Virtual Reality für spielerisches Erkunden des HTWK Campus – Entwicklung einer Anwendung für die Oculus Quest mit Unity

Valentin Merz

7. Dezember 2021

Inhaltsverzeichnis

1 Einleitung 1

2 Hintergrund 2
- 2.1 Virtual Reality . . . 2
- 2.2 Entwicklung in Unity . . . 2
- 2.3 Oculus Plugin . . . 2

3 Konzeption 3
- 3.1 Nutzungsszenario . . . 3
- 3.2 Resultierende Anforderungen . . . 3

4 Technische Umsetzung 4
- 4.1 Zerlegen der Gebäude . . . 4
- 4.2 offhandgrab / zerlegen mit beiden Händen . . . 4
- 4.3 Nutzbarkeit der exisstierenden 3D Modelle . . . 5
- 4.4 Erzeugen eines aufschlussreichen Innenlebens . . . 5
- 4.5 Performance . . . 5

5 Ausblick auf mögliche Anwendung und Erweiterung 6

6 Fazit 7

7 Anhang 8
- 7.1 Glossar . . . 8

8 Notizen 9

1 Einleitung

afd

2 Hintergrund

2.1 Virtual Reality

Begriffserklärung Ältere Verwendung verglichen mit heute Abgrenzung oder Übergang zu MR AR Herausforderungen für Anwendungen in dem Medium

2.2 Entwicklung in Unity

Was ist eine GameEngine
    Entity Component System

2.3 Oculus Plugin

Was für ein Design Pattern verwendet der Grabber? Was für eins der Player Controller? Was tut der OVR Manager?

3 Konzeption

3.1 Nutzungsszenario

Sich eine Übersicht an dem Campus zu verschaffen, sowie Informationen herauszusuchen kann eine ermüdende Aufgabe sein. Weil sie das ist gibt es dann auch Studierende, die nicht alle Institutionen kennen, die ihnen helfen könnten oder nicht alle Orte innerhalb des Campus kennen, die für sie interessant sein könnten. Indem die Tätigkeit des sich Informierens zu etwas Spielerischem wird ist sie, wenn auch nicht so schnell auch nicht erschöpfend und die Studierenden entdecken zumindest einige der für sie hilfreichen Informationen.

3.2 Resultierende Anforderungen

Da diese Anwendung für Menschen gemacht ist, die diese vermutlich nicht mehrfach nutzen ist es essentiell, dass es keine großen Hürden gibt mit, die erlernt werden müssen und dass Mechaniken, die nicht für die meisten Menschen offensichtlich sind innerhalb der Anwendung erklärt werden.

Die Gebäude der HTWK müssen klar erkennbar sein und die Information die man daraus ersehen kann muss korrekt sein.

Vollständigkeit ist nicht notwendig und auch nicht machbar.

4 Technische Umsetzung

4.1 Zerlegen der Gebäude

Eine Kernfunktion des Zerlegens sollte sein, dass die Gebäude aus den Einzelteilen einfach und richtig wieder zusammenbaubar sind. Dafür sollten die Einzelteile, wenn sie nah an ihren Verbindungsstücken sind wie Magnete passend aneinander kommen.

Für dieses aufeinander Zugehen der Teile werden immer Paare betrachtet, die im zusammengesetzten Modell eine sie verbindende Fläche haben.

Überprüfen distanz collider vs know it + Identifizieren durch in der Szene übergeben vs tags

Abprallen und das verhindern dessen

Greifen und Hochheben von ganzen Gebäuden

Joints und daraus resultierendes Zittern vs:
Mehrere Einzelteile, die schon zusammen sind sollen an einer Stelle genommen und bewegt werden können und sich dabei wie ein Ganzes verhalten. Um das zu erreichen versuchte ich als erstes fixed joints zu verwenden, über die das Unity Manual folgendes schreibt:
"Restricts the movement of a rigid body to follow the movement of the rigid body it is attached to. This is useful when you need rigid bodies that easily break apart from each other, or you want to connect the movement of two rigid bodies without parenting in a Transform hierarchy."
Anhand dieser Beschreibung erschienen sie wie eine hervorragende Lösung, bei der Verwendung erschien dann allerdings ein Verhalten, dass sie als alleinige Lösung für unbrauchbar erscheinen ließ: Wenn man ein Ende greift und bewegt erscheint nicht der gesamte Block als ganzes bewegt zu werden, sondern alle mit dem primär gegriffenen Einzelteil verbundenen Stücke ruckelten hinterher, was unangenehm anzuschauen war und nicht dem gewünschten Verhalten entsprach. Sie konnten dabei auch auseinanderbrechen und in der Luft hängenbleiben. Das Glossar erwähnt tatsächlich etwas derartiges:
"Fixed Joint: A joint type that is completely constrained, allowing two objects to be held together. Implemented as a spring so some motion may still occur."

Mein nächster Lösungsansatz war die Greif Funktion aus dem Oculus Plugin für meinen speziellen Fall abzuwandeln.
Versuch des abänderns von Oculus scripts um mehrere Objekte hochzuheben

4.2 offhandgrab / zerlegen mit beiden Händen

Wenn mit der einen hand gegriffen wird und mit der anderen verbundene Teile gegriffen werden was soll passieren?

Ich sehe drei Design Möglichkeiten, von denen ich nicht sicher bin, welche die Beste ist: 1. Teilen in der Mitte 2. Teilen neben der neuen Hand 3. alles zwischen den Händen fällt

Es soll sich natürlich anfühlen und vorhersehbar sein, nachdem man damit schon mal interagiert hat. Wenn es interessant sein kann, oder einen Spaß Moment des Zerstörens haben kann wäre das gut. Option 3 erscheint am riskantesten, aber als hätte es viel Potential. Ein aufteilen in der Mitte klingt erst mal sicher, aber vielleicht fühlt es sich sehr ungenau an. Direkt neben der neuen Hand klingt exakter, aber vielleicht bei der ersten Erfahrung unerwarteter.

Wie ist das machbar?

das Gegriffene Gebäude als Graph zu betrachten liegt nahe, da es Einzelteile sind, bei denen immer Verbindungen zwischen zwei Teilen bestehen. Wenn dieser Graph ein Pfad ist gibt es keine Komplikationen ihn aufzuteilen, aber wenn er stärker vernetzt ist erscheint es nicht mehr so offensichtlich, was alles an dem neu gegriffenen hängt. Vielleicht ist die Lösung nur die Teile, die nur durch den gegriffenen verbunden sind als an ihm hängend zu betrachten. Derzeit erscheint es mir aber sinnvoller räumliche Nähe mit einfließen zu lassen, oder Nähe im Graphen einfließen zu lassen: Falls ein Teil mit keinem Kürzeren Weg mit der Ursprungshand verbunden ist, als der Weg über das neu gegriffene Teile wäre dieses als am neuen Hängend zu betrachten. Intuitiv erscheint es mir sinnvoll den Gleichheitsfall in diesem Vergleich der neuen Hand zuzuordnen, aber sicher bin ich damit nicht.

Falls man räumliche Nähe heranzieht wäre es wichtig tatsächlich die räumliche Mitte der Teile zu vergleichen und nicht transforms, die an ihren Rändern liegen. Auch ist dabei dann die Frage, ob man die Verbindungsstruktur immer noch untersuchen muss, weil es in manchen Fällen sonst zu Problemen könnte, oder ob das dann als alleiniges Kriterium reichen wird.

Derzeit haben die Einzelteile kein Konzept ihres ganzen und speichern nur ihre direkt verbundenen, worüber sich die das gesamte Netz abfragen lässt. vielleicht wäre es nützlich einen Wert zu speichern, der die Distanz vom Einzelteil zu der Hand, die es direkt oder indirekt greift, festhält, aber wenn man diesen nur in der Situation benötigt, in der zwei Hände greifen würde dies ja nur ein wenig aufwand sparen im Vergleich dazu ihn jedes mal zu bestimmen. Eine kürze Weg Suche klingt aber recht aufwändig, wobei die Graphen ja meist recht klein sein werden, also ist es vermutlich nur aufwändig zu implementieren, falls überhaupt.

4.3 Nutzbarkeit der exisstierenden 3D Modelle

asdf

4.4 Erzeugen eines aufschlussreichen Innenlebens

Das Dezernat Technik hat mir Gebäudepläne für den Gutenbergbau zur Verfügung gestellt.

4.5 Performance

Collider vs Distanzüberprüfen in Update
    Fremde Gebäudemodelle

5 Ausblick auf mögliche Anwendung und Erweiterung

ljkhg

6 Fazit

adf

7 Anhang

7.1 Glossar

Rigidbody Joint Collider Oculus Quest Unity subclass?

8 Notizen

Ich habe ein Dictionary statt eines Hashtables verwendet, weil ich nicht gemischte Typen verwenden möchte. https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=net-5.0

Typecasting ist vermutlich verwendet, wenn ich ein GreifbaresEinzelteil spezifisch und nicht wie ein OVRGrabbable behandeln möchte. https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/types/casting-and-type-conversions damit konnte eine leere Funktion gespart werden:

```
// diese Methode erscheint notwendig um die der subklasse Greifbares Einzelteil effektiv nutzen
// zu können. Da die Baseclass sich nicht mit anderen Stücken verbindet gibt sie eine leere Menge
// zurück. TODO: etwas eleganteres finden
public virtual HashSet<OVRGrabbable> alleVerbundenen(HashSet<OVRGrabbable> schonGefundene) {
  return schonGefundene;
}
```

stattdessen in dem Greifer:

```
if (m_grabbedObj is GreifbaresEinzelteil) {
  GreifbaresEinzelteil gegriffenesEinzelteil = (GreifbaresEinzelteil)m_grabbedObj;
  foreach (var k in gegriffenesEinzelteil.alleVerbundenen(new HashSet<OVRGrabbable>())) {
    k.grabbedRigidbody.MovePosition(grabbablePosition);
    k.grabbedRigidbody.MoveRotation(grabbableRotation);
  }
}
```

Tuples statt Array, weil verschiedene Typen.