Spieleprogrammierung
Sommersemester 2025
Name: Philipp Geier

Datum: 25. Mai 2025
Übung 1

Inhaltsverzeichnis

1 Einleitung

Ziel dieses Projekts ist es, ein begehbares 3D-Labyrinth mit Raytracing zu erstellen. Das Projekt ist mit der Gameengine Unity geschrieben und nutzt die eingebaute High Definition Render Pipeline (HDRP) mit bestimmten Einstellungen, um die Szene zu erleuchten. Durch die Nutzung von transparentem, reflektierendem und absorbierendem Material wird die realistische Darstellung von Licht durch Raytracing demonstriert. Das Labyrinth hat zusätzlich spieltechnische Funktionen.

2 Unity mit HDRP

2.1 Übersicht

Unity ist eine 2D und 3D Engine, welche mit C# Skripten arbeitet. Mit HDRP bietet es eine qualitative Render-Pipeline, welche Echtzeit Raytracing unterstützt. Durch den Assetstore sind Unity Projekte auch leicht anpassbar. Mit HDRP werden Raytraced Shadows, Raytraced Reflections und Global Illumination unterstützt.

2.2 Raytracing in Unity

Damit Raytracing in Unity funktioniert, müssen paar Einstellungen aktiviert werden:

- HDRP Wizard: HDRP + DXR (fix all)
- Global Volume:
  - Contact Shadows
  - Screen Space Reflection
  - Screen Space Global Illumination
  - Screen Space Ambient Occlusion
- Sky and Fog Global Volume in Lighting Environment
- Project Settings HDRP-Lightning Maximum Shadows on Screen

3 Softwarestruktur

3.1 Spielfigur - PlayerMovement.cs

Zum Begehen des Labyrinths wird eine Spielfigur benötigt. Diese wird über das PlayerMovement-Skript gesteuert. Mit der Maus kann man sich rotieren und mit WASD sich bewegen. Dabei wird in jedem Schritt mit Hilfe eines Raycasts überprüft, ob eine Kollision mit einer Wand vorliegen würde bevor man sich in eine Richtung bewegt.

3.2 Labyrinth- MazeGenerator.cs

Das begehbare Labyrinth wird vom MazeGenerator-Skript erstellt. Dieses hat mehrere erstellte Prefabs zur Verfügung: Mauerteile mit verschiedenen Materialien, eine Lampe, ein Boden, ein Decken und ein Münzen Prefab. Aus diesen Prefabs wird das Haupt-Labyrinth generiert. Zusätzlich wird ein Anfangsraum generiert und ein Endraum über ein Prefab gespawnt. Zur Generierung des Haupt-Labyrinths liest das Skript die Datei maze.txt ein und das darin enthaltene 2-dimensionale Integer-Array wird ausgelesen. Ist die Datei leer, nicht vorhanden oder das Array nicht entsprechend der Anforderungen, so wird ein Array vom Skript mit den passenden Anforderungen generiert und in die Datei geschrieben. Dafür wird zuerst ein Integer-Array erstellt und mit 1en gefüllt. Dann werden Gänge erstellt mit einer ... todo und mit einer 0 markiert. Schließlich werden noch Schleifen hinzugefügt und ebenso mit 0en markiert. Damit das Labyrinth unterschiedliche Materialien umfasst, iteriert das Skript nun über das Array und tauscht zufällig 1en mit 2, 3 oder 4 aus für die unterschiedlichen Materialien: absorbierend, transparent und reflektierend. Dann iteriert es über die 0en und setzt eine 5 an jede zweite Stelle, um ein Licht darzustellen. Schließlich kommt an die Stelle [width-1,height-2] der Ausgang mit 6. Diese Stelle ist vordefiniert, damit hier das Prefab des Endraums platziert werden kann. Mit dem vorhandenen Array wird es nun durchgegangen und an Stellen mit 1en eine normale Wand platziert, an 2en schwarze Wände, an 3en Glaswände, an 4en Spiegelwände, an 5en Lichter an der Decke und an der 6 die Ausgangsmauer. Dann wird um den Eingang herum ein Anfangsraum aus normalen Wandstücken gebaut und das Ende der Endraum platziert. Schließlich bekommt das Labyrinth noch einen Boden und eine Decke. Zum Schluss wird eine Münze im Labyrinth platziert und der Spieler in den Anfangsraum mit Orientierung in das Labyrinth gespawnt.

3.3 Münze - CoinSpin.cs

Die Münze im Labyrinth schwebt hoch und runter und dreht sich dabei, um auffällig zu sein.

3.4 Münze - CoinPickup.cs

Der Ausgangsraum ist mit einem Exit-Schild an der Mauer versehen und bleibt geschlossen bis der Spieler die Münze einsammelt. Wird diese eingesammelt so ertönt ein Geräusch und die Ausgangstür öffnet sich.

3.5 Licht - LightSelfController.cs

Damit die Lichter im Labyrinth durchgehend aktiv sind, um Performance bei sehr großen generierten Labyrinthen Aufrecht halten zu können, hat jedes Licht im Haupt-Labyrinth einen eigenes LightSelfController-Skript. Dieses erzeugt regelmäßig einen Raycast zum Spieler und kontrolliert, ob eine undurchlässige Mauer trifft und schaltet sich in dem Falle aus. Wird nichts undurchlässiges getroffen, so geht das Licht an.

4 Ausgangsraum

Um das Echtzeit-Raytracing in voller Pracht darzustellen, wurde dieser Raum erbaut. Der Raum zeigt mit einer Lichtquelle je, wie das Licht absorbiert, durchgelassen und reflektiert wird.

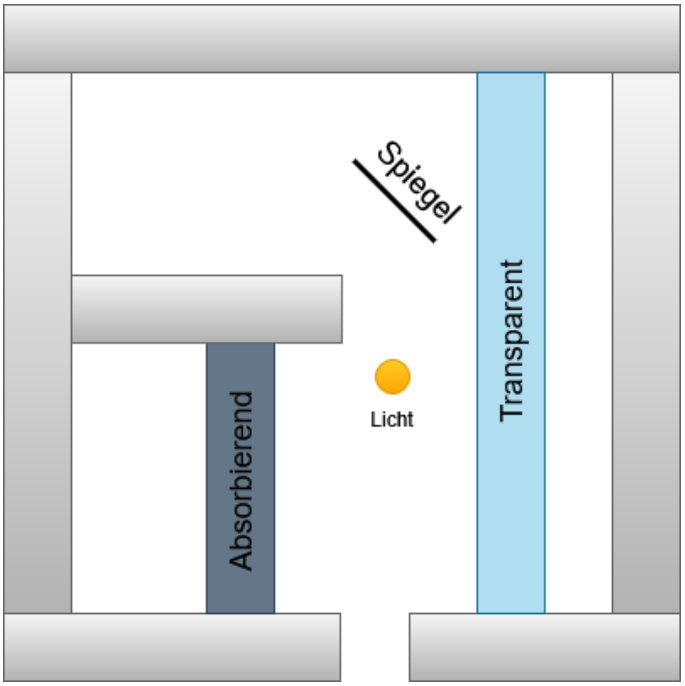

Abbildung 1: Konzept vom Ausgangsraum

5 UML-Diagramm

5.1 Aktivitätsdiagramm

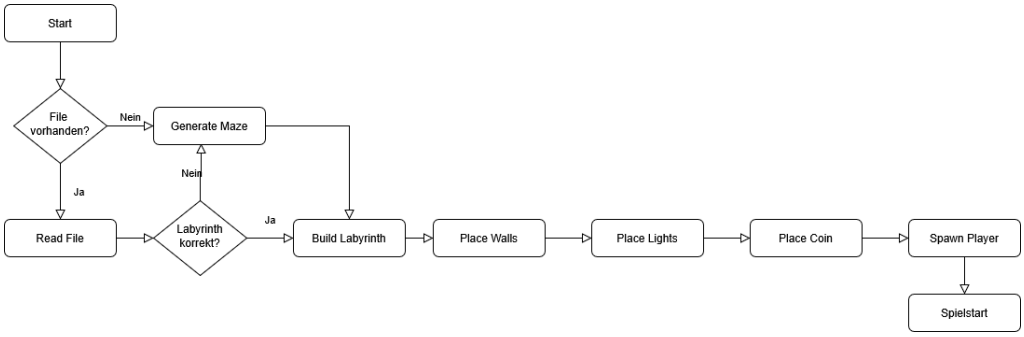

Abbildung 2: Spielstart-Diagramm

5.2 Klassendiagramm

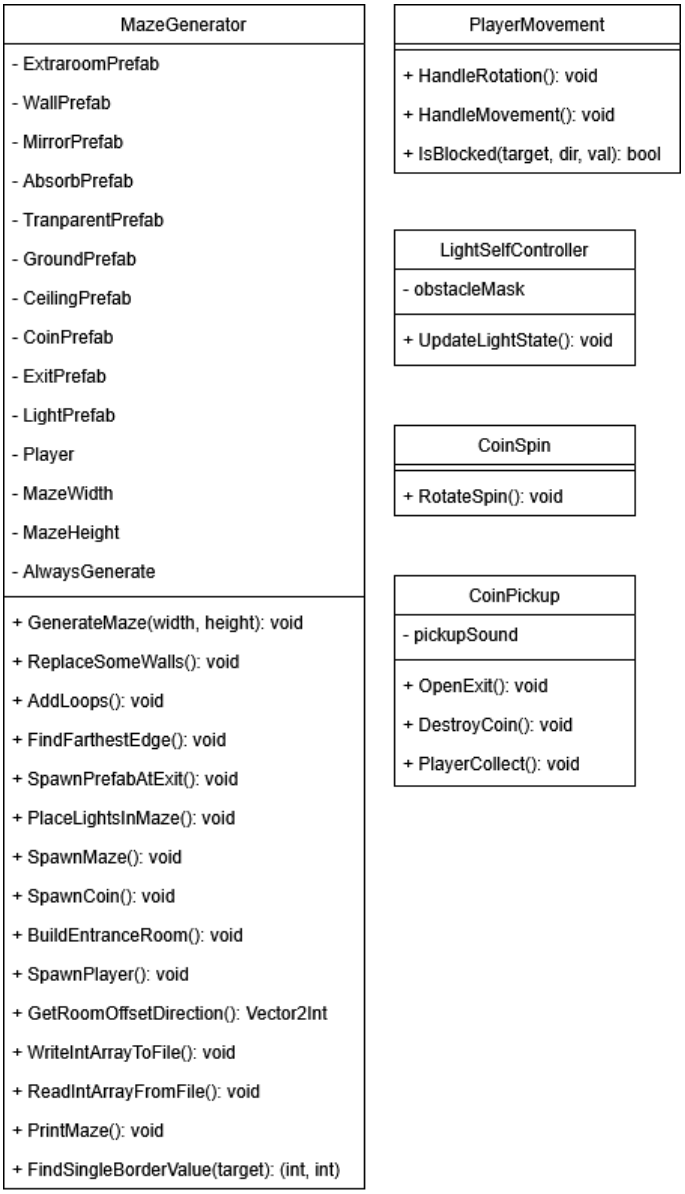

Abbildung 3: Skripte-Diagramm

6 Künstliche Intelligenz

ChatGPT hat als Tool geholfen Ideen für Methoden zu formalisieren und umzusetzen. Es kann gut mit C# und Unity Anfragen umgehen und auch bei komplexeren Anfragen zum Großteil helfen.