VR-Game Dokumentation

Aufgabenstellung

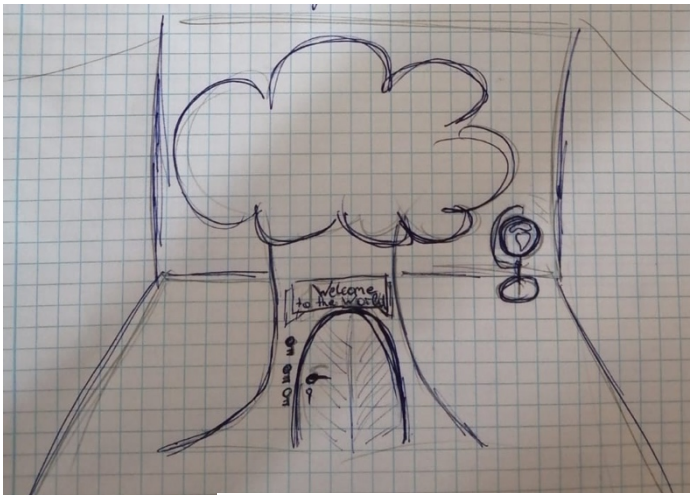
Die Aufgabenstellung war ein VR-Game zu erstellen, mit simplen Interaktionen. Grundsätzlich waren uns keine Grenzen gesetzt, die einzigen Vorgaben waren, dass das Game einen rotierenden Globus enthalten soll und dass es ein Rätsel zu lösen gibt.

Ideenfindung

Um auf eine erste Idee bzw. einen Anhaltspunkt für ein Game zu kommen haben wir uns 2 Schlagwörter ausgesucht, welche im Game umzusetzen waren. **Power und Tree.**

Zuerst hatte ich die Idee, dass sich der Globus in der Mitte des Raumes befindet. Wenn man ihn dreht wird man zu dem Ort teleportiert, an welchen die Rotation stoppt. Man wäre zu einem Apfelbaum gekommen und man hätte mit den am Boden herum liegenden Äpfel ein Ziel treffen müssen, um wieder zum Ausgangspunkt, dem Globus zu kommen.
Doch diese Idee verwarf ich beim genaueren überdenken schnell, da die Aufgabenstellung mit dem Rätsel nicht wirklich erfüllt worden wäre und mir die Umsetzung generell etwas schwierig erschien. Somit überdachte ich meine Idee noch einmal und entschied mich für eine Simplifikation.

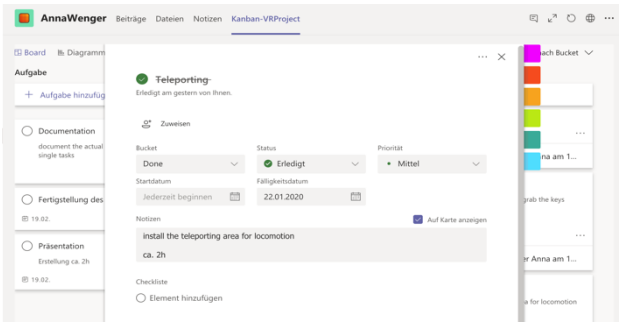
In der Mitte des Raumes soll sich ein großer Baum befinden mit einer großen Tür, welche gerade dazu einlädt, dass man sie durchschreitet. Zusätzlich fordert einen das Schild über der Tür „Welcome to the World“ dazu auf, den Raum durch die Tür zu verlassen. Doch die Tür ist verschlossen. Auch die drei Schlüssel neben der Tür sind wenig Hilfe, mit keinem lässt sich die Tür aufsperren. Das Rätsel, das es zu lösen gibt, ist das einen nur der im Eck platzierte Globus die Tür zur großen weiten Welt öffnen kann. Erst durch das Rotieren des Globus öffnet sich die Tür und man hat das Game gemeistert.



Target Plattform

Als Target Plattform verwendete ich die Oculus Rift. Um grundsätzlich mal zu wissen, ob es für die Oculus Rift irgendwelche Besonderheiten zu beachten gibt, recherchierte ich zuerst über die Hardware und beschäftigte mich mit den Spezifikationen. Doch ich kam auf keine Besonderheiten und somit entschied ich mich für eine standardmäßige Verwendung der Controlls. Man kann sich durch die Kopfbewegung umsehen, im Raum bewegt man sich mittels teleporting fort und für jegliche Interaktion mit Objekten, reicht es einen der a;b;x;y- Buttons zu drücken.

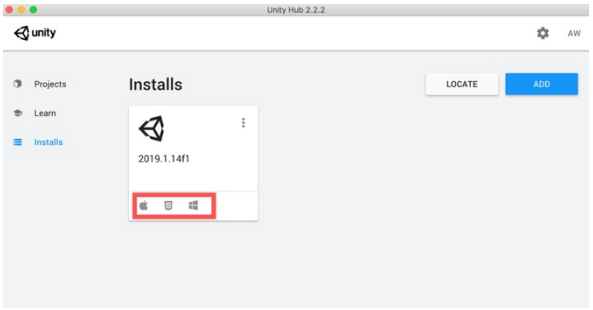
Organisation



Für die Umsetzung überlegte ich mir zuerst was ich alles für das Projekt benötige, bzw. in welche Tasks ich es aufteilen kann. Anschließend erstellet ich ein Kanbanboard und erstellte Aufgaben für die einzelnen Tasks und wies denen ein Fälligkeitsdatum zu um meinen Zeitplan nicht aus den Augen zu verlieren. Zusätzlich schrieb ich mir kurze Notizen was bei dem jeweiligen Task zu machen ist.

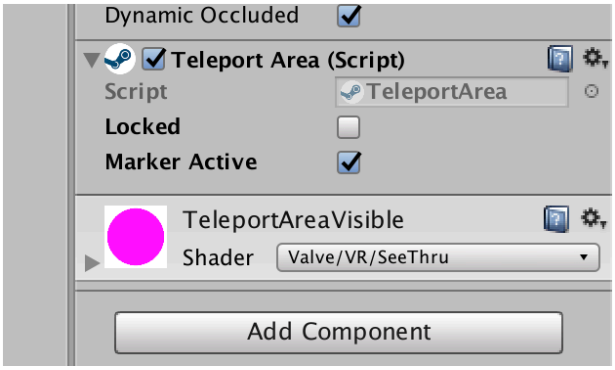
Umsetzung

Mein Unity-Project begann ich zuerst nur mit Whiteboxes und da mein Game generell sehr simpel aufgebaut ist, benötigte ich auch nur ca. 7 Whiteboxes und die Scene war immer noch sehr übersichtlich.

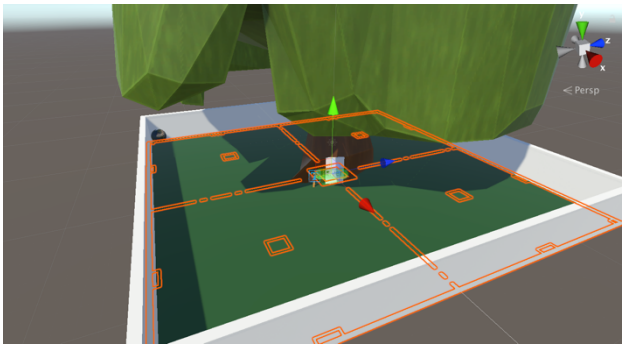


Das erste Problem erwies sich beim Installieren des VR-Plugins, denn um das Game in späterer Folge richtig zu Builden können benötigt man zusätzlich auch das Build für Windows, welches ich nicht hatte. Da ich mein Unity local am Laptop installiert hatte und nicht über Unity Hub, konnte man das nicht einfach umstellen bzw. hinzufügen. Deshalb musste ich Unity erstmal erneut über Unity Hub downloaden, was bei unserem Schul-WLAN zu einem erheblichen Zeitverlust führte.

Nachdem endlich alles notwendige installiert war, testete ich die Sample-Scene um generell mal das VR-Setup zu testen. Dies funktionierte soweit.
In weiterer Folge importierte ich die Teleport-Area in mein Projekt. Da ich grundsätzlich nicht sehr vertraut mit Unity und VR bin, recherchierte ich die Essentials für das Teleportieren in VR. Es wurden zwei verschiedene Varianten vorgestellt. Zum einen die Teleporting-Points und zum anderen die Teleporting-Area. (https://unity3d.college/2017/05/16/steamvr-locomotion-teleportation-movement/)

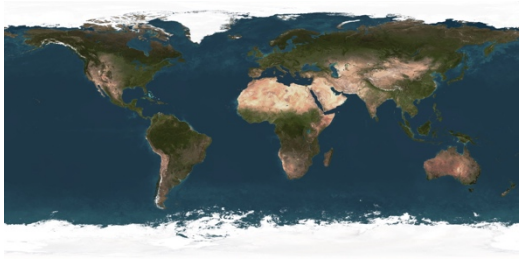


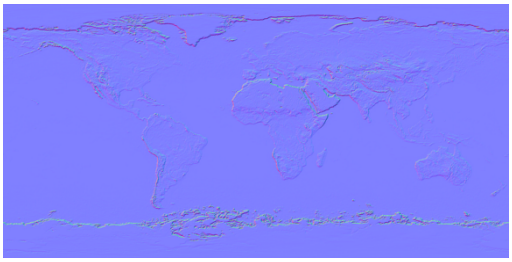
Da ich mich grob an die Vorgehensweise mit der Teleport-Area von einem anderen Projekt erinnern konnte entschied ich mich für diese Variante. Doch spätestens als die Teleporting-Area nicht wie üblicherweise in einem blauen Grid angezeigt wird sondern eine rosa Plane war, war ich mir meiner Kenntnisse nicht mehr sehr sicher. Zuerst vermutete ich nur einen Fehler beim Importieren und versuchte es erneut, doch dies führte zu keiner Lösung. Die Plane war immer noch nicht wie erwartet. Auch sämtliche Einstellungen führten zu keiner Änderung und auch Google erwies sich als äußerst unnützlich. Das Einzige was ich zu diesem Problem finden konnte, war ein Issue auf gitHub in 2017, in dem der User über dasselbe Problem klagte. Doch Lösung fand ich keine, das Issue wurde geclosed da sich dieses Problem mit einem Softwareupdate löste. Da ich jedoch sowieso eine neuere Version verwendete war dieses Softwareupdate keine Hilfe.

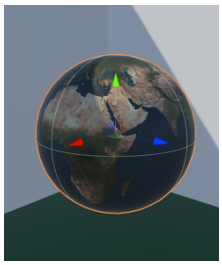


Erst nachdem ich das gesamte SteamVR Plugin neu importierte und die Teleporting-Area neu implementierte kam ich zu meinem gewünschten Ergebnis.
Um im zu testen ob das teleportieren nun endlich funktioniert und wie das Größenverhältnisse meiner zukünftigen 3D Models passt, hab ich mein Projekt auf den Schulrechner gecloned. Soweit funktionierte alles. Doch ich dürfte ausversehen etwas geändert haben und die Änderung gepusht haben. Da ich jedoch bevor ich auf meinem eigenen Rechner weitergearbeitet habe, das Projekt nicht gepullt habe kam es in weitere Folge zu einem Konflikt zwischen den zwei Verschiedenen Versionen und ich konnte weder pushen noch pullen. Da ich eben die Änderungen aus Versehen gepushed habe verging sehr viel Zeit bis ich auf den Grund für dieses Problem gekommen bin. Um dieses Problem letztendlich zu beheben musste ich die zwei Versionen mergen.

Zwischendurch begann ich damit meine Whiteboxes mit 3D Models zu ersetzen. Da ich nicht zu viel Zeit damit verschwenden wollte, beschloss ich bereits fertige Models aus dem Internet zu verwenden und entdeckte google poly. Die Seite von Google ermöglicht es jedem seine 3D Models upzuloaden und anderen diese für non-kommerzielle Zwecke kostenlos zu verwenden. Doch es ist nicht so einfach genau jene 3D Models zu finden, wie man sie sich vorgestellt hat. Deshalb blieb für die Tür vorerst die Whitebox. Für den Globus beglückte ich mit einer einfachen Sphere, und einer world map. Um dieser mehr tiefe zu geben erstellte ich zusätzlich eine Normalmap.



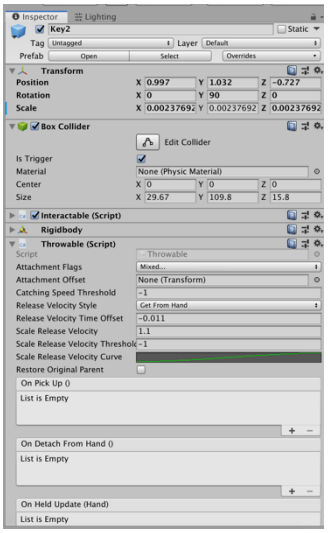




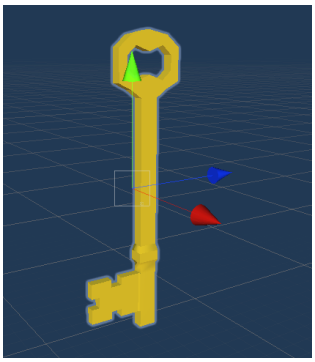
In weiterer Folge beschäftigte ich mich damit den Globus zum Rotieren zu bringen.
Um zahlreiche erfolgslose Suchen für den geeigneten Code zu vermeiden, sah ich mir die Rotation-Scripts meiner Mitschüler an und entschied mich aufgrund der Einfachheit für das von Valerian.

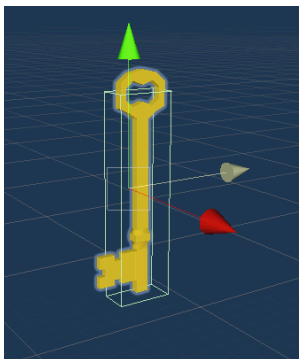
```
using UnityEngine;

public class Spinning : MonoBehaviour
{
    [Header("Settings")]
    [SerializeField, Range(-10f, 10f)] float rotationSpeed;

    void Update()
    {
        transform.Rotate(0, rotationSpeed, 0, Space.World);
    }
}

```

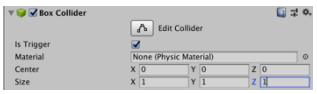
Damit man die Keys nehmen und damit interagieren kann, verwendete ich das Throwable-Script von Unity.

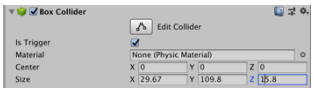


Um mein Game in VR auszuprobieren pullte ich mein Projekt erneut auf den Schulrechner und musste feststellen, dass das Projekt komplett anders aussieht und definitiv nicht auf dem neuesten Stand war. Nach langen herum probieren vergleichen, erneuten clonen, etc. musste ich feststellen, dass ich anscheinend vergessen hatte mein Unityprojekt zu speichern bevor ich es pushe. Dies bedeutete, dass die gesamten Replacements der 3D Models, sowohl auch die Anpassungen und das Throwable-Script umsonst waren und ich alles erneut machen musste.









Anschließend als dieses Problem behoben war, konnte ich nun mein Game testen und stellte fest, dass sich der Globus nicht dreht und man die Keys nicht greifen konnte.
Doch diese Probleme schienen schnell behoben. Für die Rotation hatte ich übersehen eine Rotationgeschwindigkeit hinzuzufügen und bei den Keys war das Problem, dass der Boxcollider viel zu klein war, um die Keys greifen zu können. Da ich diese Änderung zuerst am Schulrechner vorgenommen hatte, um sofort zu überprüfen ob nun alles funktionierte, führte dies erneut zu einem Versionenkonflikt. Leider ist mir bis heute noch ein Rätsel wie ich das jedes Mal aufs Neue schaffe.

Um meinen Globus nicht nur einfach zu rotieren lassen, sondern dass man ihn zusätzlich noch schneller drehen kann, suchte ich im Internet nach einem Code der mir diese Interaktion ermöglichte. Jenen Code, welchen ich fand und mir sehr passend erschien, versuchte ich auf meine Anforderungen anzupassen, doch die Interaktion funktionierte in VR nicht. Da mir die Zeit fehlte, um mich länger damit zu beschäftigen und den Code zum Laufen zu bringen, beschloss ich das Script nicht zu verwenden.

```
using UnityEngine;

public class Rotate : MonoBehaviour
{
    float rotSpeed = 20;

    void OnMouseDrag()
    {
        float rotY = Input.GetAxis ("Left Hand")*rotSpeed*Mathf.Deg2Rad;

        transform.RotateAround(Vector3.right, rotY);
    }

}
```

Da ich ein spielbares Projekt jetzt am Ende aufgegeben hatte, beschloss ich noch einzelnen Feinheiten in meiner Scene anzupassen, wie etwa das „Welcome to the World“ Schild, welches noch fehlte oder das Türschloss.

Fazit

Aus diesem Projekt lässt sich schließen, dass es für die Gameentwicklung nicht nur auf gute Programmierkenntnisse ankommt, sondern dass ebenso eine gute Organisation, Zeitmanagement und Konzentration, vor allem beim pushen und pullen, wichtig sind. Für mich lässt sich sagen, es waren viele kleine Unachtsamkeiten, die mein Projekt scheitern ließen, da diese zu fatalen Problemen führten und mich die meiste Zeit gekostet hatten. Doch ich habe auch gelernt, dass viele Probleme nicht so schlimm sind wie sie im ersten Moment erscheinen und das es wichtig ist rational an die Problembehebung anzugehen und nicht sofort panisch werden, da somit nur viel mehr Fehler entstehen.