Game Design Document

Autoren:
Franz Mörike,
René Kraus,
Jan Pagel

Letztes Update:
02.07.2021

Inhaltverzeichnis

Game Analyse & Konzept

Grundidee:

Der Spieler hat sich in einem unterirdischen Labyrinth verlaufen (Abb. 1). Dort warten gefährliche Untote auf ihn und versuchen ihn zu töten und seine Seele aus dem Leib zu reißen. Der Spieler versucht vor diesen zu fliehen und den Ausgang aus dem verwirrenden Kellergebilde zu finden. Die Flucht gestaltet sich aber nicht ganz so einfach, da das Labyrinth diverse böse Überraschungen für den Spieler parat hat. Das Labyrinth ist an die griechische Mythologie um das Labyrinth von Knossos angelehnt.

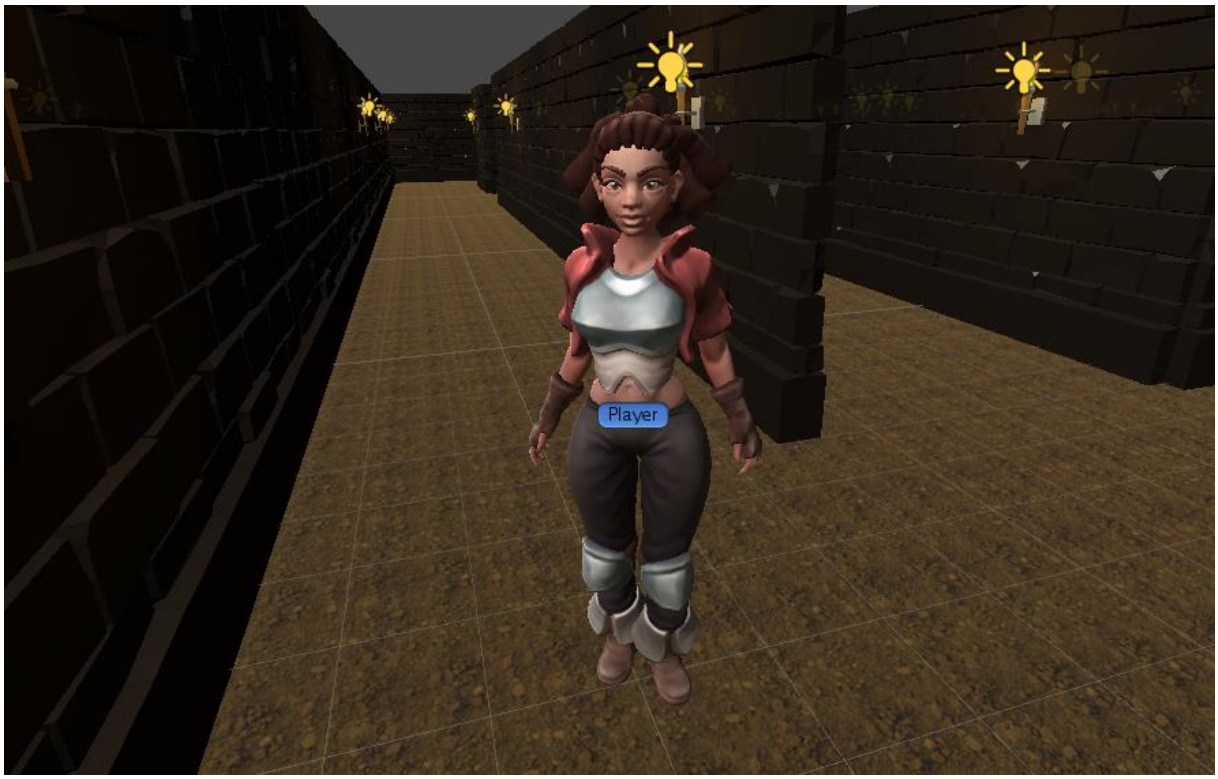

Abb. 1 Spieler inmitten des Labyrinths

Skriptreferenzen:

- GameController.cs

Genre

- Creepy Survival Adventure

Plattformen

- PC

Zielgruppe

Da die Steuerung simpel gehalten ist, ist das Spiel auch für unerfahrene Spieler geeignet.
Es soll vor allem Spieler ansprechen, die sich gerne erschrecken und sich für fast ausweglose Labyrinthe begeistern.

Gameplay

Gameplay Überblick

Das Gameplay besteht grundsätzlich darin, aus dem Labyrinth zu entkommen und zu überleben, damit die Untoten nicht die Seele des Spielers fressen. Außerdem muss der Spieler taktische Lösungen finden, um sowohl die Untoten zu umgehen als auch die Fallmechaniken auszutricksen.

Die Fallen die dem Spieler das Überleben erschweren können:

- Falltüren
- Einstürzende Decken
- Wände, die den Spieler zerquetschen

Um das Gameplay noch interessanter zu gestalten, sind *progressive gameplay mechanics* geplant, die dafür sorgen, dass der Spieler gewisse Schritte erfüllen muss, um im Level weiterzukommen und schlussendlich den Ausgang zu finden.

Gameplay Mechaniken

Game Modus	
Überlebensmodus	Der Spieler versucht aus dem Labyrinth zu entkommen.

Skriptreferenzen:

- HiddenDoor_Behavior.cs
- Portal_Behavior.cs
- TrapDoor_Behavior.cs
- TrapRocks_Behavior.cs

Performance enhancing

Um die Performance des Spiels zu verbessern, werden einige Anpassungen vorgenommen (Abb 2/3).

- Für die Lichtquellen (z. B. Fackeln) an den Wänden werden „lightmaps“ generiert.
- Die Performance ziehenden Mesh Collider werden durch Box Collider ersetzt.
- Zum Rendern des Levels wird Occlusion culling eingesetzt. Dadurch werden nur die Teile des Levels gerendert, die auch vom Spieler eingesehen werden.

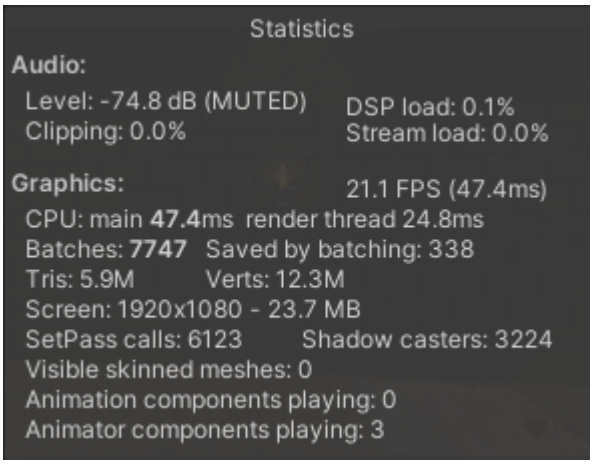

Abb. 2 Werte vor der Optimierung

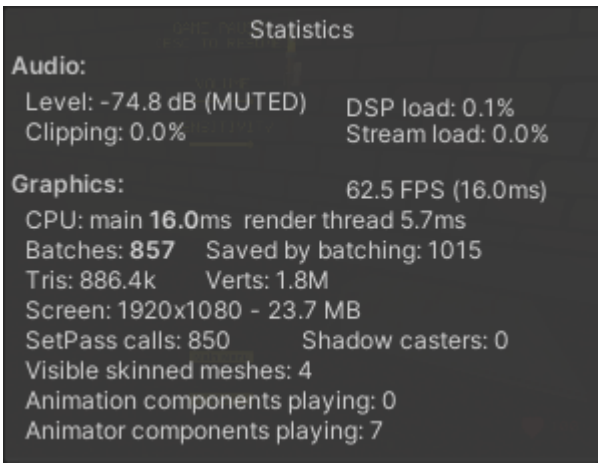

Abb. 3 Werte vor der Optimierung

KI

Das Verhalten der Untoten wird mit dem EnemyAI.cs Skript gesteuert. Dieses bestimmt die Bewegungsabläufe und -richtungen anhand von Pathfinding-mechaniken (Pathfinding.cs). Der für die KI begebbare Bereich wird durch die NavMesh bestimmt. (Abb.4/5/6 NavMesh).

Abb. 4 NavMesh vom Default Level

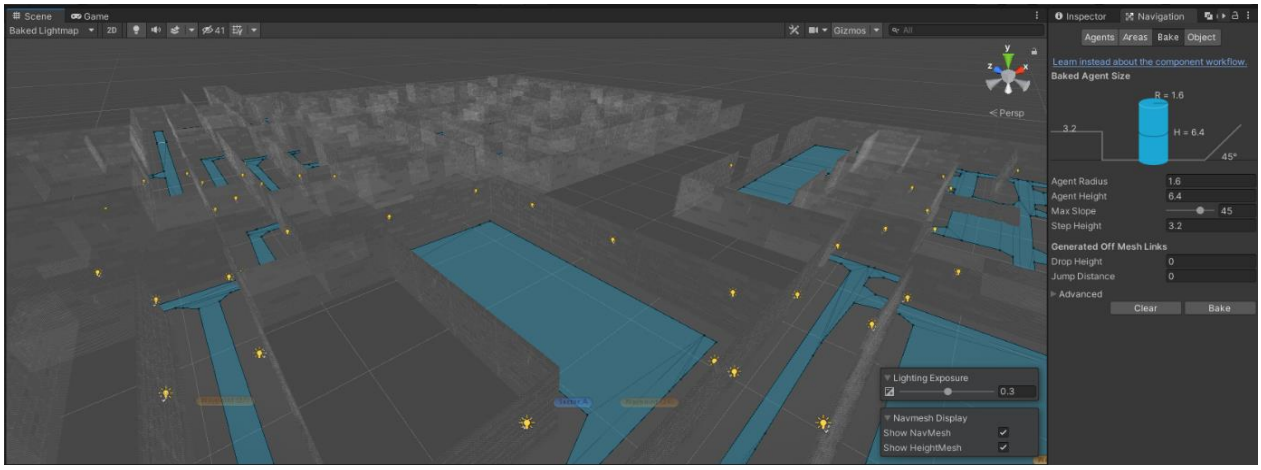

Abb. 5 NavMesh vom Default Level

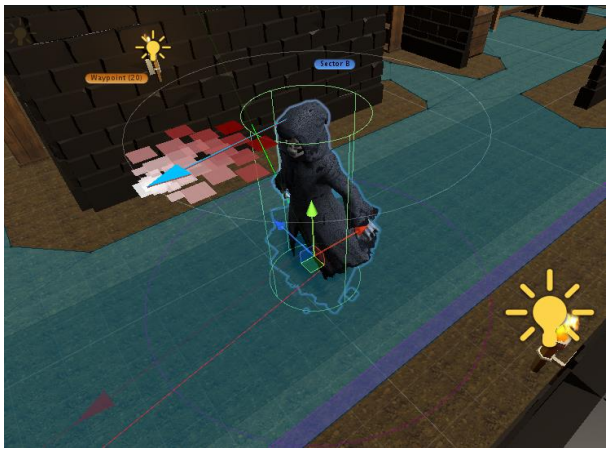

Abb. 6 KI beim Ablaufen der Wegpunkte

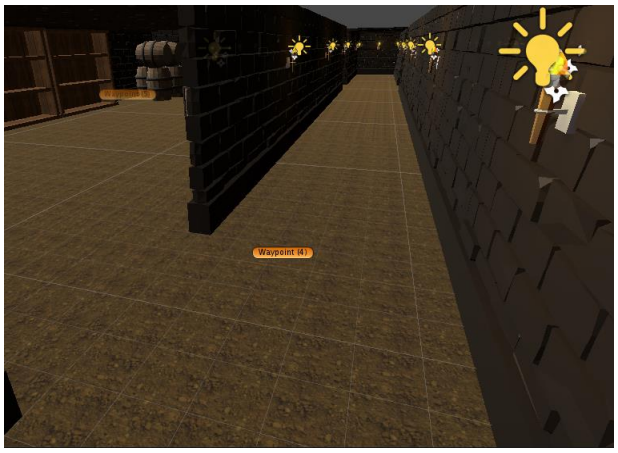

Abb. 7 Wegpunkt

Sobald der Spieler sich in der Nähe vom(/n) Gegner(n) befindet, verfolgt die KI den Spieler durch die eingebaute smart-player-detection (SearchPlayer.cs) und greift diesen an, wenn er in Reichweite ist.
Um die Wahrscheinlichkeit zu erhöhen, dass die KI den Spieler findet, wurde das Level in Sektoren unterteilt. Befindet sich der Spieler in einem Sektor, wird dessen Aufenthaltsdauer dort ermittelt und gespeichert. Jeder Sektor besitzt festgelegte Wegpunkte, die von der KI abgelaufen werden, der Sektor mit der höchsten Aufenthaltsdauer des Spielers übergibt diese an die KI.

Folglich ist es das Ziel des Spielers, möglichst den Untoten aus dem Weg zu gehen, und sollte er entdeckt werden, schnellstmöglich zu entkommen.

Skriptreferenzen:

- Pathfinding.cs
- SearchPlayer.cs
- SectorManager.cs
- Sector.cs
- Waypoint.cs

Kontrollschema

Das Spiel steuert sich mit Maus und Tastatur.

Button Input	Umsetzung im Spiel
W, A, S, D	Spieler bewegt sich in die entsprechende Richtung
Maus-Bewegung	Rotiert den Spieler („Umschauen“)
L shift – Linke-Umschalt-Taste	Sprinten

Skriptreferenzen:

- PlayerController.cs

Spiel Ästhetik / Design & Benutzeroberfläche

Ästhetik:

Das Spieldesign ist an eine düstere Kellerumgebung angelehnt, die spärlich durch Fackeln ausgeleuchtet ist (Abb. 8). Die Fantasy-Atmosphäre entsteht durch das an den Mythos von Knossos erinnernde Labyrinth. Dieses ist sehr verwinkelt und sorgt dadurch dafür, dass der Spieler sich leicht verläuft und in Räumen wiederfindet, die er vorher bereits besucht hatte. Außerdem ist das Labyrinth sowohl von gruseligen, skelettartigen Monstern besetzt, die versuchen den Spieler zu töten, als auch mit verschiedenen Fallen ausgestattet, die den Spieler zusätzlich behindern. Die düstere Atmosphäre unterstreicht animierte Elemente, wie Ratten, tropfende Decken und höhlenartige Geräuschkulissen. Die Assets sind dementsprechend von der Ästhetik her im low-Fantasy-genre, denn es werden keine „unrealistischen“ Assets verwendet abgesehen von dem Untoten. Die verlassenen größeren Kellerräume, in denen der Spieler sich anfangs wiederfindet, lassen Spielraum für Interpretation auf den Zweck, den diese Räume vor langer Zeit erfüllt haben (Abb. 9).

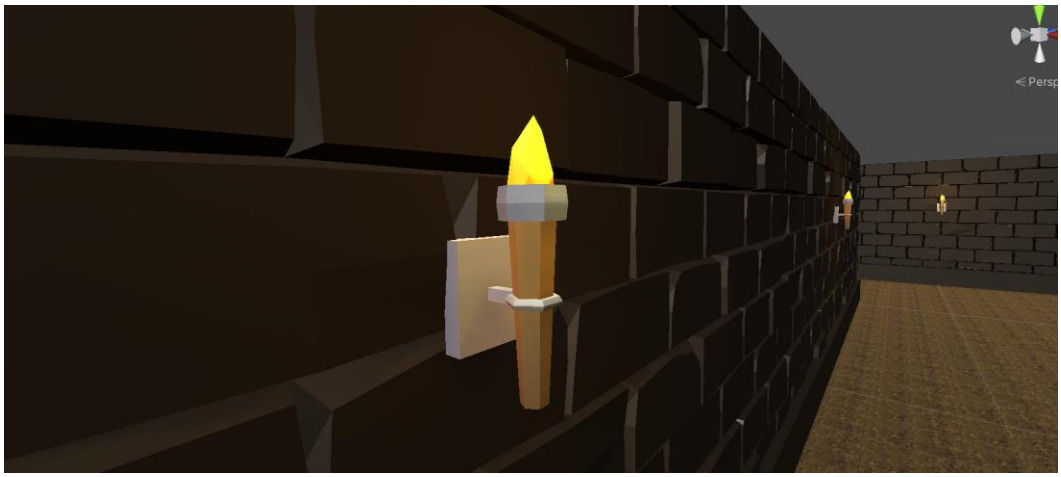

Abb. 8 Fackeln

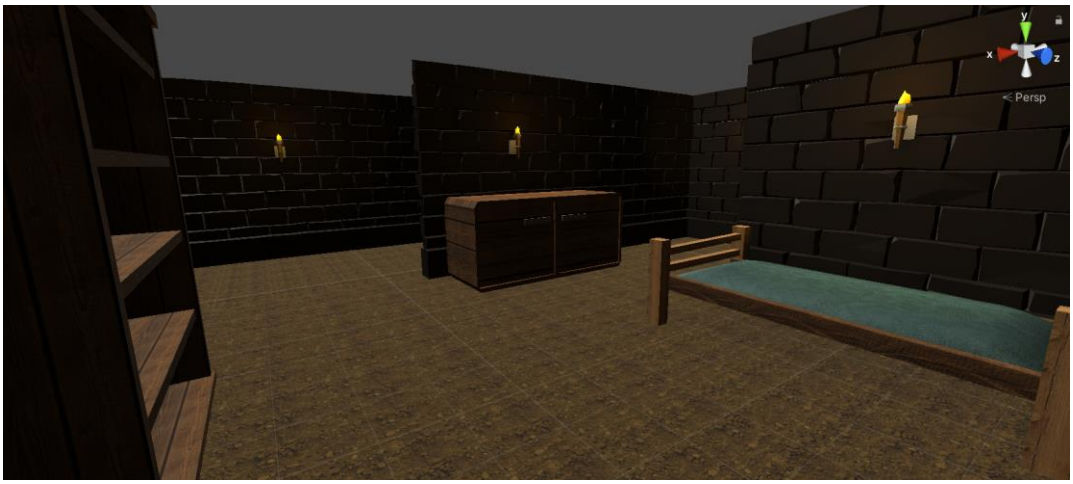

Abb. 9 Einrichtungsgegenstände der Kellerräume

GUI:

Die Nutzeroberfläche beschränkt sich größtenteils auf simple Knöpfe. Diese starten per Knopfdruck die gewünschte Aktion. Die graphische Nutzeroberfläche zeigt dem Nutzer die Lebensanzeige, die sich verringert, wenn Schaden genommen wird. Außerdem verdeutlicht ein Herz anhand der Farbe den Gesundheitszustand des Spielers.

Skriptreferenzen:

- ButtonBehaviour.cs
- GUIDeath.cs
- GUIHealth.cs
- GUIMainMenu.cs
- GUIOptionMenu.cs
- PlayerHealth.cs
- EnemyHealth.cs

Skripte

Die Autoren der Skripte *René Kraus*, *Jan Pagel* und *Franz Mörike* sind in jedem Script festgehalten und der Programmierfortschritt mit Timestamps geloggt. Es werden bislang keine externen Skripte verwendet.

Arbeitsaufteilung

Auch wenn sich die Arbeitsbereiche teilweise überschneiden, wollen wir Übersicht über unsere Fortschritte behalten.

[LEGENDE]
Fertig
In Arbeit

René	Jan	Franz
Basic movement	Sprites	GDD management
Mechanics (z.B. path finding)	Textures	GitHub project management
Enemy KI (player proximity detection, detection mechanic)	Level Design (Fallen, exit points)	Health mechanic (heart animation, damage / health system)
Enemy model		Option menu
Animationen		Death scene
Main menu		Button behavior
		GUI's

Quellen

- Unity asset store

TODO Liste

Must-have:

- Story ausarbeiten
- Musik / Hintergrundgeräusche
- Inventarsystem
- Progressive gameplay steps

Nice-to-have:

- Bullets / Schießen / stunning the enemy?
- Kriech- / Schleichmechanik
- Verschiedene Gegnerarten
- Springen
- Verstecken hinter Objekten

Probleme:

- KI bleibt manchmal einfach stehen
- Player kann noch durch Wände glitchen (evtl. Auslöser: Sprinten)