VR-Tempel

Ein Virtual Reality First Person Puzzle Spiel

Tobias Watzl, Jesko Plitt, Steffen Grill
Hochschule der Medien
jp062@hdm-stuttgart.de

Abstract

Das Ziel unseres Projektes "VR Tempel" war das Umsetzen eines kleinen Spieles in Virtual Reality (VR) mit Hilfe der Unity-Engine. Es handelt sich dabei um ein Geschicklichkeits/Puzzle - Spiel. Der Spieler muss zum Lösen eines Levels einen Abgrund überwinden und auf der anderen Seite einen Schalter betätigen. Hierfür muss er verschiedenen Fallen ausweichen. Für jedes Level gibt es ein Zeitlimit. Der Spieler sammelt dabei Punkte nach jedem abgeschlossenen Level, je nach Schwierigkeitsgrad des Levels und übriger Zeit nach dem Beenden des Levels. Nach dem Beenden des Levels fährt der Spieler mit einem Aufzug in den nächsten Abschnitt. Verschiedene Spieler können ihre Leistungen in einer Highscore vergleichen.

Einleitung

Virtual Reality ist momentan in aller Munde. Die Anwendungsmöglichkeiten der Technologie erstrecken sich über die verschiedensten Arbeitsgebiete und beschränken sich nicht nur auf Spiele. So bieten sich viele Möglichkeiten wie z.B. in der Fertigung oder Logistik. Aus den genannten Gründen ist es wahrscheinlich, dass VR auch in der Zukunft ein spannendes und aktuelles Thema bleibt.
Mit diesem Projekt erhofften wir uns erste Einblicke in die Welt der VR zu erlangen und gleichzeitig zu schauen, was man in einem relativ kleinen Team von nur drei Leuten innerhalb von einem halben Jahr umsetzen kann.

Herausforderungen

1. Erstellen einer 3D-Umgebung.
2. Integration der HTC-Vive in die Unity-Engine.
3. Integration der Leap Motion in die Unity-Engine zum Tracken der Hände.
4. Erstellen eines fordernden Gameplays, welchen den Spieler in gleichem Maße fordert und Spaß macht.
5. Eine Spielmechanik einbauen, welche den Spieler zum erneuten Spielen motiviert.
6. Verwenden der HTC-Vive Controller zum Tracken der Füße.
7. Zufälliges Generieren von Level mit steigender Komplexität.
8. Befestigung der Controller an den Füßen, so dass diese immer den gleichen Abstand vom Boden haben wenn der Spieler still steht.

Verwendete Hard- und Software

Als VR-Headset verwenden wir eine HTC-Vive. Für die Vive haben wir uns vor allem wegen der Möglichkeit zum Tracken des Headsets im Raum entschieden. So war es uns möglich, den Spieler real quer durch den Raum laufen zu lassen, diese Bewegung zu tracken und auf die Spielfigur zu übertragen.
Bei der Engine haben wir uns für Unity entschieden. Das liegt zum einen an der schon guten Integration der VR-Headsets, sowie unseren Vorkenntnissen mit dieser Engine. Des Weiteren bietet Unity viele freie Bibliotheken sowie eine große Anzahl an frei verfügbaren Tutorials an.
Zum Tracken der Hände verwenden wir die Leap Motion, da diese momentan der mit Abstand am weitesten verbreitete Sensor ist, welcher unsere Anforderungen erfüllt.
Des Weiteren verwenden wir die externe Bibliothek Virtual Reality Toolkit (VRTK). Die Bibliothek bietet viele Hilfreiche Funktionen zum Einbinden der HTC Vive in Unity.

Code-Beispiele

In folgendem Code-Beispiel erkennen wir, ob ein Spieler momentan eigentlich in den Abgrund stürzen müsste. Hierzu wird ein Raycast vom Controller, welcher am Fuß des Spielers angebracht ist, in Richtung des Bodens abgesendet. Falls der Raycast innerhalb von einem definierten Abstand auf kein Objekt trifft, wird die Fallen-Animation abgespielt.
Davor prüfen wir noch, ob der Spieler momentan seinen Fuß anhebt um über einen Abgrund zu steigen. Ist dies der Fall, wird keine Fallen-Animation abgespielt, auch wenn der Raycast auf kein anderes Objekt trifft.
Diese Abfrage wird bei jedem FixedUpdate ausgeführt.

Verwendete Variablen:
dwd = Ein Vector, welcher in Richtung des Bodens zeigt.
leftControllerPosition = Die lokale Position des linken Controllers
rightControllerPosition = Die lokale Position des rechten Controllers
rcfd = Die Höhe des Controllers zu Beginn des Spiels. Dient als Referenzwert.

```
private void FixedUpdate(){
        Debug.DrawRay(leftControllerPosition, dwd, Color.yellow);
        Debug.DrawRay(rightControllerPosition, dwd, Color.green);
        if (controllerOnFloor(leftController, lcfd)){
                if (!Physics.Raycast(leftControllerPosition,dwd,1)){
                        isFalling = true;
                }
        }else{
                isFalling = false;
        }
        if (controllerOnFloor(rightController, rcfd)){
                if (!Physics.Raycast(rightControllerPosition,dwd,1)){
                        isFalling = true;
                }
        }else{
                isFalling = false;
        }
}
```

Ergebnis

Insgesamt sind wir mit dem Ergebnis sehr zufrieden.
Trotzdem müssen wir uns eingestehen, dass ein halbes Jahr nicht ausgereicht hat, um alle am Anfang von uns gestellten Anforderungen zu erfüllen. So ist der Rätsel-Teil deutlich kürzer ausgefallen als ursprünglich geplant.

Figure 1: Beispiel für ein generiertes Level

Positiv für uns war, dass wir bei unserem Projekt keinerlei Probleme mit Motion Sickness hatten. Was wir aber festgestellt haben ist, dass Fallen als Animation sehr schlecht umsetzbar sind, da der Spieler nicht in der Realität fallen kann. Lässt man den Spieler nun in dem Spiel wo herunterfallen, ist dies ein sehr unangenehmes Gefühl, da das Gehirn eine andere Bewegung von den Augen mitgeteilt bekommt als der Körper eigentlich erfährt. Außerdem ist es äußerst unangenehm, wenn an der Hand des Spielers etwas im Spiel befestigt ist, welches er in Echt nicht an seiner Hand befestigt hat.
Generell ist es unangenehm für den Spieler, wenn es zu einer Diskrepanz beim virtuellen Körper und dem realen Körper des Spielers kommt.
Viele Probleme hat uns zu Beginn das Herunterfallen beim Treten neben eine Plattform gemacht. Dieses Problem konnten wir jedoch mit Hilfe eines Raycasts wie im Code-Beispiel lösen.

Figure 2: Der Arm und Hand des Spielers werden getrackt

Positiv zu vermerken ist noch, dass wir relativ hochauflösende Texturen und detaillierte Objekte verwenden konnten, ohne dass das Spiel anfing zu ruckeln oder Unity überfordert war.
Auch positiv überrascht waren wir vom Höhengefühl, welches dem Spieler in unserem Spiel von der 3D Umgebung in Verbindung mit VR vermittelt wird.
Einen hohen Beitrag zur Immersion leistet das Tracking im Raum, welches einwandfrei funktioniert und sich kaum Aussetzer leistet.

Zusammenfassung

Abschließend lässt sich feststellen, dass es überraschend schnell und mit einfachen Mitteln möglich ist, eine funktionierende 3D-Anwendung zu erstellen, durch welche man sich in VR bewegen kann. Hierzu sind auch keine großen Vorkenntnisse notwendig.
Inzwischen bieten die Hersteller der VR-Brillen sowie die Engine-Entwickler einiges an Bibliotheken und Tutorials an, welche den Einstieg sehr erleichtern.
In den Hilfestellungen der Hersteller findet man zu herkömmlichen Fragestellungen meist eine Antwort, wenn man allerdings ein sehr spezifisches und komplexes Problem hat, ist man momentan noch ziemlich auf sich alleine gestellt. Es existiert noch keine große Community, welche einem bei spezielleren Problemen weiterhelfen könnte. Man ist also auf die Hilfestellungen der Hersteller angewiesen.

Zukunft des Projekts

Leider ist es keinem von uns möglich an dem Projekt weiterzuarbeiten nach diesem Semester. Den Code sowie die Assets stellen wir bei Interesse jedoch gerne zur Verfügung.

Referenzen

Unity
https://unity3d.com/
Unity Scripting Reference
https://docs.unity3d.com/ScriptReference/
VRTK
https://vrtoolkit.readme.io/
HTC Vive
https://www.vive.com/
LEAP Motion
https://www.leapmotion.com/