MIB14 – Softwareentwicklungsprojekt

WISE 2021/22

Prof. Dr. Carsten Lucke

Softwarearchitektur

„Metroidvania“

von

Nada Savic (5316524)

Noah Fahrholz (5336038)

Azzedin Berrou (5321564)

Robin Prause (5323834)

Inhaltsverzeichnis

Template

2022-02-03

Über arc42

arc42, das Template zur Dokumentation von Software- und Systemarchitekturen.

Erstellt von Dr. Gernot Starke, Dr. Peter Hruschka und Mitwirkenden.

Template Revision: 8.0 DE (asciidoc-based), Februar 2022

Einführung und Ziele

Aufgabenstellung

Inhalt

Ziel ist das Umsetzen eines Plattformers nach der Art von „MetroidVania“.

Motivation

Das Spiel “Ori and the Blind Forest“ ist aufgrund der Designelemente unverwechselbar und zeigt die Spieleklassiker nach der Art Metroidvania in einem modernen und mystischen Stil. Wir möchten uns dies als Vorbild nehmen und entwerfen einen Plattformer im ähnlichen Stil.

Form

Das Spiel kann durch downloaden der .exe / .dmg Datei heruntergeladen, installiert und gespielt werden.

Qualitätsziele

Inhalt

Leistung	*Sehr gut*	*Gut*	*Normal*	*Irrelevant*
Funktionalität		x		
Zuverlässigkeit		X		
Benutzbarkeit	x			
Effizienz		x		
Änderbarkeit			x	
Portierbarkeit			x	

Motivation

Grundsätzlich steht bei einem Spiel die Benutzbarkeit an höchster Stelle. Es soll Spaß bereiten, das Spiel zu spielen und nicht zu einer Herausforderung werden, es erst zum Laufen zu bekommen. Wichtig sind weiterhin Funktionalität, Zuverlässigkeit und Effizienz. Die Level mit ihren Features und Gegnern sollen ohne Einschränkungen funktionieren und zuverlässig zu starten und zu spielen sein.

Stakeholder

Inhalt

Stakeholder sind in unserem Szenario das Team, welches im Folgenden vorgestellt wird, sowie Prof. Dr. Carsten Lucke, der als Kunde das Spiel letztendlich abnimmt.

Form

Rolle	**Kontakt**
Kunde	Prof. Dr. Carsten Lucke
Projektmanager	*Robin Prause*
Projektleiter	*Noah Fahrholz*
Chefdesigner	*Azzedin Berrou*
Qualitätsbeauftragte	*Nada Savic*

Kontextabgrenzung

Inhalt

Das Spiel wird als Installationsdatei und als WebGl Anwendung zur Verfügung gestellt.

Motivation

Durch die hochwertigen Grafiken soll es keine Einschränkung beim Spielerlebnis geben. Wir möchten aber auch den Komfort behalten ein Spiel einfach über eine Website starten zu

können und das Platformunabhängig. Daher entschieden wir uns parallel zur performanteren Exe noch einen zusätzlichen Web-build zu veröffentlichen.

Lösungsstrategie

Inhalt

Idee ist das Entwerfen eines Prototyps, um die Kreativität zu erhöhen und zu ermitteln, was gut umsetzbar ist. Gewählt wurde also ein agiler Ansatz, der erlaubt, dass die Dokumentation inkrementell erweitert wird. Unity gilt mit als Vorreiter der Entwicklungsumgebungen für Spiele. Weiterhin bestanden im Entwicklungsteam gute Kenntnisse im Umgang mit Unity. Hier können einfach sogenannte Builds erstellt werden, die das Endprodukt repräsentieren.

Motivation

Durch den Prototyp werden bereits die Grundsätze des Spiels und der implementierten Grafiken gelegt. Somit kann sich auf weitere Details und Besonderheiten fokussiert werden, die das Spieleerlebnis erheblich verbessern.

Form

Am wichtigsten ist uns die Benutzbarkeit des Spiels. Durch das Nutzen der Unity internen Library werden unter anderem wichtige Sicherheitsmerkmale bereitgestellt, die verhindern, dass eigene Bugs eingebaut werden. Da es sich um einen 2D Plattformer handelt, kann Unitys Input System verwendet werden, um Skript und Charakter zu verbinden und so einfach einfaches Bewegungssystem zu erstellen.

Bausteinsicht

Inhalt

Im Folgenden werden mit Übersichtsdiagrammen die verwendeten Strukturen verdeutlicht. Da Unity bereits den Großteil der verwendeten Technologien bietet, kann viel mit Blackboxen gearbeitet werden, die viele Funktionalitäten abbilden. Im Fokus stehen dabei die Komponenten und Skripte der sogenannten GameObjects.

Motivation

Um auch nachträglich das Spiel zu skalieren, neue Features zu entwickeln und den Überblick zu behalten wird hier auf abstrakter Ebene die Bausteine und deren Interaktionen in Unity beschrieben.

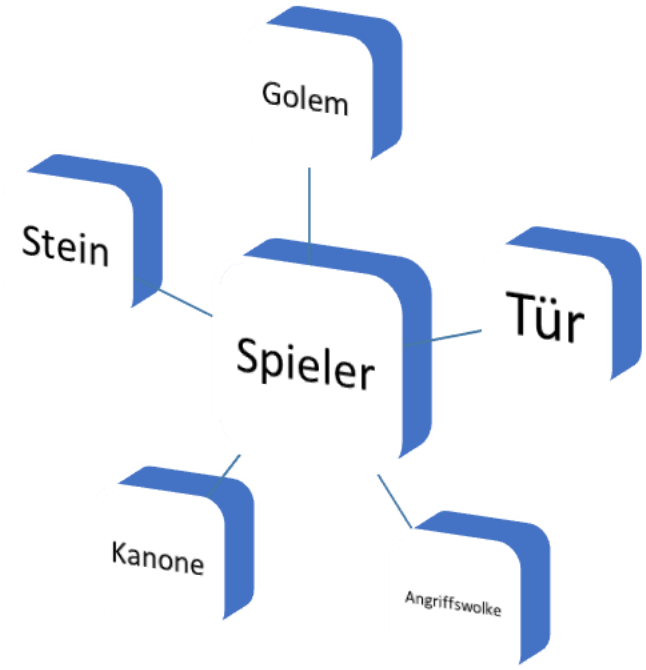
Form

Ebene 1 ist die Whitebox-Beschreibung des Gesamtsystems, zusammen mit Blackbox-Beschreibungen der darin enthaltenen Bausteine.

Ebene 2 zoomt in einige Bausteine der Ebene 1 hinein. Sie enthält somit die Whitebox-Beschreibungen ausgewählter Bausteine der Ebene 1, jeweils zusammen mit Blackbox-Beschreibungen darin enthaltener Bausteine.

Ebene 3 zoomt in einige Bausteine der Ebene 2 hinein, usw.

Whitebox Gesamtsystem

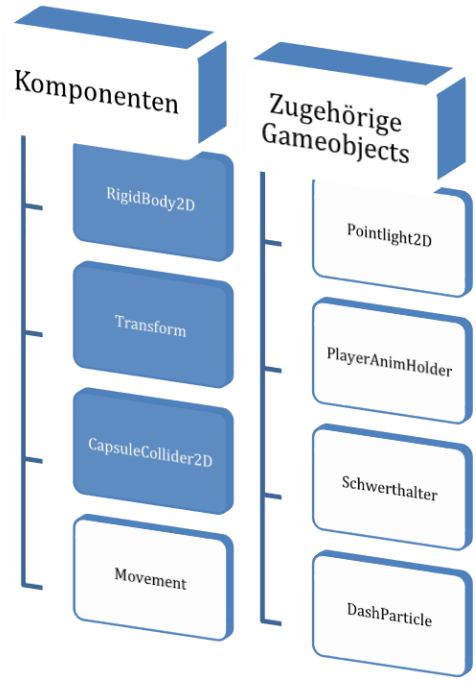


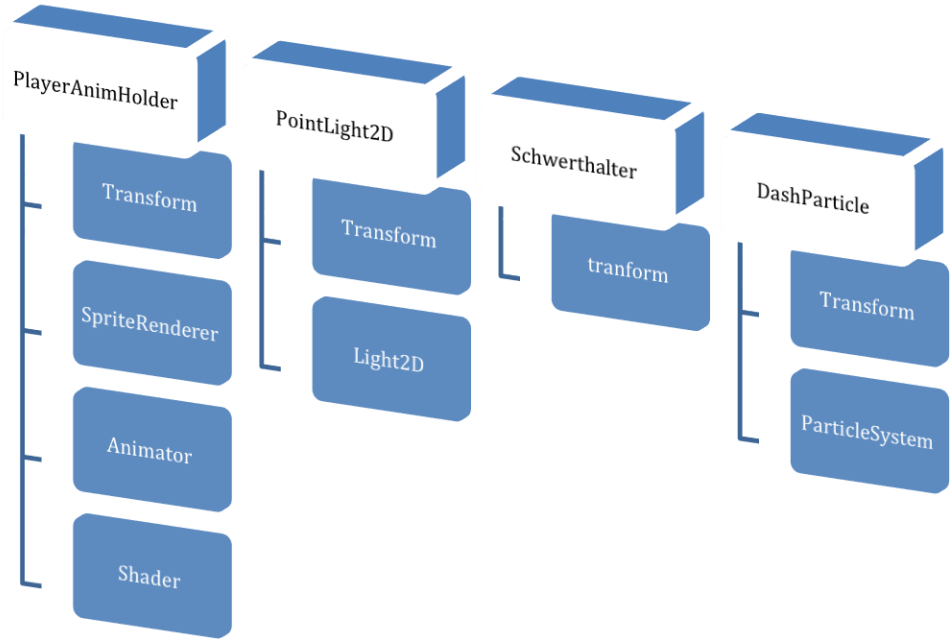
Enthaltene Bausteine

Name	Verantwortung
Spieler	Wird bewegt – kollidiert mit Gegenständen
Angriffswolke	Kann bei Treffer Schaden zufügen
Golem	Kann durch Angriff Schaden zufügen
Stein	Kann bei Auftreffen Schaden zufügen
Kanone	Kann schießen und bei Treffen des Spielers Schaden zufügen
Tür	Kann durch Spieler geöffnet werden

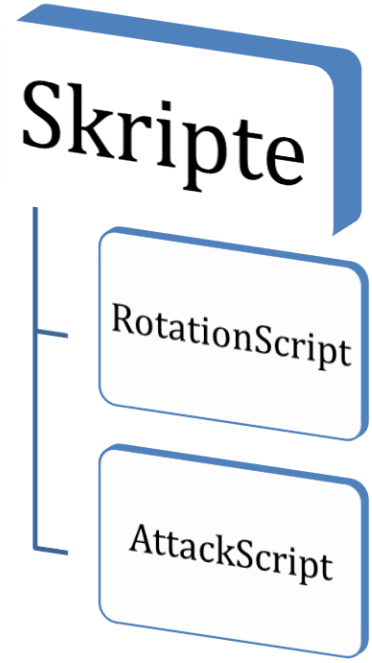
Ebene 2 - Gameobjects

Whitebox<*Spieler*>

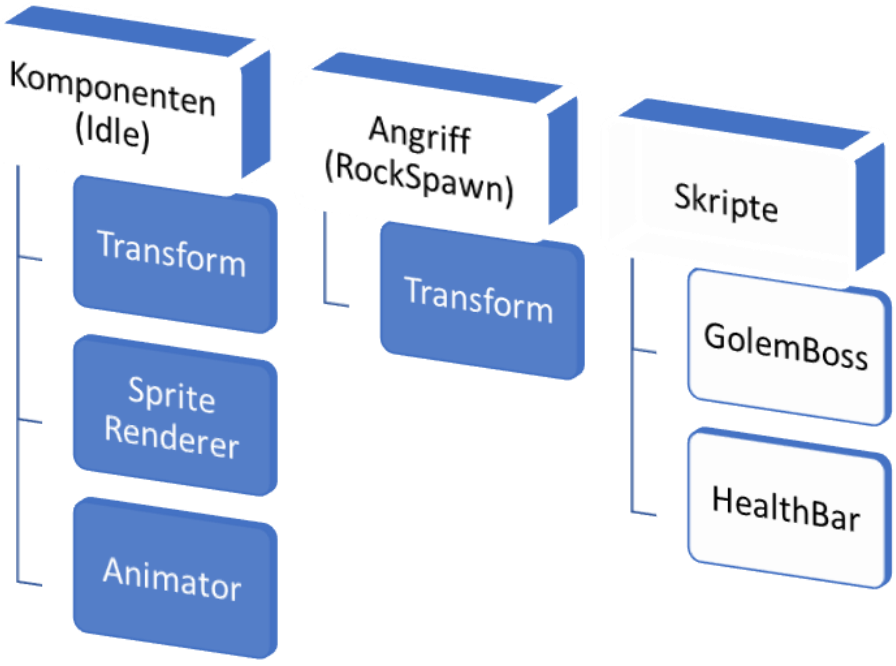




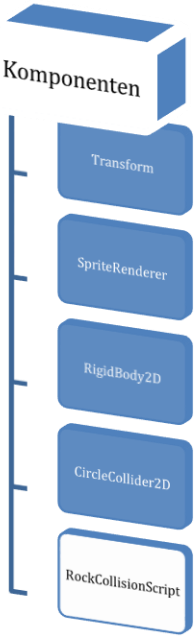
Whitebox *<Angriffswolke>*



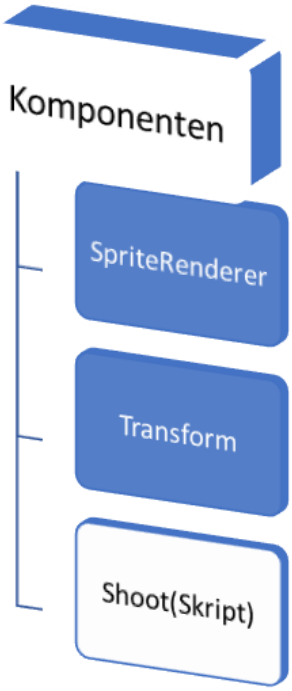
Whitebox *<Golem>*



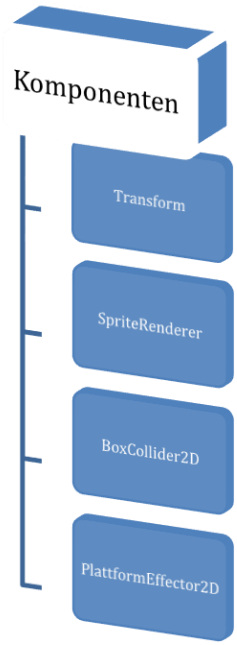
Whitebox *<Stein>*



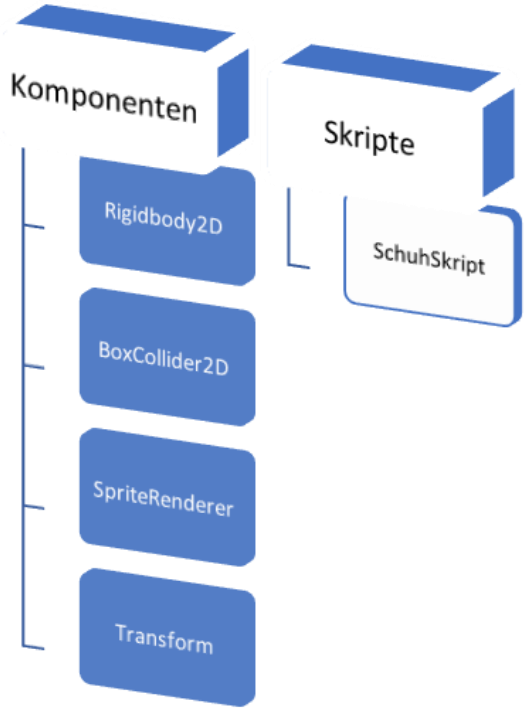
Whitebox *<Kanone>*



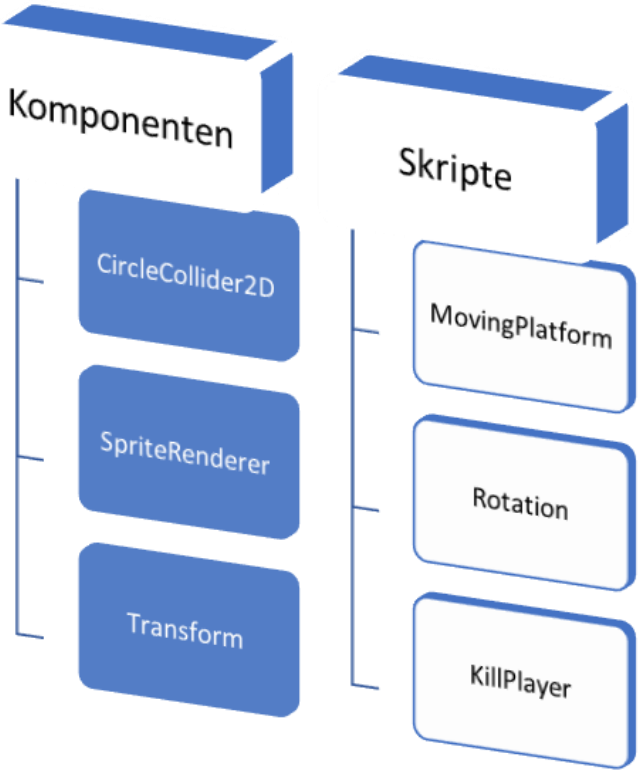
Whitebox *<Tür>*



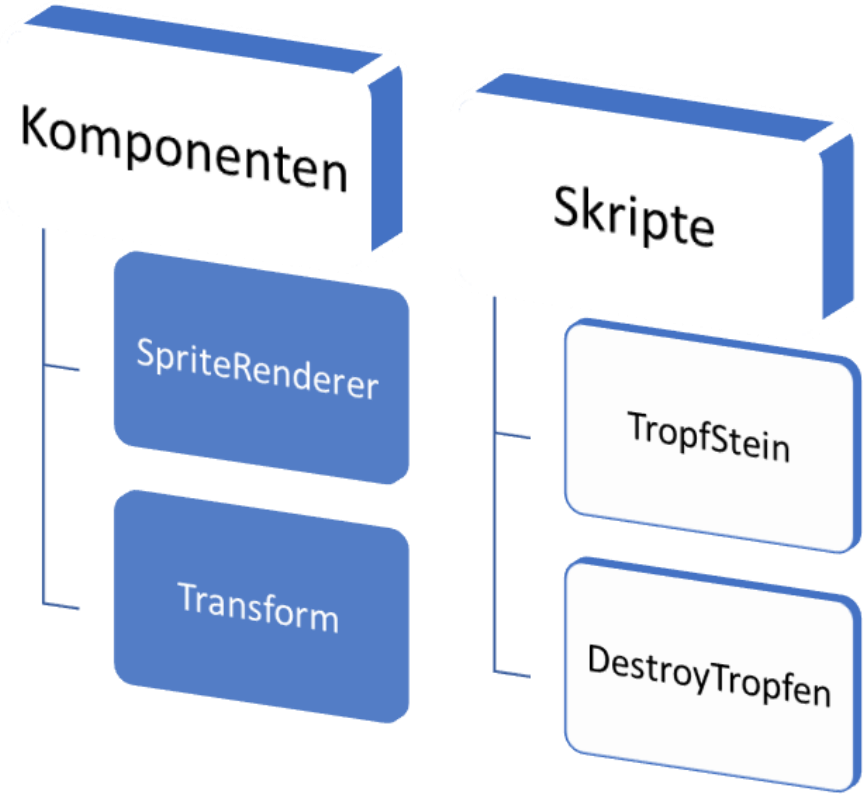
Whitebox *<Schuh>*



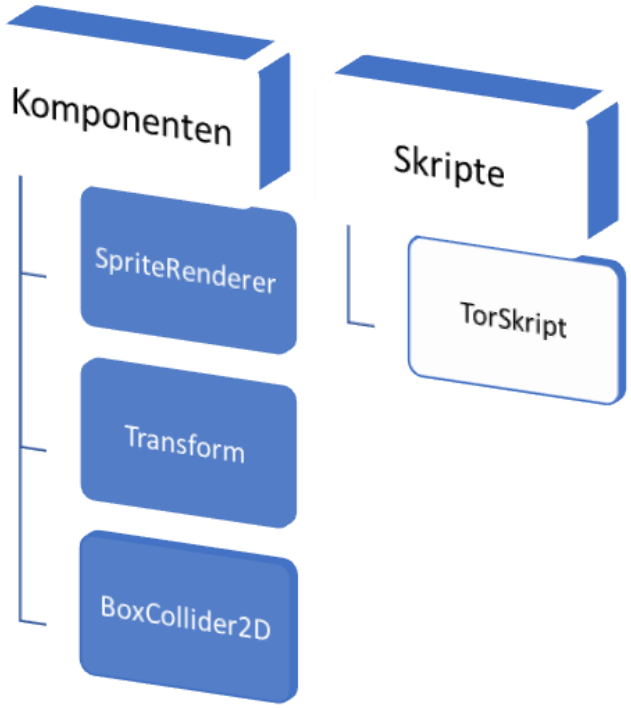
Whitebox *<MovingBarrels>*



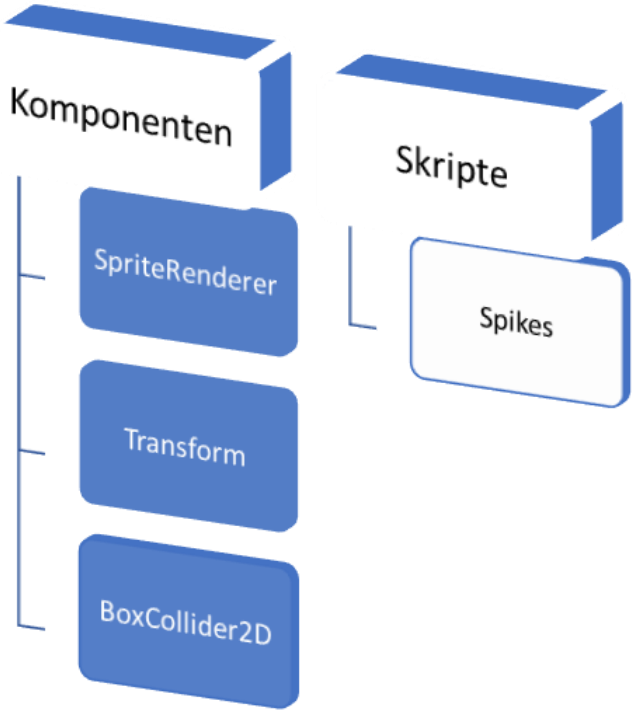
Whitebox *<TropfStein>*



Whitebox *<Tor>*



Whitebox *<Spikes>*



Ebene 3 – Skripte/ Klassen

Whitebox <GameManager>

Kümmert sich um das Szenenmanagement, die Pausenmenüabfrage und das Speichern der Schlüsselartefakte die man einsammeln muss.

Whitebox <MovementScript>

Bewegt den Spieler und kommuniziert die Bewegungen mit dem Animator vom Spieler.

Whitebox <Levelloader>

Kümmert sich um ein flüssiges Fade In zwischen zwei Szenen

Whitebox <Kanonen>

Fügt der Kugel der Kanone das Verhalten und die Selbstzerstörung hinzu

Whitebox <Shoot>

Sorgt für das regelmäßige Feuern der Kanone

Whitebox <Platform>

Checkt die Collision der Platform mit dem Player und stellt sich selbst als Elternelement des Players

Whitebox <MovingPlatform>

Bewegt die Platform zwischen zwei Punkten

Whitebox <Spikes>

Checkt ob Spieler kollidiert und tötet diesen dann.

Whitebox <Health>

Kümmert sich um das Leben des Spielers im Kampf gegen den Golem

Whitebox <Player>

Kümmert sich um den Tod des Spielers.

Whitebox <AreaSkript>

Definiert den Bereich in welchem der Schuh aktiviert werden soll

Whitebox <Schuh>

Beschreibt das Verhalten des Schuhs : Er soll dem Spieler auf der x Achse folgen und Stampfen sobald dieser stehen bleibt.

Whitebox <Tor>

Lädt die nächste Szene des Spiels, aber nur wenn alle Schlüssel vorhanden sind.

Whitebox <GolemBoss>

Regelt das Leben des Golems und leitet den Steinhagel in der zweiten Kampfphase ein.

Whitebox <GolemDying>

Beendet die Animationen und die Zweite Kampfphase

Whitebox <GolemRun>

Sorgt dafür, dass der Golem auf den Spieler zu rennt.

Whitebox <GolemJump>

Sorgt dafür, dass der Golem auf den Spieler zu springt und wieder Richtung Boden fällt.

Whitebox <RockCollision>

Sorgt dafür, dass der geworfene Stein des Golems einen Effekt Instanziiert, wenn er auftritt und zerstört den Stein, wenn dieser kollidiert.

Whitebox <RockThrow>

Sorgt dafür, dass der Stein von dem Golem nach links oder rechts auf der x - Achse geworfen wird.

Whitebox <MakePunchDamage>

Sorgt dafür, dass der Spieler während der Schlag Animation schaden bekommt.

Laufzeitsicht

Inhalt

Golem Boss

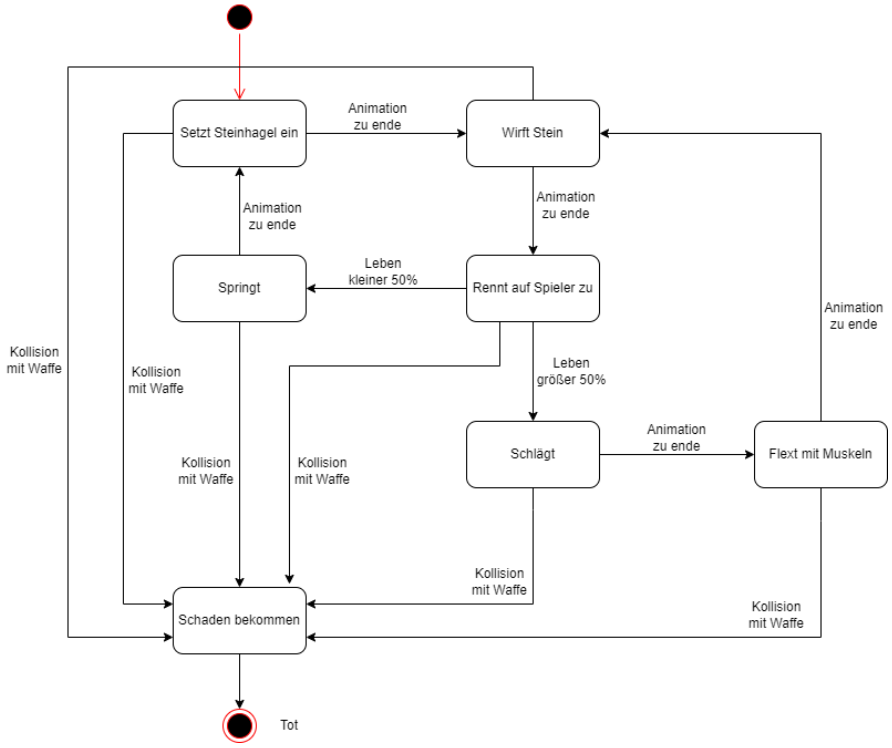
Der Golem hat einen Animator, welcher die Animationen des Golems beinhaltet. Diese heißen wie folgt: “GolemRoar”, “GolemSwiping”, “GolemFlexingMuscles”, “GolemPunch”, “GolemJumpAttack”, “GolemRun” und “GolemDying”. Auf diesen Animationen befinden sich die Skripte um den Golem zu Bewegen. Auf dem Golem selbst befindet sich das Skript “GolemBoss”. Dieses fungiert als Lebens Manager des Golems. Folgende Animationen haben Skripte zugeteilt:

GolemSwiping -> RockThrow Skript

GolemRun -> GolemRun Skript

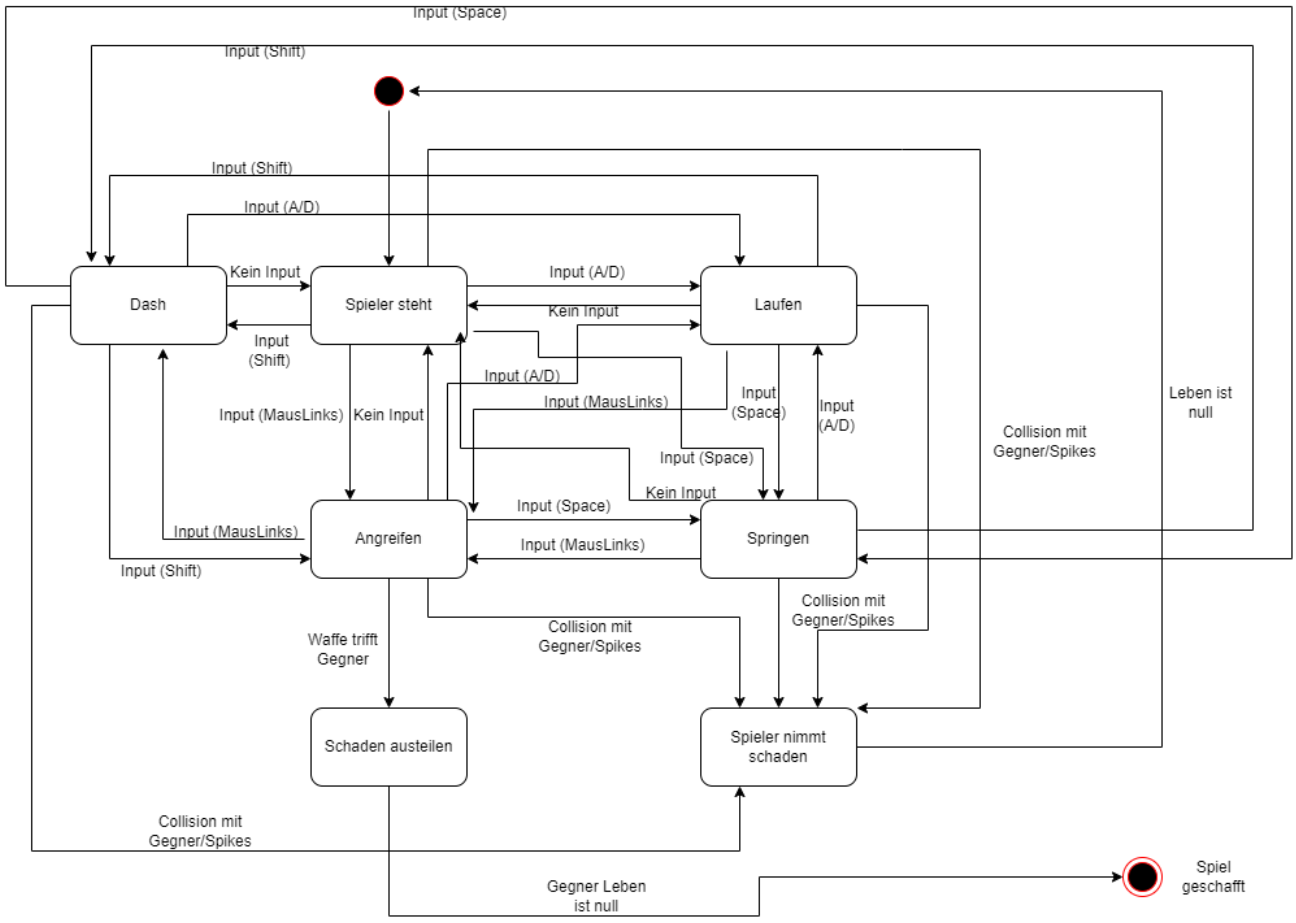
GolemJumpAttack -> GolemJump Skript

GolemDying -> GolemDying Skript

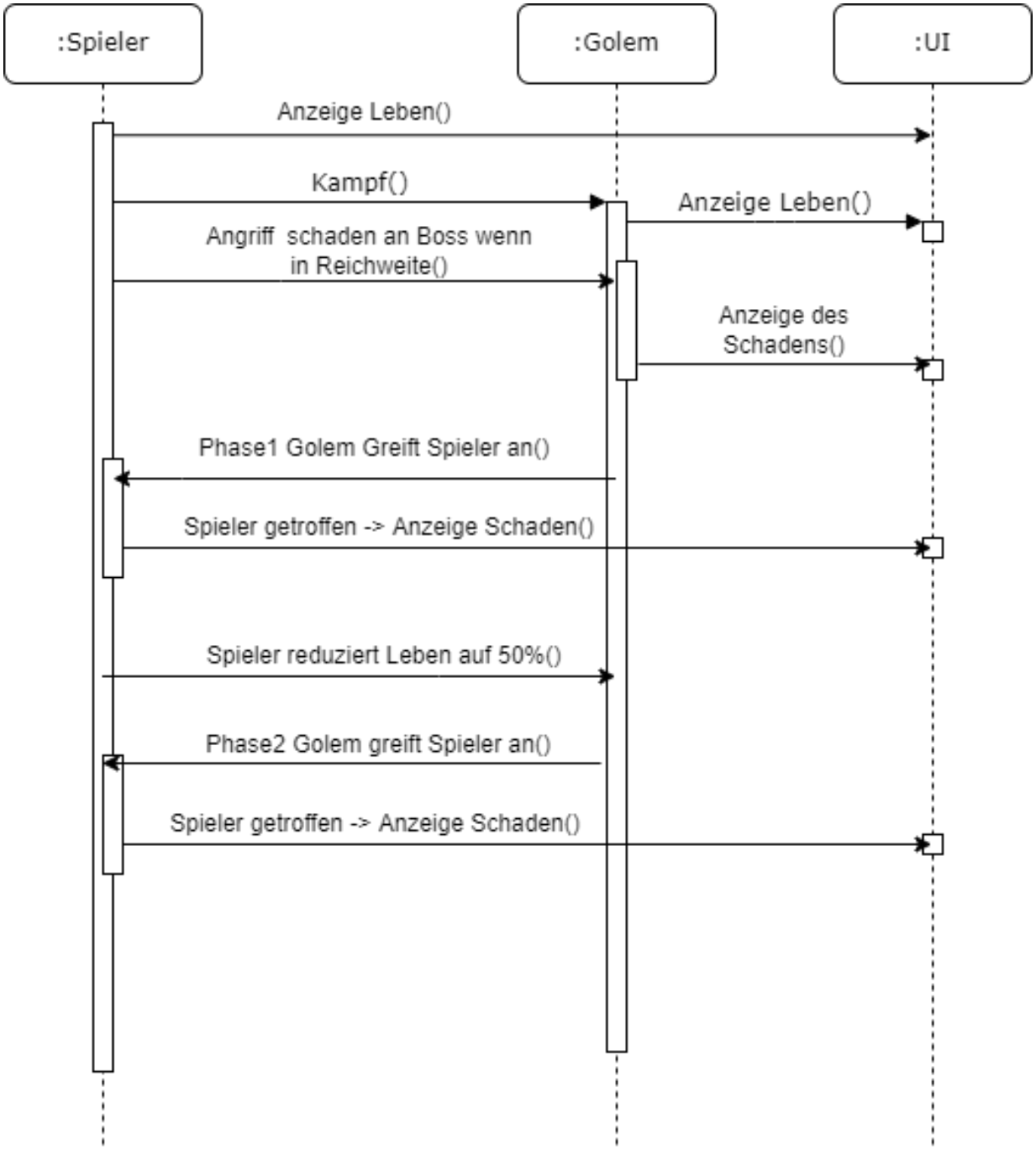


Der Spieler

Der Spieler hat drei Skripte. Zum einen haben wir das Movement Skript, welches für die Bewegung zuständig ist. Dann gibt es noch das Health Skript, welches für das Lebens Management des Spielers zuständig ist und zum Schluss noch das Player Skript, welches dafür sorgt, dass der Spieler sterben kann und wieder diesen wieder an den Anfang schickt.



Boss Kampf


:Spieler
:Golem
:UI
Anzeige Leben()
Kampf()
Anzeige Leben()
Angriff schaden an Boss wenn in Reichweite()
Anzeige des Schadens()
Phase1 Golem Greift Spieler an()
Spieler getroffen -> Anzeige Schaden()
Spieler reduziert Leben auf 50%()
Phase2 Golem greift Spieler an()
Spieler getroffen -> Anzeige Schaden()

<Bezeichnung Laufzeitszenario 1> - Menü betreten

Zunächst befindet man sich nach Start des Spiels im Hauptmenü wo man die Möglichkeit hat das Spiel durch zwei Knöpfe zu starten oder zu beenden. Wenn Start gedrückt wird, dann wird die erste Szene geladen. Wenn man auf verlassen drückt wird das Spiel geschlossen.

<Bezeichnung Laufzeitszenario 2> - Das Spiel

Wenn man nun starten drückt steigt man ein in eine Mystische Welt, welche in einem hellen Blau erstrahlt. Man bewegt sich mit einer kleinen Kreatur vorwärts durch ein Labyrinth, das mit schönen Pflanzen bewachsen ist. Ebenso ist diese Wundersame Welt mit Fallen und Gegnern versehen. Zunächst begegnet man einem Stiefel, vor dem man sich in Acht nehmen muss. Dieser verfolgt den Spieler und droht auf diesen drauf zu treten. Anschließend kommt eine Passage, in der man sich durch tödliche Stacheln manövrieren muss. Wenn das nun geschafft ist, erwartet den Spieler ein schmaler Gang, wo er beschossen wird und den Kugeln einer Kanone ausweichen muss. Für diese Hindernisse stehen dem Spieler verschiedene Bewegungsmöglichkeiten zur Verfügung. Zum einen kann man sich nach links und rechts bewegen in dem man mit den Tasten „A“ und „D“ fortbewegt. Ebenso kann man mit der Lehrtaste springen und sich mit rechtsklick der Maus an Wänden festhalten. Für noch mehr Ausweichmöglichkeiten sorgt ein „Dash“(Sprint) mit welchem man sich schnell in eine gezielte Richtung bewegen kann. Das Ziel in diesem Labyrinth aus fallen ist es Schlüsselfragmente zu Sammeln. Diese werden benötigt, um eine Tür zu öffnen, welche den Spieler zur Endstufe des Spiels befördert. Hierbei handelt es sich um einen Boss Kampf. Bei diesem Boss handelt es sich um einen Golem, welcher mit einer Waffe bekämpft wird. Diese Waffe schwebt dem Spieler hinterher und kann mit der linken Maustaste in Richtung Mauszeiger bewegt werden. Bei diesem Kampf ist ebenso Geschick gefragt denn die Attacken des Golems welche aus Schlägen, Sprüngen, Steinwürfen und Steinhageln bestehen müssen gekonnt ausgewichen werden. Die Attacken werden in zwei Phasen des Bosses aufgeteilt. Die erste Phase startet von Anfang an und die zweite bei einem Lebensverlust des Golems von 50%. Hierbei ist es möglich die Steine mit der Waffe des Spielers zu zerschlagen. Für den Bosskampf hat man insgesamt sieben Leben, diese können schnell verloren gehen in dem man von Steinen, Schlägen oder dem Golem selbst getroffen wird. Wenn die Leben auf null fallen, muss man den Boss erneut bekämpfen, bis dieser Besiegt ist. Die Leben werden oben links in der Ecke des Bildschirms angezeigt. Ebenso hat der Golem eine Lebensleiste sowie einen Namen. Diese Elemente sind mittig am Unteren Rande des Bildschirms positioniert. Wenn die Lebensleiste des Golems gelehrt wurde und auf null fällt, stirbt der Golem und das Spiel gilt als gewonnen. Während dem Spiel kann die Escape Taste gedrückt werden und das Spiel pausiert werden. Hier hat man dann zwei Knöpfe. Einmal kann man zurück in das Spiel oder verlassen drücken und das Spiel wird beendet.

<Bezeichnung Laufzeitszenario 3> - Spiel verlassen

Während dem Spiel kann die Escape Taste gedrückt werden und das Spiel kann verlassen werden.

Querschnittliche Konzepte

Inhalt

Für das Realisieren unseres Spiels verwenden wir Unity. Dabei handelt es sich um eine Entwicklungsplattform, die primär zum Entwickeln von Spielen, aber auch für das Erstellen interaktiver 3D Erlebnisse verwendet wird. Dadurch sind bereits grundlegende Libraries und Packages inkludiert, die dabei helfen, kaum auf weitere Schnittstellen oder externe Frameworks zugreifen zu müssen.

Motivation

Durch das Verwenden von Unity Internen Packages wird ein höherer Zuverlässigkeitsstandard bereitgestellt und der Fokus auf spezifische und für das Spiel besondere Aspekte gelegt.

Form

=== *<Unity Input System >*

- ⇨ *Dient dem Binden der UI und des Spielers an Nutzereingaben*
- Klasse / Interface
- Methoden automatisiert Abfragen
- Keys werden über GUI gesetzt -> kein manuelles Key Binding notwendig

=== *<Sprite Renderer>*

- ⇨ *Rendert sämtliche Sprites*
- Sprite entspricht einer Grafik

- Fügt dem GameObject eine Grafik hinzu,

=== *<Animator>*

⇨ *Animiert die Bewegung des Spielers und des Golems*

- *StateMachine*
- Je nachdem, in welchem State sich der Spieler befindet, wird die Animation gewechselt und durch den Sprite Renderer angezeigt (State-idle oder State-walking)

=== *<Particle System>*

⇨ *Dient dazu dem Spiel Effekte hinzuzufügen*

- *Nimmt Particles und weist diesen Eigenschaften zu, um verschiedenste Effekte zu erzeugen*
    - *Eigenschaften: Lebensspanne, Größe, Farbe, Verhalten, etc.*

=== *<Render Pipeline (URP, Universal Render Pipeline)>*

- URP hat einen 2D Renderer und kann dadurch 2D Beleuchtung anzeigen
- rendered am Ende das Spiel, sodass alles angezeigt werden kann

=== *<ShaderEditor>*

⇨ *Für das Erzeugen von Oberflächen*

⇨ *verändert das Aussehen, nicht aber die Form von Objekten*

⇨ *Erzeugt unter anderem Emissionen oder Transluzenz*

=== *<Audiomanager>*

- *Audiodateien werden hier eingebunden und abgerufen*