte angewählte Reihe, Twister-Modus -> Alle Reihen, die betroffen sind)
- Das Vergleichen der eigenen Würfelwerte mit denen des Gegners und gegebenenfalls Entfernen der gegnerischen Würfel, sofern keine Schild-Schutzwirkung aktiv ist
- Das Entfernen von Würfeln aus einer Reihe oder das gezielte Entfernen des zuletzt gesetzten Würfels, wenn eine Schutzreihe existiert, um Angriffe abzuwehren
- Das Anzeigen der aktuellen Punktzahl jeder Reihe und der Gesamtsumme auf dem UI jedes Spielers

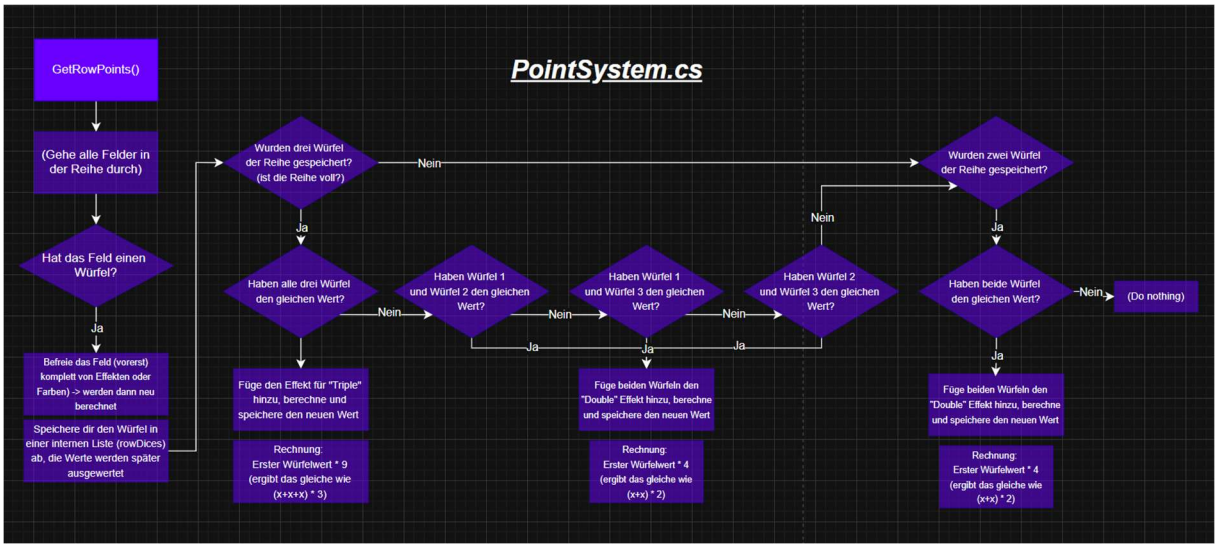
5.4.6.2 Properties

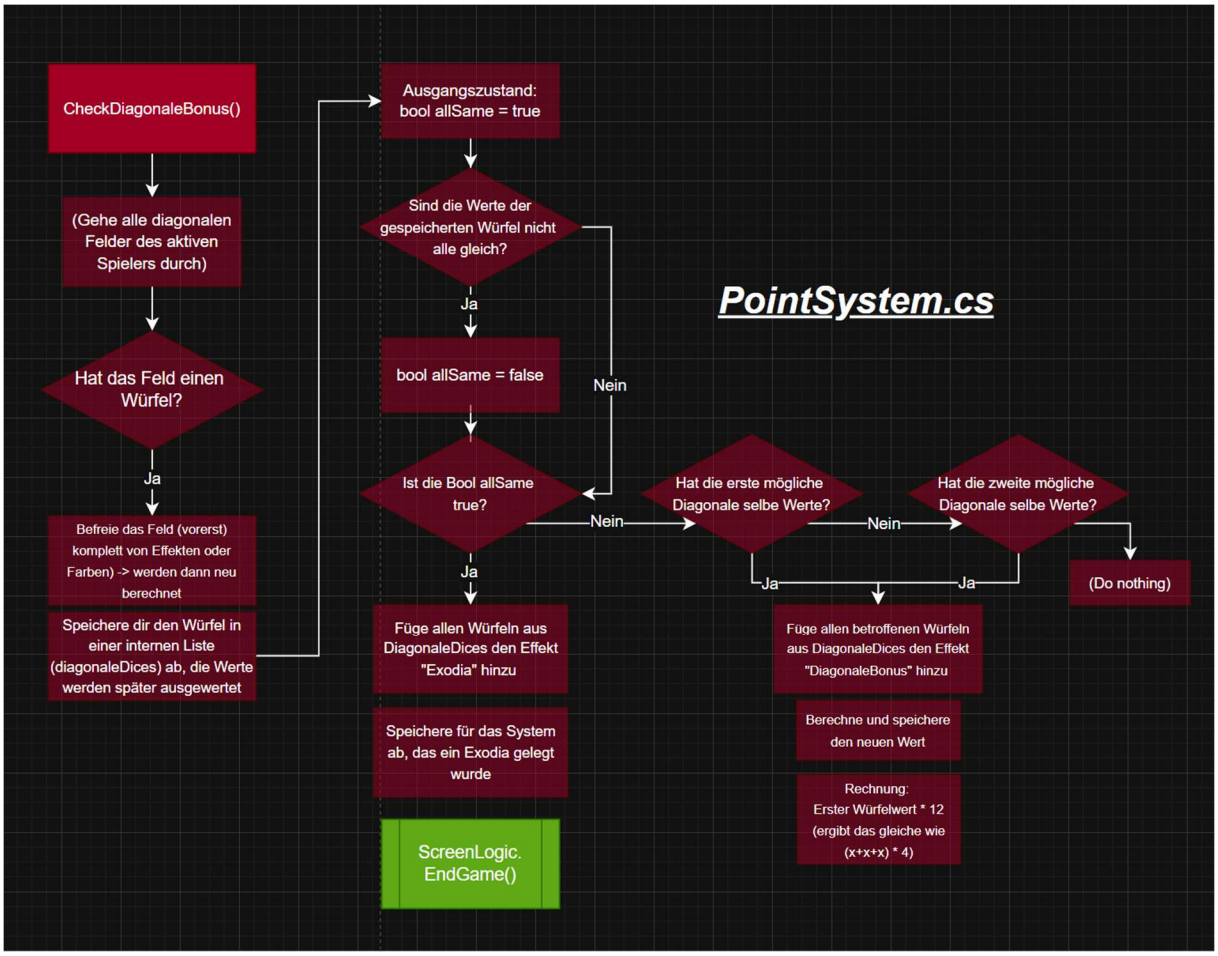
- /

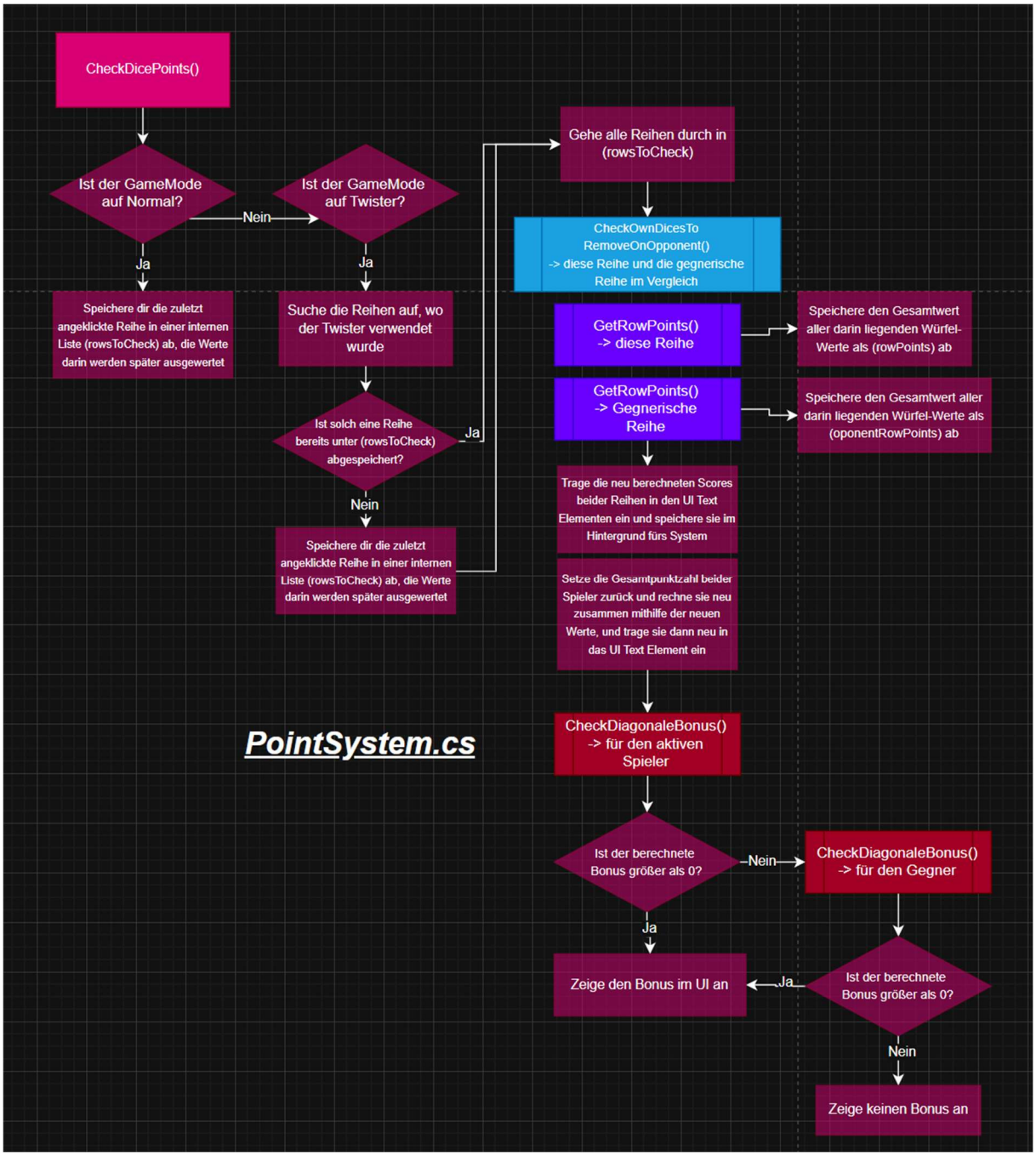
5.4.6.3 Setup

- Liegt auf: GameObject „Logic“
- Benötigt: /
- Einrichten der „Diagonale Fields“:
    - Bitte folgendes GENAU beachten!
    - In die beiden Arrays „Diagonale Fields_Player1“ und „Diagonale Fields_Player2“ gehören die Felder jeweils beider Spieler, die eine Diagonale bzw. ein Exodia bilden können, also:
        - Feld links oben
        - Feld links unten
        - Feld mitte mitte
        - Feld rechts oben
        - Feld rechts unten
    - In genau dieser DIESER Reihenfolge müssen beide Arrays gefüllt werden (da diese Reihenfolge entscheidend für Handlungen im Script sind!)




PointSystem.cs
GetRowPoints()
(Gehe alle Felder in der Reihe durch)
Hat das Feld einen Würfel?
Ja
Befreie das Feld (vorerst) komplett von Effekten oder Farben) -> werden dann neu berechnet
Speichere dir den Würfel in einer internen Liste (rowDices) ab, die Werte werden später ausgewertet
Wurden drei Würfel der Reihe gespeichert? (ist die Reihe voll?)
Nein
Ja
Haben alle drei Würfel den gleichen Wert?
Nein
Füge den Effekt für "Triple" hinzu, berechne und speichere den neuen Wert
Rechnung: Erster Würfelwert * 9 (ergibt das gleiche wie (x+x+x) * 3)
Haben Würfel 1 und Würfel 2 den gleichen Wert?
Nein
Ja
Haben Würfel 1 und Würfel 3 den gleichen Wert?
Nein
Ja
Füge beiden Würfeln den "Double" Effekt hinzu, berechne und speichere den neuen Wert
Rechnung: Erster Würfelwert * 4 (ergibt das gleiche wie (x+x) * 2)
Haben Würfel 2 und Würfel 3 den gleichen Wert?
Nein
Ja
Wurden zwei Würfel der Reihe gespeichert?
Nein
Ja
Haben beide Würfel den gleichen Wert?
Nein
(Do nothing)
Ja
Füge beiden Würfeln den "Double" Effekt hinzu, berechne und speichere den neuen Wert
Rechnung: Erster Würfelwert * 4 (ergibt das gleiche wie (x+x) * 2)


PointSystem.cs
CheckDiagonaleBonus()
(Gehe alle diagonalen Felder des aktiven Spielers durch)
Hat das Feld einen Würfel?
Ja
Befreie das Feld (vorerst) komplett von Effekten oder Farben) -> werden dann neu berechnet
Speichere dir den Würfel in einer internen Liste (diagonaleDices) ab, die Werte werden später ausgewertet
Ausgangszustand: bool allSame = true
Sind die Werte der gespeicherten Würfel nicht alle gleich?
Ja
bool allSame = false
Nein
Ist die Bool allSame true?
Nein
Ja
Füge allen Würfeln aus DiagonaleDices den Effekt "Exodia" hinzu
Speichere für das System ab, das ein Exodia gelegt wurde
ScreenLogic. EndGame()
Hat die erste mögliche Diagonale selbe Werte?
Nein
Ja
Hat die zweite mögliche Diagonale selbe Werte?
Nein
Ja
(Do nothing)
Füge allen betroffenen Würfeln aus DiagonaleDices den Effekt "DiagonaleBonus" hinzu
Berechne und speichere den neuen Wert
Rechnung: Erster Würfelwert * 12 (ergibt das gleiche wie (x+x+x) * 4)


CheckDicePoints()
Ist der GameMode auf Normal?
Nein
Ist der GameMode auf Twister?
Ja
Ja
Speichere dir die zuletzt angeklickte Reihe in einer internen Liste (rowsToCheck) ab, die Werte darin werden später ausgewertet
Suche die Reihen auf, wo der Twister verwendet wurde
Ist solch eine Reihe bereits unter (rowsToCheck) abgespeichert?
Ja
Nein
Speichere dir die zuletzt angeklickte Reihe in einer internen Liste (rowsToCheck) ab, die Werte darin werden später ausgewertet
Gehe alle Reihen durch in (rowsToCheck)
CheckOwnDicesTo RemoveOnOpponent() -> diese Reihe und die gegnerische Reihe im Vergleich
GetRowPoints() -> diese Reihe
Speichere den Gesamtwert aller darin liegenden Würfel-Werte als (rowPoints) ab
GetRowPoints() -> Gegnerische Reihe
Speichere den Gesamtwert aller darin liegenden Würfel-Werte als (oponentRowPoints) ab
Trage die neu berechneten Scores beider Reihen in den UI Text Elementen ein und speichere sie im Hintergrund fürs System
Setze die Gesamtpunktzahl beider Spieler zurück und rechne sie neu zusammen mithilfe der neuen Werte, und trage sie dann neu in das UI Text Element ein
PointSystem.cs
CheckDiagonaleBonus() -> für den aktiven Spieler
Ist der berechnete Bonus größer als 0?
Nein
CheckDiagonaleBonus() -> für den Gegner
Ja
Zeige den Bonus im UI an
Ja
Ist der berechnete Bonus größer als 0?
Nein
Zeige keinen Bonus an


RemoveDice()
Wurden alle Gegner-Würfel geprüft?
Nein
Gehe alle Felder in der Reihe durch
Hat dieses Feld einen Würfel?
Ja
Wurde im letzten Durchgang ein Würfel gelöscht?
Nein
Stimmt dieser Würfelwert mit dem des aktiven Spielers überein?
Ja
Nein
Lagere den Würfel auf ein unischtbares GameObject (DestroyerParent) aus und zerstöre daraufhin den Würfel
(Das Auslagern ist notwendig, da das System von Unity nicht so schnell mitbekommt wenn ein GameObject destroyed wurde und wir aber diese Prüfung sofort brauchen)
Trägt die gegnerische Reihe ein Schild?
Ja
RemoveLastPlacedDice()
Ja
Setze diesen Würfel auf das Feld des zerstörten Würfels
Liegt in dieser Reihe noch ein weiterer Würfel?
Ja
Setze den zweiten Würfel auf das gerade frei gewordene Feld
PointSystem.cs
CheckOwnDicesTo RemoveOnOpponent()
Gehe alle Würfel in der eigenen Reihe durch und vergleiche dessen Wert mit der gegnerischen Reihe
RemoveDice()
RemoveLastPlacedDice()
Zerstöre den Würfel, der zum Zerstören ermittelt wurde

5.4.7 ShieldPower.cs

→ Dieses Script verwaltet ein Schild, der eine Reihe schützt und beim Spielerwechsel nach jedem Zug an Wert verliert. Sobald das Schild aufgebraucht ist, wird es aus dem Spiel entfernt und verliert seine Wirkung.

5.4.7.1 Funktionen

- Das Initialisieren eines Schildes mit einem Maximalwert von 3 und einem Zähler für Spielerwechsel, um die Dauer der Schildwirkung zu verfolgen
- Das Hinzufügen des Schildes zur zentralen Liste im GameManager zur späteren Verwaltung
- Das Reduzieren des Schildwertes um 1, wenn der Gegner seinen Zug beendet hat

5.4.7.2 Properties

- /

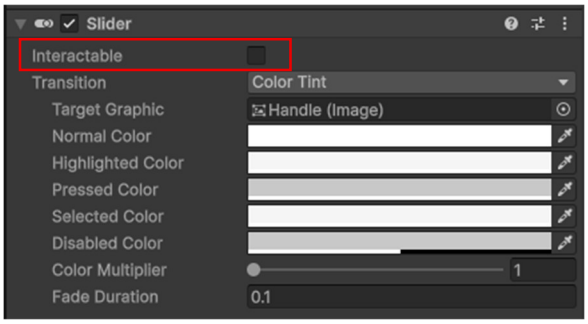
5.4.7.3 Setup

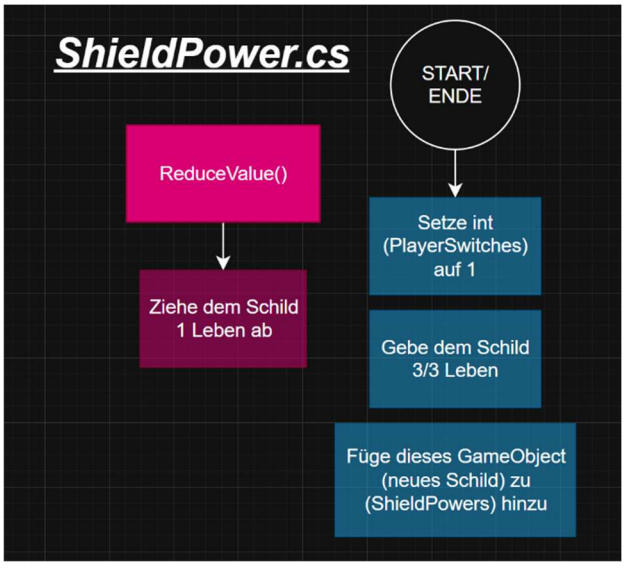
- Liegt auf: GameObject „Shield“

  Hier bitte beachten!
  ➢ Dieses GameObject muss unter dem Ordner „Resources“ liegen!
  (Assets > Resources)

- Benötigt: Slider

  Hier bitte beachten!
  ➢ Der Slider darf KEINEN Haken bei „Interactable“ haben! (siehe Bild)





5.5 VISUAL EFFECTS (NTHS)

5.5.1 BackgroundHueShift.cs

→ Dieses Script sorgt dafür, dass das eingestellte Image im Hintergrund einen leichten Farbwechsel anzeigt.

5.5.1.1 Funktionen

- Das Hin- und Herspringen zwischen zwei einstellbare Farben in einer bestimmten Geschwindigkeit und mit smoother Farbänderung

5.5.1.2 Properties

- Color A: Die erste Farbe, die mit einem Farbregler eingestellt werden kann (default setting: FFFFFF, Transparenz 42)
- Color B: Die zweite Farbe, die mit einem Farbregler eingestellt werden kann (default setting: CF80B2, Transparenz 42)
- Speed: Die Geschwindigkeit, in der der Farbwechsel stattfinden soll (default setting: 0.2)

5.5.1.3 Setup

- Liegt auf: Hintergrund GameObject
- Benötigt: Image

5.5.2 BlinkingEffect.cs

→ Dieses Script sorgt für den Blink-Effekt auf den Würfelfeldern sowohl beim Erschaffen einer Diagonalen Würfelreihe, als auch bei einem Exodia.

5.5.2.1 Funktionen

- Das Hin- und Herwechseln von volle Farbe zu leichter Transparenz in einer bestimmten Geschwindigkeit (was den Blick-Effekt erzeugt)
- Das Farbewechseln im Regenbogen, wenn ein Exodia ausgelöst wurde
- Das Zurücksetzen der Farbe, wenn eine bool „isAllowedToBlink“ nicht mehr true ist

5.5.2.2 Properties

- Speed_Diagonale Bonus: Die Geschwindigkeit, in der der Transparente Blick-Effekt laufen soll (default setting: 2f)
- Speed_Exodia: Die Geschwindigkeit, in der der Regenbogen-Effekt laufen soll (default speed: 0.3f)

5.5.2.3 Setup

- Liegt auf: Alle Betroffenen Würfelfelder die ein Exodia bilden können (links oben, links unten, mitte mitte, rechts oben, rechts unten)
- Benötigt: Image

5.5.3 DiceShake.cs

→ Dieses Script ist dafür verantwortlich, das der Würfel eine kleine „Würfelanimation“ durchläuft, wenn dieser gewürfelt wird.

5.5.3.1 Funktionen

- Das random Schütteln des Würfels in einer bestimmten Stärke und für eine bestimmte Zeit
- Das durchlaufen Random Zahlen von 1 bis 6 auf dem Würfel, während des Schüttelns
- Das Zurücksetzen der Position und das anzeigen des (im Hintergrund) gewürfelten Wertes (-> aus DiceRoll.DiceValue!)

5.5.3.2 Properties

- Duration: Die Zeit, wie lang der Würfel geschüttelt werden soll
- Strength: Die Stärke, in der der Würfel geschüttelt werden soll

5.5.3.3 Setup

- Liegt auf: GameObject „Dice“
- Benötigt: Image

5.5.4 FlippingCard.cs

→ Dieses Script sorgt dafür, das Ability Cards umgedreht werden.

5.5.4.1 Funktionen

- Das Rotieren des GameObjects auf der Y-Achse in einer bestimmten Geschwindigkeit
- Das Wechseln des Sprites der Karte, sobald die Rotation 90° erreicht hat
- Das Stoppen der Rotation, sobald eine Rotation von 180° erreicht wurde + das zurücksetzen der Rotation auf 0

5.5.4.2 Properties

- Flip Speed: Die Geschwindigkeit, in der die Karte gedreht werden soll

5.5.4.3 Setup

- Liegt auf: Alle Ability-Cards GameObjects
- Benötigt: Image
- Der leere Sprite-Verweis „Front Sprite“ muss mit der jeweiligen Vorderseite der Karte gefüllt werden (im Ordner „Sprites“/“Cards“)
- Der leere Sprite-Verweis „Back Sprite“ muss mit der Rückseite der Karten gefüllt werden (ebenfalls unter „Sprites“/“Cards“)

5.5.5 UIMovement.cs

→ Dieses Script sorgt dafür, das sich die Ability Cards nach oben und unten bewegen, je nachdem ob sie selektiert sind oder nicht.

5.5.5.1 Funktionen

- Das hochbewegen von einer Start Position zu einer höheren Position mit festgelegtem Abstand, smoother Bewegung und einer bestimmten Schnelligkeit (MoveUp())
- Das runterbewegen von einer Start Position zu der neuen Zielposition (von MoveUp()) in smoother Bewegung und einer bestimmten Schnelligkeit (MoveDown())
- Das Setzen der Zielposition zu der Start Position für die nächste Routine

5.5.5.2 Properties

- Move Distance: Die Strecke, die zwischen „Karte unten“ und „Karte oben“ liegt (default setting: 10)
- Duration: Die Schnelligkeit, in der diese Bewegung ausgeführt werden soll (default setting: 0.05f)

5.5.5.3 Setup

- Liegt auf: Alle Ability-Cards GameObjects
- Benötigt: Rect Transform (Transform Komponente von UI)

6 QUELLEN

Hier sind alle verwendeten Fremd Assets und KI-Prompts aufgelistet, die zur Gestaltung des Spiels beigetragen haben. Nicht aufgeführte Grafiken wurden dementsprechend selber mit Adobe Photoshop erstellt.

6.1 GRAFIKEN

6.1.1 Bilder/Grafiken Quellen

6.1.1.1 UI-Grafiken

https://static.thenounproject.com/png/1067328-200.png
https://opengameart.org/sites/default/files/psControllerShareOptionsHome.png

6.1.1.2 Text

https://www.1001fonts.com/gnf-font.html

6.1.1.3 Spieler Figuren

https://i.pinimg.com/736x/0f/d7/5e/0fd75eb3074ce851127dc96cb4dd8f65.jpg (geändert in Photoshop)

6.1.1.4 (Hintergrund Bild)

https://i.pinimg.com/1200x/15/79/a2/1579a2c7255d02a622fac9ceeb45026a.jpg

6.1.2 KI-generierte Bilder/Grafiken

Alle hier aufgeführten Bilder wurden mit dem Image-Generator von ChatGPT selber generiert und dann später angepasst (siehe Coding und Engine Lexika > Photoshop Anpassungen)

Fähigkeitskarten-Rand (und Stilfindung für die Karten danach)

https://chatgpt.com/backend-api/estuary/content?id=file_000000007eb471f489d28ae2290b9632&ts=491385&p=fs&cid=1&sig=0968761a561e957f6c6d8ee719c3dbfbdc8987c79a0631a0a77b834a36acd378&v=0

Prompt 1: „Ich habe in ein eigenes Videospiel im Retro Pixel Art Neon Style neue Spielelemente eingebaut, u.a. mit Spielkarten, die aus einem Deck gezogen werden können. Kannst du mir Ability Cards mit einem groben Layout und Farben, passend zu dem beschriebenen Stil, generieren?“

Daraufhin hat er mir mehrere Karten auf einer Hand generiert, mir gefiel der Style aber noch nicht.

Prompt 2: „Könntest du den style anpassen an die Bilder?“

(Im Anhang: https://i.pinimg.com/736x/b4/6d/e6/b46de6f45dc9d3e80ad675fe654d7487.jpg

https://i.pinimg.com/736x/35/d4/62/35d462cf2109c2a0720050728d953ec3.jpg

https://i.pinimg.com/736x/61/0b/03/610b030e4f59b9b0f02f86fb7ba2c4e1.jpg)

Karten haben mir gefallen, aber ich wollte nur eine als Layout haben

Prompt 3: „Kannst du mir die blaue Karte als Preset nochmal einzeln und in groß geben?“

Schicksalswürfel Symbolik

https://chatgpt.com/backend-api/estuary/content?id=file_000000006d48720abfb7b53e588b5f55&ts=491386&p=fs&cid=1&sig=3643a7d4c51fa84437b2ddfa87becba80e62e79d92983941e0180082ac8179b7&v=0

Prompt: „Kannst du mir eine Pixel Grafik in einem ähnlichen Retro Neon Stil generieren, die zwei Würfel zeigt, der eine mit geraden Zahlen, der andere mit Ungeraden“

Langfinger-Würfel Symbolik

https://chatgpt.com/backend-api/estuary/content?id=file_000000002c48724687f71590a2a256cb&ts=491386&p=fs&cid=1&sig=1f6503884f0f307f156ff4d254c97ffd6ca4a0a5366629514e464274d77c23a2&v=0

Prompt: „Jetzt generiere mir bitte ein grünes Schild“

Schild Symbolik

https://chatgpt.com/backend-api/estuary/content?id=file_0000000012c071f4bc918bd998fd44bd&ts=491386&p=fs&cid=1&sig=c90a24d89b8014cd5c41d744194475a38c713fea4ae7f3af6f6fe2a00b298a70&v=0

Prompt: „Jetzt generiere mir bitte ein Symbol für "Stehlen" oder „Dieb“

Twister Symbolik

https://chatgpt.com/backend-api/estuary/content?id=file_00000000f760720ab0ef2cf1dfc137c5&ts=491385&p=fs&cid=1&sig=fbcce2ff4af8384ac7314655b6adccf528f08a39dbdf603bf775f640adbd3371&v=0

Prompt: „Bitte generiere mir ein Symbol im Retro Pixel Art Neon Stil mit zwei Würfeln, die ein "Tausch" Symbol haben -> Twister“ (Im Anhang die erste generierte Fähigkeitskarte)

Karten-Rückseite

https://chatgpt.com/backend-api/estuary/content?id=file_0000000044c07246bce52656dd4a77e4&ts=491386&p=fs&cid=1&sig=fedffa2a8a2eadbf2244cee70f55230a6dbba3fa5183f25c9d6495d3ee059f51&v=0

Prompt: „Generiere mir jetzt die Karten Rückseite“

Schicksalswürfel Gerade/Ungerade Zahlen

https://chatgpt.com/backend-api/estuary/content?id=file_0000000013cc71f4bfb0c8d6e88dec5b&ts=491386&p=fs&cid=1&sig=0393a9c60732de74b4b4bdffae87148613d90c094df789a0bccd00ac15d7f0af&v=0 (geändert und angepasst in Photoshop)

Prompt: „Generiere mir einen einzelnen Würfel, ebenfalls im Retro Pixel Neon style, der etwas besonderer aussieht. Das Image zeigt den Würfel von oben.“

6.2 RECHERCHE

Hier sind die relevantesten Quellen angegeben, die ich zum Programmieren und zum Arbeiten in der Engine verwendet habe und neue brauchbare Informationen beinhalteten. Daneben habe ich für kleine kurze Recherchen außerdem mit Forenbeiträgen von https://discussions.unity.com/, https://gamedev.stackexchange.com/, https://stackoverflow.com/ und https://www.reddit.com/ gearbeitet.

Original „Knucklebones“

(Selber gespielt und analysiert)

Controller Input und Input System für Multiplayer:

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/HowDoI.html#find-all-connected-gamepads

https://discussions.unity.com/t/how-to-disable-input-from-other-controllers-using-playerinput/843617/2

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/Devices.html

https://www.youtube.com/watch?v=u3KoWI92blE&list=PLyeuxHOiqLIQsU5m3fNwb-GjK9HiqSDdf&index=1

https://www.youtube.com/watch?v=miI82pJCxSY&list=PLyeuxHOiqLIQsU5m3fNwb-GjK9HiqSDdf&index=2

https://www.youtube.com/watch?v=N_ggBtEYQQs&list=PLyeuxHOiqLIQsU5m3fNwb-GjK9HiqSDdf&index=3

Menu Management:

https://www.youtube.com/watch?v=e30i55Hp-to&list=PLyeuxHOiqLIQsU5m3fNwb-GjK9HiqSDdf&index=4

Post Processing mit Global Volume:

https://www.youtube.com/watch?v=9tjYz6Ab0oc&list=PLyeuxHOiqLIQsU5m3fNwb-GjK9HiqSDdf&index=5

Particle System:

https://www.youtube.com/watch?v=FEA1wTMJAR0 (+ eigene Erfahrungen)

Visual Effects:

https://www.youtube.com/watch?v=BQGTdRhGmE4&list=PLyeuxHOiqLIQsU5m3fNwb-GjK9HiqSDdf&index=6

http://youtube.com/watch?v=I4PVGffuoB8

6.2.1 ChatGPT Recherche Hilfe

Neben normaler Google und YouTube Recherche habe ich auch an bewussten Stellen ChatGPT fragen müssen. Ich verwendete ChatGPT vor Allem, um (für mich) „unlösbare“ Probleme schnell zu beseitigen oder Ergänzungen zu Internet-Quellen zu erfragen.

Ich verwendete ChatGPT bei:

- → Rainbow Effekt für Exodia
- → Kartendrehung Logic
- → Aufwendig geschriebener Code simpler schreiben
- → Erklärungen für bestimmte Code-Zeilen in einfacher Sprache
- → Vorgehensweise für das PlayerJoinManager Script
- → Probleme in der Logik verstehen

Die genaue Dokumentation der Prompts und Antworten von ChatGPT ist gelistet in dem externen Dokument „lin_hoppe_GDS23_2026_ChatGPT_Hilfe“.