**Allgemein**

**GameController.cs**

Beschreibung:

In diesem Skript sollen allgemeine Abläufe behandelt werden, die für das Spiel und das Level notwendig sind.

Änderungslauf:

Version 1.0 (Erstellung)

* Deaktiviert den Cursor

Version 1.01

* Deaktiviert den Cursor und zentriert diesen in der Mitte des Bildschirms (Cursor.visible durch Cursor.lockState ersetzt)

**Spieler**

**PlayerController.cs**

Beschreibung:

Das Skript fragt die von Nutzer gedrückten Tasten und die Bewegung der Maus ab und berechnet daraus die Bewegungsrichtung des Spielers.

Mögliche Bewegungsabläufe sind:

* „W“ für das Vorwärtsgehen
* „S“ für das Rückwärtsgehen
* „A“ nach links gehen
* „D“ nach rechts gehen
* „linke Umschalttaste“ für Sprint
* „Leertaste“ für Sprung

Einschränkungen:

Der Spieler wird nicht rückwärts sprinten können.

Das Springen ist nur optional (wird voraussichtlich am Ende der Produktion deaktiviert).

Referenz Skripts:

* PlayerAnimator.cs

Änderungslauf:

Version 1.0 (Erstellung)

* Spieler kann sich bewegen mit W, A, S, D
* Spieler kann sich mit der Maus umsehen
* Spieler kann sprinten
* Spieler kann springen

Version 1.01

* Blickwinkel begrenzt für das nach oben und unten sehen
* Ein rückwärts sprinten ist nun nicht mehr möglich
* Animation Ausführung

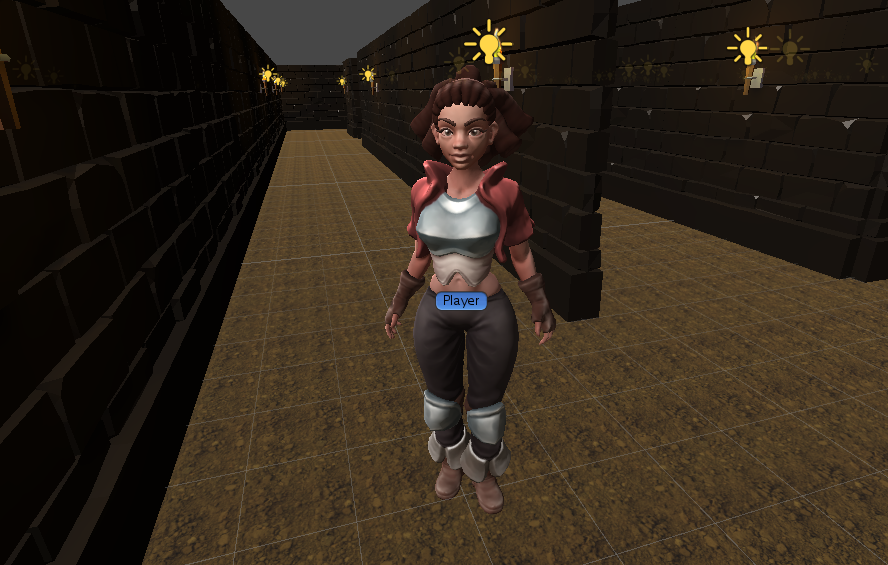


Abb. 1 Spieler inmitten des Labyrinths

**Künstliche Intelligenz**

**EnemyAI.cs**

Beschreibung:

Dieses Skript kümmert sich um das Verhalten des Gegners (KI). Es behandelt das Ablaufen der Wegpunkte und sucht währenddessen den Spieler im Level.

Erkennt der Gegner den Spieler, wird dieser zu seinem neuen Ziel und er nimmt die Verfolgung auf.

Kommt der Gegner an den Spieler nah genug heran, geht dieser zum Angriff über, bis der Spieler besiegt wurde. Der Spieler hat aber auch die Möglichkeit, dem Gegner zu entkommen, indem er es schafft, eine möglichst große Distanz zum Gegner aufzubauen und somit aus seinem Sichtfeld verschwindet. Wenn dies geschieht, nimmt der Gegner das Ablaufen der Wegpunkte wieder auf.

Referenz Skripts:

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

Änderungslauf:

Version 1.0 (Erstellung)

* Läuft Wegpunkte nacheinander ab
* Wegpunkte können zufällig bestimmt werden
* Spielererkennung

Version 1.01

* Verbesserung der Feinderkennung
* Animation Ausführung
* Enemy kann nun den Spieler aus dem Sichtfeld verlieren und geht danach die Wegpunkte wieder ab
* Sichtfeld vergrößert von 60 auf 120
* Sichtweite erhöht von 10 auf 100
* Sektoren hinzugefügt, damit der Enemy den Player schneller findet

**NavMesh**

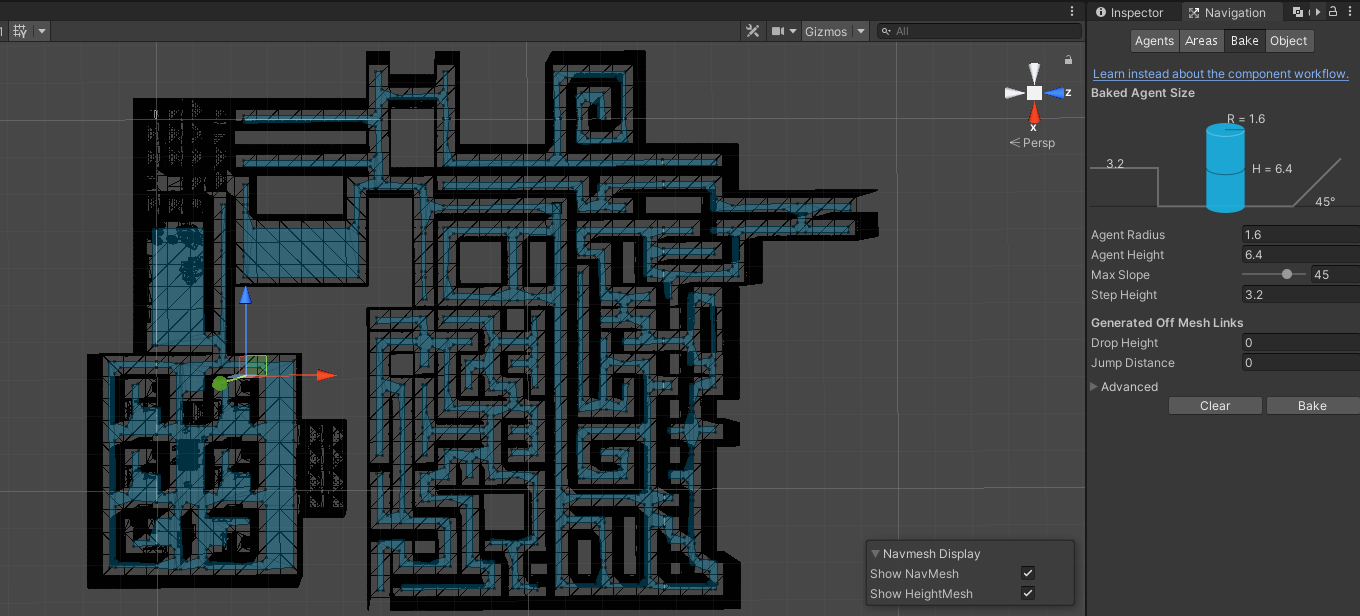


Abb. 2 NavMesh vom Default Level

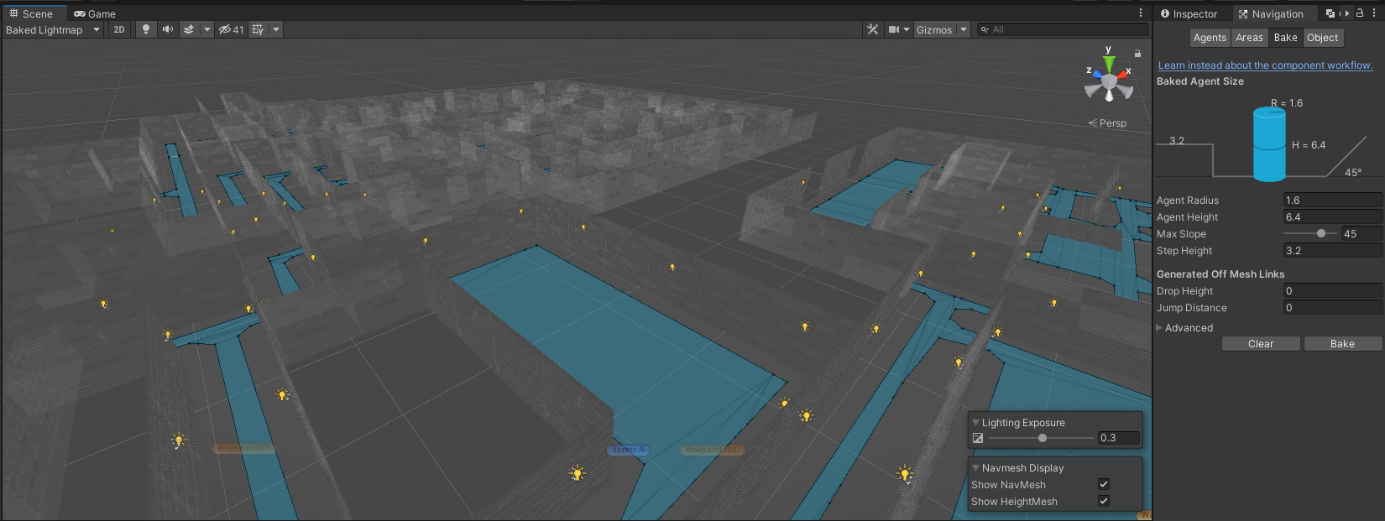


Abb. 3 NavMesh vom Default Level

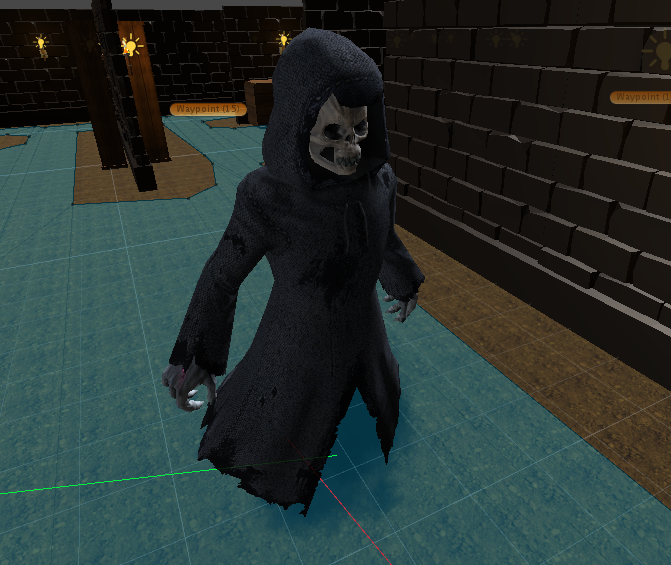
**Gegner (Künstliche Intellignez)**

Abb. 5 Gegner (KI)



Abb. 4 KI beim Ablaufen der Wegpunkte

**Sektoren**

**SectorManager.cs**

Beschreibung:

Die Sektoren zählen in Sekunden, wie lange sich der Player in diesem Sektor aufhält. Eine Methode ermittelt den höchsten Zeitwert aus allen Sektoren.

Den Sektoren sind feste Wegpunkte zugeordnet, die von der KI verfolgt werden. Der Sektor mit dem höchsten Zeitwert übergibt seine Wegpunkte an die KI, wodurch eine höhere Chance besteht, dass die KI den Spieler findet.

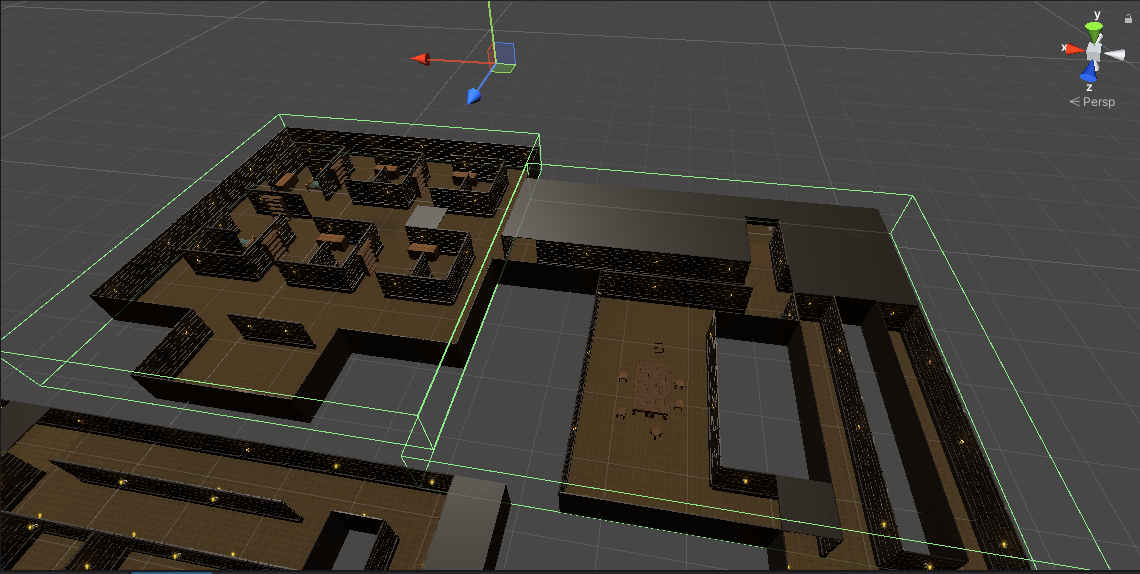
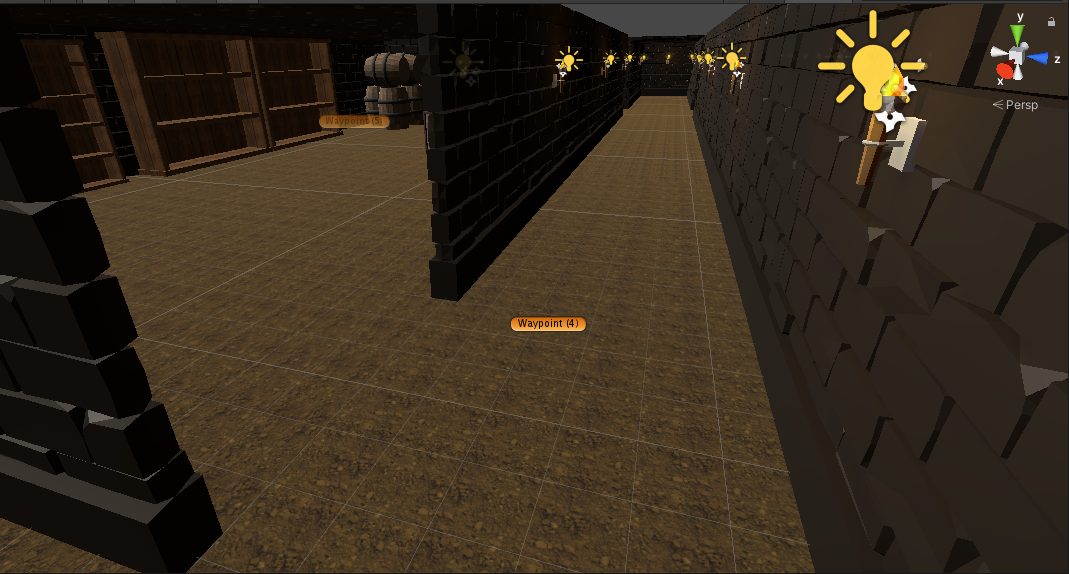


Abb. 6 Sektor A und Sektor B

**Waypoint.cs**

Die Wegpunkte enthalten Information für die KI.

Trifft die KI auf den Collinder eines Wegpunktes, übergibt dieser seine festgelegten Informationen für diesen Wegpunkt.

Informationen:

* Maximaler Sichtwinkel links
* Maximaler Sichtwinkel rechts
* Maximaler Drehgeschwindigkeit
* Verweildauer an diesen Wegpunkt

Abb. 7 Wegpunkt 4

**Performance**

Um die Leistung des Levels zu verbessern, wurden folgende Maßnahmen durchgeführt.

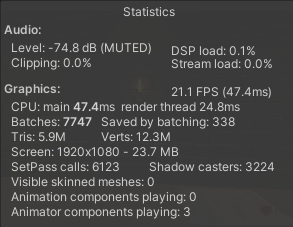
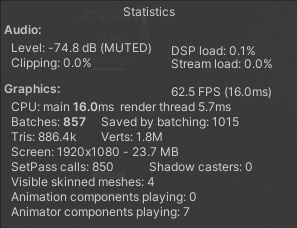
* Für die Lichtquellen z.B. Fackeln an den Wänden wurden *„lightmaps*“ generiert.
* Die Mesh Collinder der Wände wurden durch Box Collinder ersetzt.
* Zum Rendern des Levels wird Occlusion culling eingesetzt, dadurch werden nur Teile des Levels gerendert, die auch vom Spieler eingesehen werden.

Abb. 9 Werte nach der Optimierung

Abb. 8 Werte vor der Optimierung