Computergrafik und Bildverarbeitung – Abschlussprojekt  
Ausarbeitung

*Florian Tobusch, Nico Daßler, Elias Miorin*

**Escape The Maze**

**1. Spielidee**

Grundlegende Spielidee ist die Flucht aus einem Irrgarten. Um zu gewinnen, muss der Spieler einen Weg aus dem Irrgarten finden. Das primäre Hindernis ist der Irrgarten selbst. Allerdings wird die Flucht durch Portale, niedrige Decken und blockierte Durchgänge erschwert. Als zusätzliche Motivation soll ein Bewertungssystem dienen. Die Leistung des Spielers wird, abhängig von der Geschwindigkeit mit der er das Labyrinth durchläuft, unterschiedlich gut (oder schlecht) bewertet. Die Atmosphäre soll düster sein, was vor allem durch die Musikauswahl erreicht wird.

**2. Spielelemente (Implementierung)**

**2.1 Spielfigur (FirstPersonCharacter-Blueprint)**

Als Grundlage für die Implementierung unseres Spiels wurde das Template „First Person“ verwendet. Alles was mit der Spielfigur selbst zu tun hat findet sich im Blueprint „FirstPersonCharacter“ wieder. Hier waren bereits die Waffe und einige Basisbewegungen des Characters implementiert. Von uns wurde noch zusätzlich Rennen (Run) und Ducken (Crouch) implementiert.

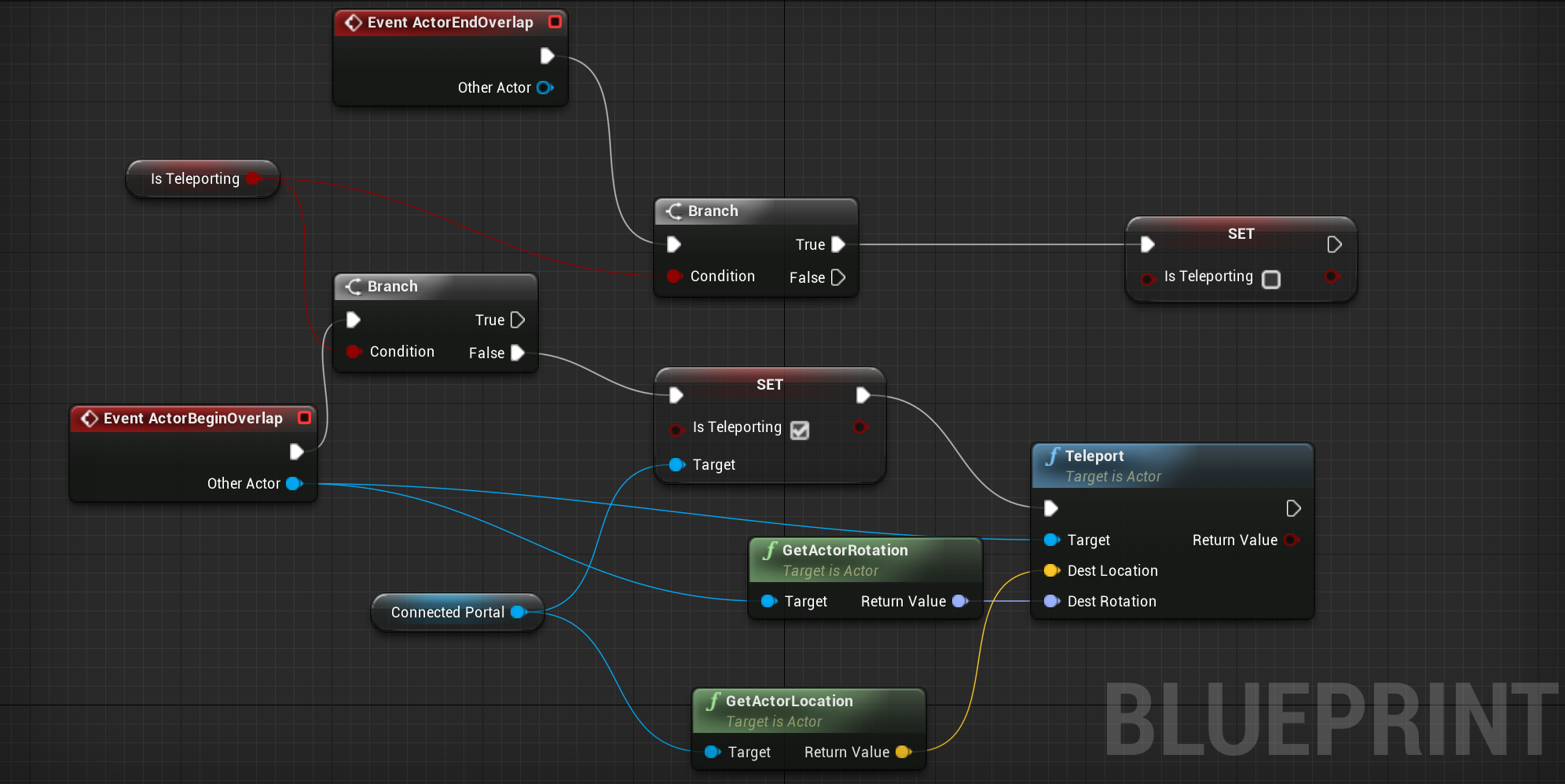
|  |  |
| --- | --- |
| In den Project-Settings werden die Bewegungen und Actions der Spielfigur and Eingaben über die Tastatur und die Motioncontroller gebunden.  Im Blueprint des Characters stehen Events zur Verfügung, mit denen dann entsprechende Aktion ausgeführt werden können. Beispielsweise beim drücken der Shift-Taste wird das Event „InputAction Run“ ausgelöst was die Lauf-Geschwindigkeit des Spielers erhöht. |  |

Implementierung von Sprinten (Run):   
Die Geschwindigkeit (Walk Speed) wird auf das doppelte der normalen Lauf-Geschwindigkeit erhöht. Wenn sich der Character bereits im Crouching-Modus befindet, dann kann er nicht mehr sprinten.

Implementierung von Crouching:  
Die Geschwindigkeit (Walk Speed) des Characters wird auf ein Drittel reduziert. Zusätzlich wird auch das Sichtfeld des Characters nach unten verschoben, bzw. nach dem Crouching wieder zurück in ursprüngliche Position verschoben.

**2.2 Portale**

**((**Physik der Portale -> Waffe kann „durchschießen“))

*  
Blueprint für ein Portal*

Im Spiel sind immer zwei Portale miteinander verlinkt. Betritt ein Actor (z.B. Character oder Projektil der Waffe) die Triggerbox des Portals wird der Actor zum verlinkten Portal teleportiert. Im Start- und Zielportal muss das jeweils verlinkte Portal manuell ausgewählt werden, ansonsten ist ein Rücksprung des Actors nicht mehr möglich.

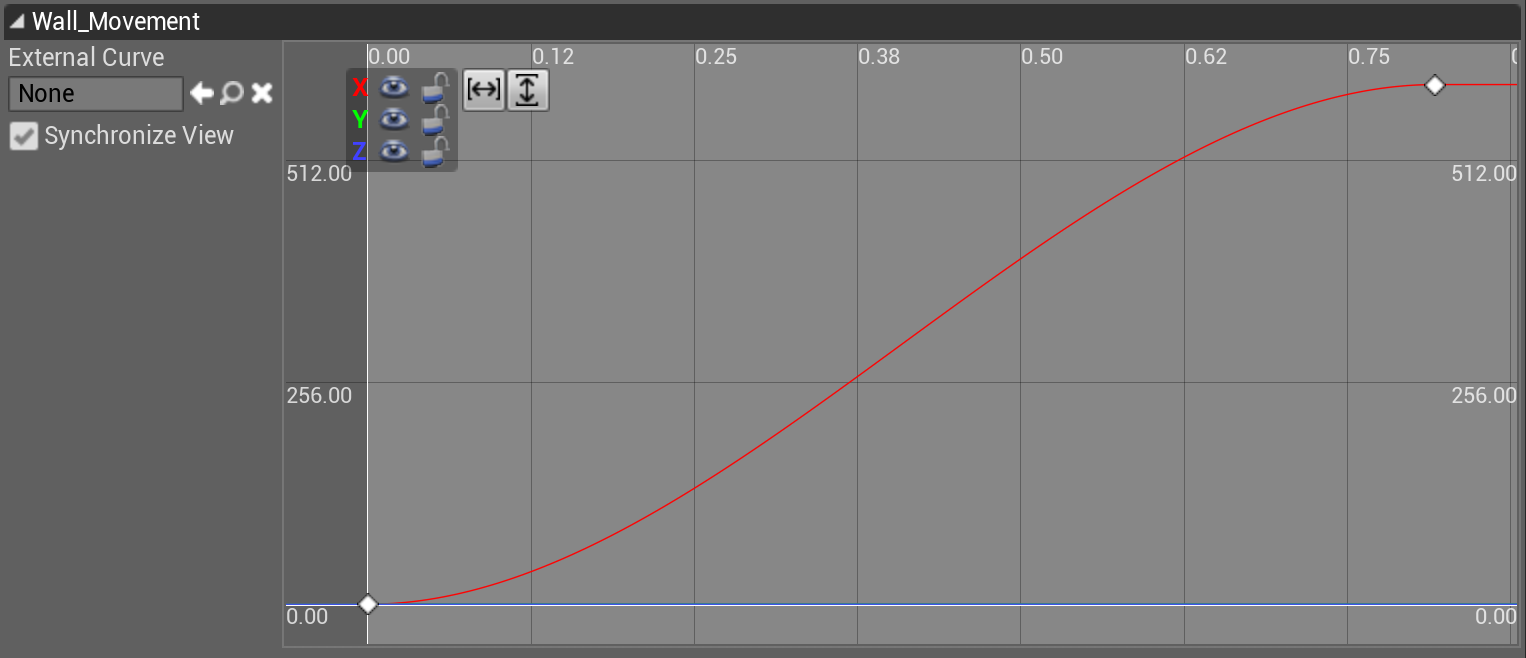
Implementierung des Teleportieren im Blueprint:  
Beim Betreten der Triggerbox wird die Position des Zielportals geholt und der UE4-Funktion „Teleport“ als Parameter für die „Destination Location“ übergeben. Die Rotation des Actors wird beibehalten und nicht an die entsprechende Rotation des Zielportals angepasst.  
Ein sofortiges zurück teleportieren beim Betreten des Zielportals wird durch den Boolean „isTeleporting“ vermieden. Erst wenn der Actor die Triggerbox des Zielportals verlässt, wird auch der Boolean wieder auf false gesetzt. Bei erneutem Betreten der Triggerbox des Zielportals, wird der Actor wieder zurück teleportiert.

Animation der Portale über „Particle System“  
Die Portale besitzen neben der Triggerbox noch ein Particle System. Hierbei wurde verschiedenen Texturen das selbe Leuchten, wie den Wänden hinzugefügt. In der Mitte des Portals befindet sich eine Rauchwolke, der eine Rotation hinzugefügt wurde.

**2.3 Wände**

Kern eines Labyrinths sind die Wände. Hierbei gibt es folgende Wände:

Sliding Wall (Blueprint):  
Beim Betreten einer Trigger-Box wird die Wand verschoben und somit das Tor geöffnet. Wird die Trigger auf der anderen Seite wieder verlassen, schließt sich die Tür wieder. Damit die Tür nicht von der geschlossenen Position in die geöffnete Position springt wurde eine Timeline verwendet. Mit der Timeline kann über den Verlauf der Zeit die Position der Wand langsam verändert werden.

*  
Timeline für die Sliding Wall*

Maze Wall (Static Mesh):  
Damit für das gesamte Labyrinth Wände in einheitlichem Design verwendet werden, wurde hierfür ein Static Mesh erstellt. Änderungen an diesem Static Mesh wirken sich auf alle bereits im Labyrinth platzierten Wände aus. Damit der Spieler nicht durch Wände laufen und auch nicht hindurch schießen kann, musste dem Static Mesh noch eine Collision Box hinzugefügt werden.

Leuchten der Wände (Material):  
Das Leuchten (Glow) der Wände wurde über das Asset „Material“ realisiert. Durch die Anwendungen verschiedener mathematischer Operationen auf eine Textur-Koordinate wird ein Farbwert erstellt, der anschließend an die Emissive Farbe ausgegeben wird. Verschiedene Variablen ermöglichen eine individuelle Anpassungen des Materials. Für die Labyrinthwände wurde ein heller Blauton verwendet, für die Kugel der Waffe ein Rotton und die Slidingwall leuchtet Grün.

**2.4 Musik/Sounds**

Es sind zwei verschiedene Arten von Sounds im Spiel vorhanden. Zum einen ein Soundeffekt beim Abfeuern der Waffe, der vom Standardprojekt *First Person* aus der Unreal Engine übernommen wurde. Der andere Sound ist eine im ganzen Spiel vorhandene Hintergrundmusik. Als Lied wurde ein Track aus dem Portal-Soundtrack (ein Spiel, von dem wir uns inspirieren haben lassen) verwendet:

<https://www.youtube.com/watch?v=aqGXCQ_5WOc>

((Implementierung kurz beschreiben.))

**2.5 Timer und Rating**

Um dem Durchlaufen des Labyrinths noch den Charakter eines Spiels hinzuzufügen, bzw. das Durchlaufen mit anderen Spielern vergleichbar zu machen, wurde noch ein Timer hinzugefügt. Im Blueprint des FirstPersonCharacters befindet sich dafür ein Counter, der jede Sekunde um eins erhöht wird. Über „Widget Blueprints“ wird die Zeit eingeblendet.

Am Ende des Labyrinths befindet sich eine Trigger-Box, die den Timer stoppt und die aktuelle Zeit mit einer kleinen Bewertung ausgibt. Bei sehr guten und guten Zeiten wird „Impressive!“ oder „Ok!“ ausgegeben. Für schlechte Durchläufe wird nur „Embarrasing!“ eingeblendet.

**3**. **Anleitung**

Das Spiel wird mit einer HTC Vive gespielt, jedoch nicht mit den beiliegenden Controllern gesteuert, sondern klassisch mit Maus und Tastatur. Die Maus dient dabei zur Ausrichtung der Waffe und der Spielfigur. Mit den Tasten W, A, S, D wird die Spielfigur bewegt. Die linke Umschalt-Taste wird benutzt um zu Sprinten (gedrückt halten zum Sprinten). Die linke Steuerung-Taste wird benutzt um sich zu ducken. Um die Waffe abzufeuern wird die Linke Maustaste verwendet.

Der Spieler startet auf einer Plattform (vgl. Abbildung x). Es gibt nur einen Weg: In das Labyrinth. Die Zeit beginnt sofort. Im Labyrinth wird der Spieler auf die oben beschriebenen Hindernisse treffen.

**4. Bewertung/Fazit**

Mit Game-Engines, wie der Unreal Engine, lassen sich eigene Spiele schnell konstruieren. Die tatsächlichen Herausforderungen liegen in der kreativen Arbeit, die geleistet werden muss. Wir haben schnell gemerkt, dass es schwierig ist eine Spielidee mit einer stimmigen Atmosphäre und passenden Spielmechaniken (Steuerung etc.) umzusetzen. Unser primäres Ziel war es möglichst viele Funktionen einer Game-Engine auszuprobieren. Wir haben Physik (Waffe, Portale, einstürzende Mauern), Sound (Hintergrundmusik), Beleuchtung (Leuchtende Wände), UI (Timer und Bewertung), Steuerung/HMI (Sprinten, Ducken etc.) in unserem Spiel untergebracht.

Leider sind auch einige Kleinigkeiten auf der Strecke geblieben. Zum Beispiel ist die Austrittsrichtung der Kugeln aus der Waffe nicht wie man es intuitiv erwartet. Das gleiche gilt für die Blickrichtung nach dem Teleportieren der Spielfigur. Auch ist die Umsetzung mit der HTC Vive nicht optimal gelungen. Der Spieler kann, wenn er sich vor eine Wand stellt und den Kopf nach vorne bewegt, etwas durch die Wand hindurchsehen.

Wir hatten großen Spaß bei der Umsetzung unserer Spielidee.