Abgabe
9.3.3 Game Physics
Programmierspezialisierung GD1017

“Quantenverschränkung”

Arne Haefs
Matrikelnummer: 222305
20.01.2020

Eidesstattliche Erklärung

Eidesstattliche Erklärung des Verfassers:

Hiermit versichere ich, Arne Haefs(Matrikelnummer: 222305), das ich die Beiliegende Arbeit mit dem Titel “Quantenverschränkung”, sowie den praktischen Teil im Github Repository: https://github.com/ArHaefs/GamePhysics selbst geschrieben und gecoded habe.

Berlin, 20.01.2020 Unterschrift: Arne Haefs

Inhaltsverzeichnis

Projekt Dokumentation

Geplantes Projekt

Für die Abgabe bestand meine Projektidee aus einem Spiel, in welchem sich der Hauptcharakter nicht selber bewegen kann und stattdessen sich direkt mit anderen Objekten verbinden kann und dadurch deren Bewegung teilen kann.

In meinem ursprünglichen Plan war es vorgesehen, dass ich das Projekt in Unity und einer dreidimensionalen Umgebung umzusetzen, entgegen der Empfehlung es zweidimensional zu umzusetzen. Weiterhin habe ich mich entschlossen die in Unity enthaltene Physik Komponente zu nutzen um mir meine Arbeit und meine Modifikationen zu erleichtern.

Diese Modifikationen waren:

Quantenverschränkung: Die Fähigkeit des Spielers sich mit einem Objekt auch über distanz zu verkuppeln. Hierbei würden alle Kräfte die auf die beiden Objekte wirken zusammengezählt werden und die Bewegungen der beiden abgeglichen und synchronisiert werden.
Reibungswiderstand: Im Rahmen geplanter Bodentypen wollte ich ebenfalls ein System einbauen welches Reibung auf einfache Art und Weise nachstellt.

Geplante Objekte waren:

Bewegbare Objekte: Objekte welche von ihrer Umgebung bewegt werden und so dem Spieler die Möglichkeit geben sich selber zu bewegen.
Selbstbewegende Objekte: Objekte welche sich von ihrer Umgebung unabhängig bewegen und so dem Spieler andere Möglichkeiten eröffnen.(z.B “Auto” welches zwischen zwei Nahen Punkten hin und her fährt und so dem Spieler durch verkuppeln und ablösen Vorwärtsbewegung ermöglicht)
Fließbänder: Bodenplatten welche jedem Bewegbaren Objekt auf ihnen Krafteinwirkung in eine bestimmte Richtung gibt.
Schlamm: Bodenplatten welche dem Spieler eine stärkere Reibung verpassen und ihn so abbremsen, es ihm erlauben die Objekte die ihn bewegen zu manipulieren.

Impuls Objekte: Objekte welche keine gleichförmige Bewegung haben sondern kurze starke Kraftstöße aussenden um sich dann langsam zurückzuziehen. Würden es geschickten Spielern erlauben diese Kraftstöße auf sich oder andere Objekte zu übertragen)
Fallgruben: Löcher im Boden durch welche man herabfallen und verlieren kann. Können überwunden werden indem man sich mit einem Objekt verbindet, welches auf festem Boden ist und so einen Fall verhindert.

Ausführung der Planung

Die Ausführung der Planung erwies sich als unzufrieden und brachte mehrere größere Probleme ans Licht.

Selbstüberschätzung: Zwei Dinge erwiesen sich als besonders Zeitfressend. Die Entscheidung auf 3D zu wechseln, welche meine Arbeit nur verkomplizierte. Und die Entscheidung mit Unities eigner Physik Komponente zu arbeiten. Welche, während gut geeignet für normale Spiele, bei diesem Projekt ihre Grenzen zeigte, besonders bei der Verarbeitung von Bewegung an stellen welche ich nicht erreichen konnte.
Fehleinschätzung der Benutzten Software: Wie oben schon angesprochen war eines meiner größeren Probleme Unitys eingebaute Physik Komponente, gegen welche ich öfter vorgehen musste und welche der Grund vieler Bugs war.
Fehleinschätzung der gebrauchten Zeit: Mit der Bewerbungsphase gleichzeitig meine Zeiteinteilung hat an einigen Stellen wertvolle Zeit die ich für das Projekt gut hätte gebrauchen können für andere Dinge geopfert.
Zusammenwirken aller oberen Dinge: Welches dazu führte dass ich das Ende der Arbeitsphase damit verbracht habe verzweifelt aus meinem verbuggten Code etwas präsentierbares zu machen und am Ende für Funktion über Komplexität gegangen bin. Und zum Teil Features schneiden musste weil ich keine Zeit mehr hatte die Bugs zu Fixen.

Ergebnis

Von den vorher geplanten Features haben es bis in die Abgabe die folgenden Überlebt:

Quantenverschränkung: Es ist möglich sich mit Objekten zu verbinden und ihre Bewegung zu teilen, auch wenn es zu einigen Bugs kommt, besonders wenn es zu Kollisionen und ähnlichem kommt.
Bewegbare Objekte: Nicht bugfrei und ohne komplexere Kollisionen aber funktionierend.

Fließbänder: In der Lage Spieler und Bewegbare Objekte zu transportieren.
Fallgruben: Mit einem rudimentären Respawn-System.

Zukunftsausblick

Für das Projekt direkt gibt es nur einen Zukunftsausblick. Ein mahnmal an mich selber und hoffentlich ein Lernerlebnis, welches mir hilft zukünftige Projekte realistischer anzugehen. Die Idee selber jedoch hat immer noch potential und wenn ich jemals die Zeit finde sehe ich zwei mögliche Wege:

Weniger Komplexität mehr Design: Ein 2D top down Spiel, welches im Gegensatz zu meinem Prototypen Reibung und Kollisionen mit auch komplexeren Berechnungen und selbstgeschrieben Physik Komponente beinhalten würde.
Mehr Zeit nehmen und diesmal Gut: Das wichtigste wäre sich mehr Zeit zu nehmen und eine eigene Physik Komponente zu schreiben, welche für Grundauf auf die Quantenverschränkung ausgelegt ist und als solche es erlaubt die Gegenseitig beeinflusste Bewegung besser zu berechnen.

Code Dokumentation

Dank der last-minute Axt welche ich ich dem Projekt angetan habe um es in ein Funktionsfähigen Zustand zu bekommen ist mein Code nun recht übersichtlich. Die vier Skripts, welche Interagieren sind der LinkedMovementManager, welcher für die Quantenverschränkung zuständig ist. Der MoverFloor, welcher nicht nur die Fließbänder, sonder jeden Boden darstellt und vor der Axt auch Funktionen für den Schlamm hatte. Den MoveableCube, mein Bewegungs Script, welches jedes Objekt behandelt welches sich bewegt, sowohl Spieler als auch Bewegbare Objekte. Und zu guter letzt das Player Script, welches nur geringe Koordinatoren Rollen hat, sowie der Identifizierung des Spielers dient.

MoverFloor

Das MoverFloor Script ist bewusst simpel gehalten das es schnell in viele Richtungen adaptiert werden kann. Es weiß welche GameObject mit ihm kollidiert sodass es mit ihnen interagieren kann. Und gibt Beschleunigung an das MoveableCube Objekt weiter.

MoveableCube

Der Moveable Cube enthält eine einfache Logik sich selber zu bewegen. Die Movement Variable existiert, sodass andere Objekte Einfluss nehmen können ohne direkt auf den Rigidbody zugreifen zu müssen. Hierbei filtert der MoveableCube etwas, sodass er nicht von zwei Fließbändern den gleichen Vector bekommen kann. Dies ist um zu verhindern das auf den Übergängen von Fließbändern es zu einer zu hohen beschleunigung kommt. Weiterhin kennt der Cube welches Fließband ihn gerade beeinflusst, sodass er ein stoppsignal weitergeben kann wenn das nächste Fließband übernimmt. Sollte der Cube mit einem anderen Objekt verlinkt sein, berechnet er seine Bewegung nicht mehr selber, bekommt aber noch die Bewegung direkt mitgeteilt, sodass Krafteinwirkungen vom herausgeschnittenen Schlamm richtig berechnet werden können.

Weiterhin kann der Cube überprüfen ob er sich gerade auf festen Boden befindet(sodass ein eventuell verlinktes Objekt schweben kann) und kann die Schwerkrafteinwirkung auf sich selber ausschalten, sollte das mit ihm verlinkte Objekt auf festem Boden sein.

Zu guter letzt, fällt der Cube aus der Welt respawnt er an seinem Herkunftsort und setzt all seine variablen zurück.

LinkedMovementManager

Das größte der Scripte und verantwortlich für die Quantenverschränkung in dem es die Bewegungen der beiden Verschränkten Objekte berechnet. Das Skript beginnt mit der überprüfung ob eines oder beide der Objekte auf festen Boden sind, sodass, sollte dies nicht der Fall sein, die Schwerkraft Einwirkung auf das schwebende Objekt ausgestellt werden kann.
Danach folgt die Verarbeitung der Bewegung des letzten Updates. Da Kraft Einflüsse und Bewegungen zwischen aufrufen der Update-Funktion sind, macht es keinen Sinn die Kraft Einflüsse dieser Runde auf ihre Bewegung zu untersuchen. Hierbei werden die Geschwindigkeiten der beiden verglichen auf allen drei Axen und für jede Achse wird der Wert am nächsten zu null gewählt. Unter der Prämisse, dass irgendetwas dort die Bewegung erschwert oder blockiert und dieses sich auch auf das Objekt auswirkt.

Danach wird die Bewegung für den nächsten Zug vorbereitet in dem die Kräfte die auf die beiden Objekte wirken zusammengerechnet werden und dann auf die Objekt angewendet werden.

Player

Zu guter Letzt das Player Skript, welches nur zwei Aufgaben hat. Im Falle, dass das Objekt das zurzeit mit dem Spieler verbunden ist respawned es abzutrennen. Damit durch den Respawn nicht Dinge durcheinander gebracht werden. Und das managen der Verknüpfung zweier Objekte in dem auf einen Mausklick ein Raycast abgefeuert wird und auf einem Rechtsklick die Verbindung unterbrochen wird. Außerdem wird über das Vorhandensein dieses Skripts der Spieler identifiziert.