Übung 1

Einleitung

Dieses Projekt entstand im Rahmen einer Übung in dem Modul *Spieleprogrammierung*. Ziel des Arbeitsblattes war es, sich mit dem Konzept des Ray Tracings vertraut zu machen und dies in einem kleinen Spiel umzusetzen. In dem Spiel soll ein begehbares 3D-Labyrinth erstellt werden, bei dem verschiedene Materialien eingebaut werden sollen, die entsprechend auf das Licht reagieren. Dabei soll sich mindestens ein reflektierendes, ein absorbierendes und ein transparentes Material in dem Labyrinth wiederfinden. Die Struktur des Labyrinths soll aus einer Datei eingelesen werden. Dabei ist das Format der Datei frei wählbar. Optional kann noch ein Objekt platziert werden, welches in dem Labyrinth gefunden werden muss.

verwendete Engine

Ich habe mich bei dieser Aufgabe für die Engine *Unity* in Version Unity 6.1 (6000.1.3f1) entschieden. Dafür gab es mehrere Gründe. Der größte Punkt ist vermutlich, dass ich bereits mit Unity gearbeitet habe und daher die Grundfunktionen bereits kannte. Nachdem ich mich mit den Vor- und Nachteilen von Unity, Godot und Unreal beschäftigt hatte, wurde mein erster Impuls *Unity* zu nutzen nur bestärkt. Unity kommt am besten mit Ray Tracing zurecht. Dort wird über die HDRP bereits eine Ray Tracing Unterstützung angeboten, sodass Schatten und Reflektionen auf Hardwareebene realistisch umgesetzt werden können. Grundlage hierbei ist DirectX Ray Tracing. Auch nützlich ist die Umsetzung in Echtzeit, was das Spielerlebnis angenehmer macht. Zudem war das Integrieren von Ray Tracing in die Szene recht einfach, da *Unity* die Unterstützung anbietet und es somit nur ein paar Einstellungen waren, die angepasst werden mussten. Deutlich aufwendiger wäre es gewesen, das Ray Tracing selber zu implementieren. Bei der Engine *Unreal* wäre es ebenfalls ziemlich einfach gewesen, Ray Tracing in das Projekt einzubinden. Da ich *Unity*, wie bereits erwähnt, schon kannte, habe ich mich für *Unity* entschieden.

Spielbeschreibung

Da der Hauptbestandteil der Aufgabe auf dem Ray Tracing lag, habe ich mich bei der Implementierung auf die Grundformen beschränkt.

In meiner Umsetzung bestehen die Wände des Labyrinths aus verschiedenen Blöcken, die unterschiedliche Materialien darstellen. Dabei ist der Großteil der Wände aus dem absorbierenden Material. Vereinzelt sind Wandteile aus dem reflektierenden und dem transparenten Material eingebaut. Das reflektierende Material hat gleichzeitig auch eine spiegelnde Eigenschaft.

Um das Labyrinth erstellen zu lassen existiert ein 2D-Plan, der die Wände mit den entsprechenden Materialien beschreibt. Ich habe mich dabei für eine CSV-Datei entschieden. In dieser Datei finden sich die Werte *0,1,2* und *3*. Die Zahl *1* beschreibt dabei die Wände aus dem absorbierenden Material. Mit der *2* sind transparente Wände gekennzeichnet und die reflektierenden Wände erkennt man an der Zahl *3*. Zusätzlich existiert die Zahl *0*, welche die Bereiche kennzeichnet, bei denen Blöcke als Boden verteilt werden sollen. Dieser hat keine besonderen Eigenschaften, die in der Aufgabe gefordert waren. Die wichtigste Eigenschaft bei dem Boden ist die Sichtbarkeit. Da die Wände zum Großteil aus dem absorbierenden Material bestehen,

ist es wichtig, dass der Boden gut gesehen werden kann. Dazu exisitert eine Grundbeleuchtung der Szene über das *Directional Light*.

Eine weitere Lichtquelle befindet sich an dem Spieler selber. Dieser ist mit einem Licht ausgestattet, welches eine Art Taschenlampe darstellt. Sie leuchtet immer in Blickrichtung und sorgt so dafür, dass entstandene Schatten nicht als Wände gesehen werden. Außerdem sind durch die Taschenlampe auch die Eigenschaften der reflektierenden und transparenten Materialien deutlich zu erkennen.

Der Spieler ist in diesem Spiel ein violetter Würfel, der sich mit den Tasten *w,a,s,d* steuern lässt. Zusätzlich hat er, wie bereits geschildert, eine Taschenlampe, mit der er den Weg ausleuchten kann.

Um das Spiel ein wenig interessanter zu gestalten, gibt es einen Gegenstand, der in einer bestimmten Zeit gefunden werden muss. Der Spieler hat zwei Minuten (120 Sekunden) Zeit, um den Gegenstand zu finden. Der Gegenstand ist hierbei eine stark leuchtene Kugel. Sobald der Spieler diese Kugel berührt, ist das Spiel vorbei. Der Spieler kann nicht mehr gesteuert werden und es erscheint das Wort ***Gewonnen*** groß in der Mitte des Bildschirms.
Wenn der Spieler innerhalb der zwei Minuten die leuchtene Kugel nicht gefunden und berührt hat, kann der Spieler ebenfalls nicht mehr gesteuert werden und es erscheint das Wort ***Verloren***.

zentrale Module und Klassen

Das Wichtigste waren die Einstellungen in Bezug auf Ray Tracing. Dabei mussten hauptsächlich Einstellungen in den Project Settings angepasst werden. Dabei handelt es sich um die im RTX Rendering enthaltenen Funktionen (Global Illumination, Ambient Occlusion, Shadows und Reflections), die aktiviert werden mussten.
Ich habe vier Skripts selber erstellt: *labyrinth.cs, movement.cs, CameraMovement.cs* und *Timer.cs*.

labyrinth.cs
In dieser Klasse befindet sich die Logik, die das Labyrinth zu Beginn des Spiels erstellt. Dazu wird zuerst die CSV-Datei eingelesen. Der Algorithmus geht dabei jede Zahl aus der Datei durch und entscheidet dann, je nachdem welche Zahl an der aktuellen Stelle steht, welche Eigenschaft der aktuelle Block bekommen soll. Nach der Entscheidung erhält der Block eine Höhe von 4 in der Skalierung, wenn er kein Boden-Block ist. Bodenblöcke haben in der Skalierung in y-Richtung lediglich den Wert 0,1.
Sobald das Labyrinth fertig gebaut ist, werden der Spieler und das Zielobjekt platziert und das Spiel kann beginnen.

movement.cs
In der Klasse movement ist die Steuerung des Spielers untergebracht. Dieser kann mit den Tasten *w,a,s,d* gesteuert werden. Die Tasten *w* und *s* bewegen den Spieler vorwärts bzw. rückwärts in seiner Blickrichtung. Die Tasten *a* und *d* sorgen für die Rotation des Spielers um seine Y-Achse. Somit dreht die Taste *a* den Spieler links herum und die Taste *d* den Spieler rechts herum. Diese Tasten funktionieren allerdings nur, wenn die Variable *ended* auf *false* gesetzt ist.

Sobald das Spiel laut dem Timer vorbei ist, werden die Funktionen dieser Tasten ausgestellt. Zudem ist hier auch eine Funktion untergebracht, die die Kollision des Spielers mit dem Zielobjekt erkennt, das Spiel beendet und die Nachricht *Gewonnen* anzeigt.

Timer.cs
Diese Klasse misst die Zeit, die der Spieler zur Verfügung hat, um das Zielobjekt zu finden. Sobald der Timer die 120 Sekunden überschreitet, wird eine Methode aufgerufen, die das Spiel beendet und die Nachricht *Verloren* anzeigt.

CameraMovement.cs
Mithilfe dieser Klasse wird die Kamera immer so positioniert, dass sie den Spieler im Blick behält. Dies funktioniert, indem die Kamera immer ein wenig hinter den Spieler gesetzt wird und mit der Funktion *lookAt* dafür gesorgt wird, dass sie den Spieler immer im Blick behält.

Herausforderungen und Lösungen

Wie bei jedem Projekt, gab es auch hier einige Herausforderungen. Das erste große Problem war die Grafikkarte in meinem Laptop. Sie erfüllt leider nicht die Voraussetzungen, um das Labyrinth mit Ray Tracing auszuleuchten. Ich habe also eine Weile damit verbracht, die benötigten Anwendungen auf einem anderen Laptop zum Laufen zu bringen.
Das nächste Problem entstand beim Setzen der verschiedenen Lichtquellen, sodass die Materialien ihre Eigenschaften im Licht zeigen können. Nachdem ich zwei Lampen gesetzt hatte, hat der Laptop schon schwer gearbeitet. Nach der dritten Lampe hat er sich bei jedem Starten des Spiels aufgehangen. Daher habe ich dem Spieler eine Taschenlampe gegeben, sodass nur eine Lichtquelle erforderlich ist.