1. Überblick

Dieses Projekt implementiert ein begehbares 3D-Labyrinth in Unity, dessen Struktur und Materialien vollständig über eine JSON-Datei definiert werden. Die Wände bestehen aus verschiedenen Materialien (absorbierend, reflektierend, transparent) und werden zur Laufzeit durch das Skript BuildLabyrinth.cs aufgebaut. Die Beleuchtung erfolgt mittels Unity HDRP).

2. Aufbau der Labyrinth-Pläne (JSON)

Die Labyrinthstruktur wird in einer JSON-Datei (z.B. plan2.json oder bigMaze.json) abgelegt. Die Datei enthält folgende Felder:

wallHeight: Höhe der Wände in Unity-Einheiten.

squareSize: Seitenlänge eines Labyrinthfeldes.

wallsX: Array von Arrays, beschreibt alle Wände, die parallel zur Z-Achse verlaufen (also „vertikale“ Wände im Raster).

wallsZ: Array von Arrays, beschreibt alle Wände, die parallel zur X-Achse verlaufen (also „horizontale“ Wände im Raster).

Wandtypen (Materialien)

0: Keine Wand

1: Absorbierend (z.B. matt, kein Licht/reflektiert wenig)

2: Reflektiv (z.B. Spiegel)

3: Transparent (z.B. Glas)

Beispielstruktur

A screen shot of a computer program

AI-generated content may be incorrect.

Interpretation

wallsX[z].array[x]: Wand links von Feld (x, z), verläuft von (x, z) nach (x, z+1), parallel zur Z-Achse.

wallsZ[z].array[x]: Wand unter Feld (x, z), verläuft von (x, z) nach (x+1, z), parallel zur X-Achse.

Die Arrays sind jeweils um 1 größer als die Anzahl der Felder in der jeweiligen Richtung, um die äußeren Begrenzungen abbilden zu können.

3. Das Skript BuildLabyrinth.cs

Zweck

Das Skript liest die JSON-Datei ein und baut daraus das Labyrinth in der Szene auf. Es platziert die Wand-Prefabs an den richtigen Positionen und weist ihnen das passende Material zu.

Aufbau

LabyrinthPlan: Datenklasse, die die JSON-Struktur abbildet.

IntArray: Hilfsklasse für das Deserialisieren der Array-Wrapper in der JSON.

BuildLabyrinth: Hauptklasse, die das Labyrinth aufbaut.

Wichtige Felder

wallPrefab: Prefab für eine Wand (im Inspector zuweisen).

planJson: Die zu verwendende JSON-Datei (als TextAsset im Inspector zuweisen).

absorbentMaterial, reflectiveMaterial, transparentMaterial: Materialien für die Wandtypen (im Inspector zuweisen).

Ablauf

Start():

Prüft, ob eine JSON-Datei zugewiesen ist.

Liest und deserialisiert die Datei in ein LabyrinthPlan-Objekt.

Ruft für beide Wandrichtungen (wallsX und wallsZ) die Methode BuildWalls auf.

BuildWalls():

Iteriert über alle Wände der jeweiligen Richtung.

Platziert für jede Wand mit Wert ≠ 0 ein Wand-Objekt an der korrekten Position.

Weist das passende Material zu.

Wandplatzierung

wallsX (parallel Z):

Position: (x \* squareSize, wallHeight / 2, (z + 0.5) \* squareSize)

Rotation: keine

wallsZ (parallel X):

Position: ((x + 0.5) \* squareSize, wallHeight / 2, z \* squareSize)

Rotation: 90° um Y-Achse

Materialzuweisung

Die Methode GetMaterial(int mat) gibt das im Inspector zugewiesene Material für den jeweiligen Typ zurück.

4. Hinweise zur Verwendung

Prefab- und Materialzuweisung:

Im Unity-Editor müssen das Wand-Prefab und die drei Materialien im Inspector zugewiesen werden.

JSON-Datei:

Die gewünschte Plan-JSON als TextAsset im Inspector zuweisen.

Labyrinthgröße:

Die Größe ergibt sich aus der Länge der Arrays in wallsX und wallsZ.

Eingang/Ausgang:

Offene Stellen (Wert 0) an den Rändern definieren den Eingang und Ausgang.