

Anleitung zum Erstellen von Virtual Reality Apps in Unity

basierend auf Unity Version 2021.2.0f1
getestet mit Oculus Rift S und Oculus Quest 2

18. FEBRUAR 2022

Hochschule Fulda
Verfasst von: Stefan Friesen



Hochschule Fulda

Inhaltsverzeichnis

| | |
|--|-----------|
| Einleitung | 3 |
| Installation | 3 |
| Setup Unity..... | 5 |
| PC VR – das Headset | 5 |
| PC VR – die Controller | 6 |
| Android VR (Quest / Quest 2) | 8 |
| Export | 12 |
| Setup Debugging | 13 |
| Setup Open Sound Control (optional) | 13 |
| Zusätzliche Ressourcen | 14 |

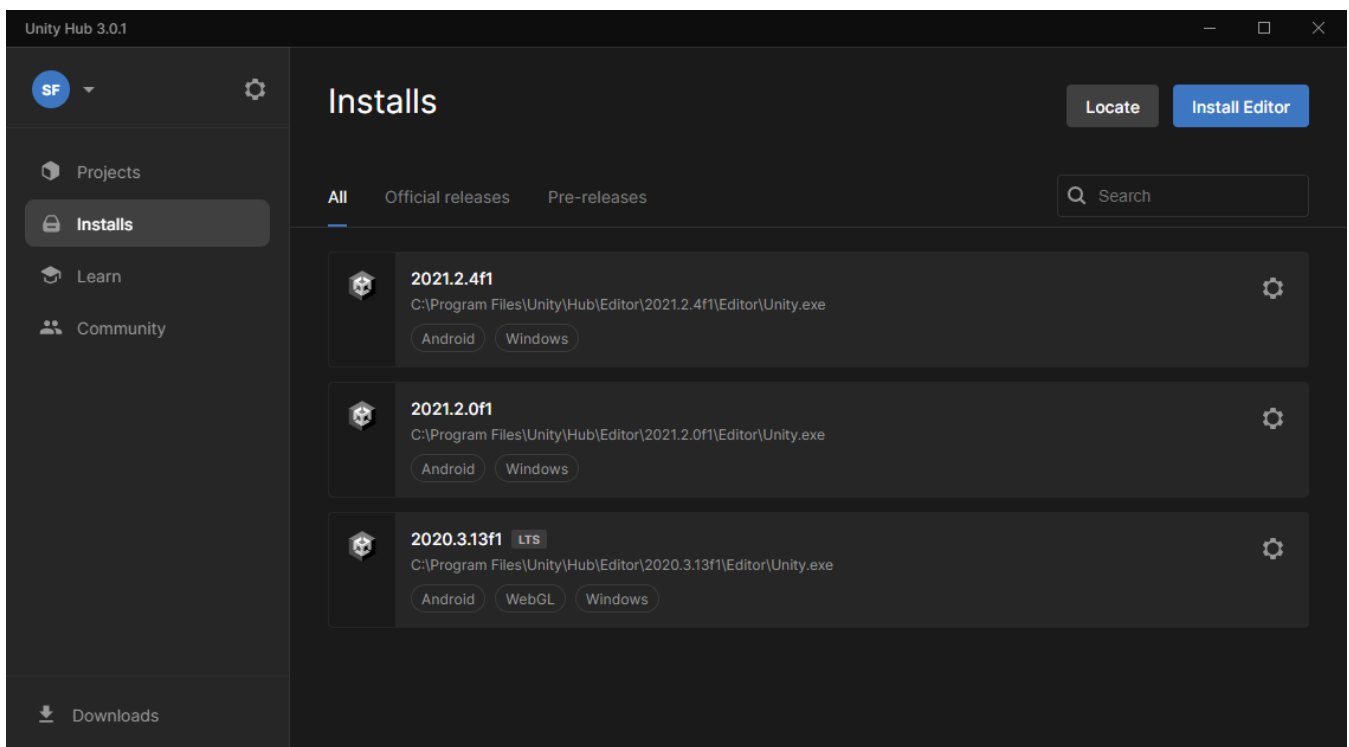
Einleitung

Diese Anleitung soll das Erstellen von Anwendungen in der virtuellen Realität mit Unity erleichtern und verschiedene Wege zur Fertigstellung einer VR-Anwendung aufzeigen. Da die Entwicklungslandschaft für VR-Brillen sich stetig ändert und immer wieder neue Modelle erscheinen kann es gut sein, dass einige hier in der Anleitung erwähnten Softwaretools durch neuere ersetzt werden und der Support von HTC, Valve, Meta (ehemalig Oculus bzw. Facebook) oder Microsoft eingestellt werden kann.

Der Einfachheit halber werden alle Installationsschritte mit Screenshots von Windows 10 begleitet, die meisten Installationsschritte sind unter MacOS ähnlich aufgebaut. Unity bietet die Möglichkeit der kostenlosen Lizenzen für private Zwecke an, welche ich für Projekte an der Hochschule nur empfehlen kann. Außerdem bietet auch die Hochschule auf Nachfrage Educationlizenzen an Computern vor Ort an.

Installation

Das Setup verwendet die 3D-Engine Unity, empfohlen ist immer die neueste Version aus dem Unity Hub zu installieren, da viele VR-Pakete und Tools oft in Preview sind und zusammen mit den neuesten Versionen aktualisiert werden.

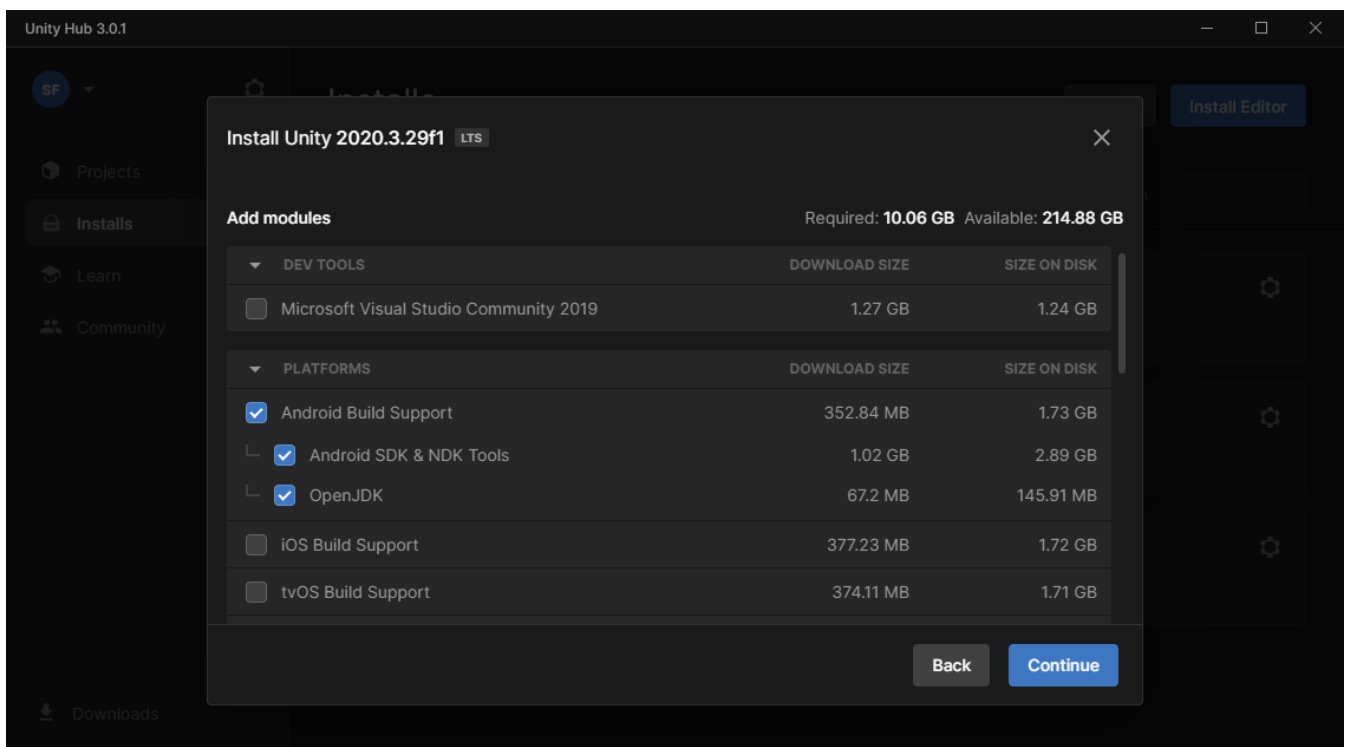


Das Unity Hub lässt sich unter <https://www.unity.com/download> herunterladen und mit der Unity Hub Setupdatei installieren.

Im linken Reiter unter „Installs“ werden alle vom Unity Hub installierten Versionen aufgelistet und es können oben rechts unter dem blauen „Install Editor“-Knopf neue Versionen hinzugefügt werden. Der empfohlene Release („Recommended“) ist der neueste technisch stabile LTS-Release, den auch ich hier zur Installation empfehle. Wichtig für die Entwicklung von Virtual Reality ist hier die Auswahl der korrekten Module, die zur Installation hinzugefügt werden sollen.

Ist eine spezielle Version notwendig, lassen sich diese im Unity Archiv unter <https://unity3d.com/get-unity/download/archive> herunterladen. Hier besteht auch die Möglichkeit für Mac-User zwischen Versionen für Intel und Apple Silicon Macs auszuwählen.

Für die Entwicklung in Standalone Headsets wie der Oculus Quest 2 ist der Android Build Support notwendig, für an den PC gebundene VR-Headsets wie der HTC Vive oder Oculus Rift S benötigt man Windows Build Support.



Nach dem Download und der Installation von Unity kann ein neues Projekt erstellt werden.

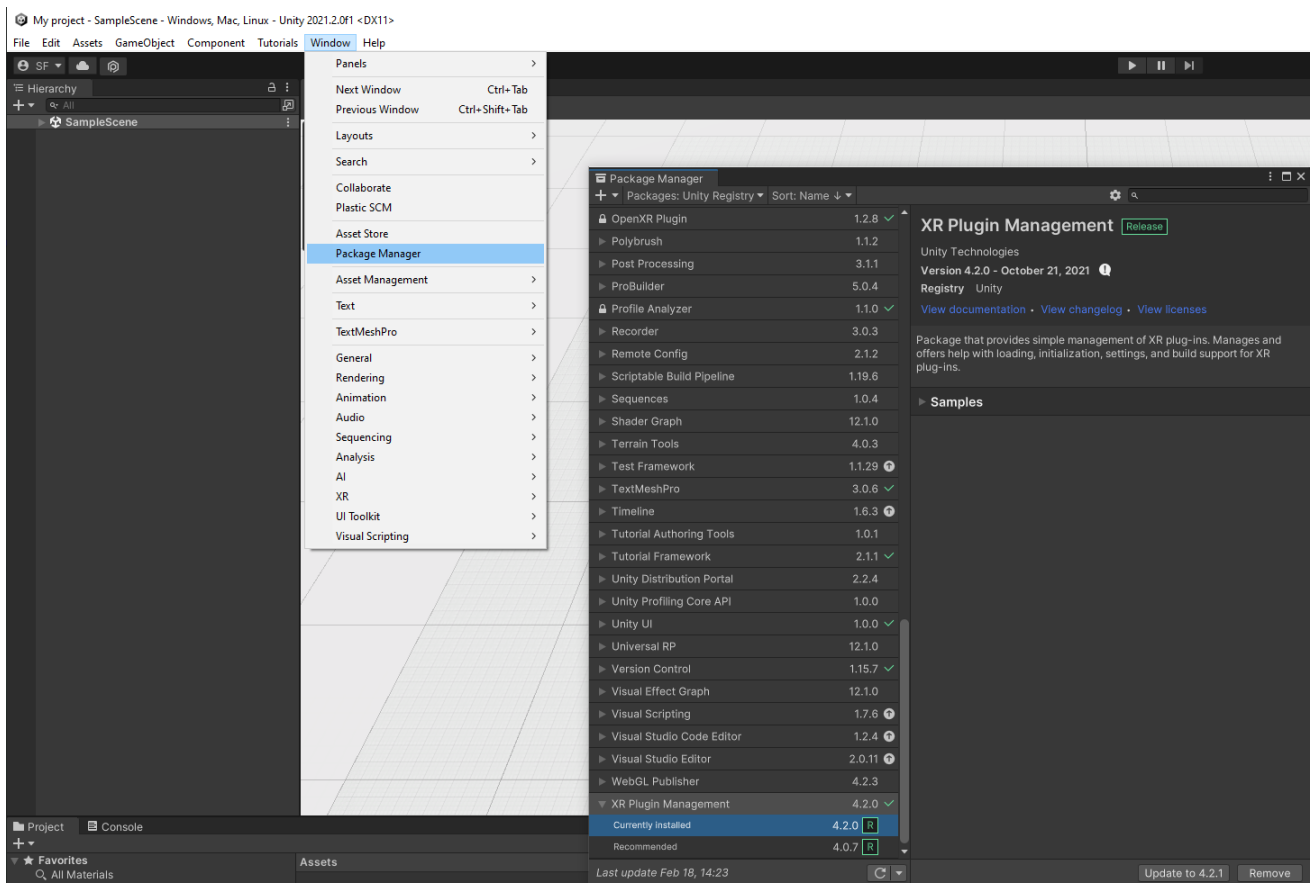
Setup Unity

Bei der Erstellung eines Unityprojektes wird zuerst ein vorgegebenes Template angepasst an die Entwicklungsumgebung empfohlen. In den meisten Fällen reicht der einfache 3D Core (eventuell mit der Universellen Render Pipeline URP), aber es wird auch ein VR Core zur Verfügung gestellt.

Der VR Core beinhaltet eine Beispielszene mit Controllermodellen, einem vorgefertigten XR Rig (für PC-Anwendungen) und Tutorials von Unity. In der weiteren Anleitung wird beschrieben, wie das Setup ohne den VR Core von Hand konfigurierbar ist, um ein Verständnis für die einzelnen Komponenten eines sogenannten XR Rigs zu entwickeln.




PC VR – das Headset

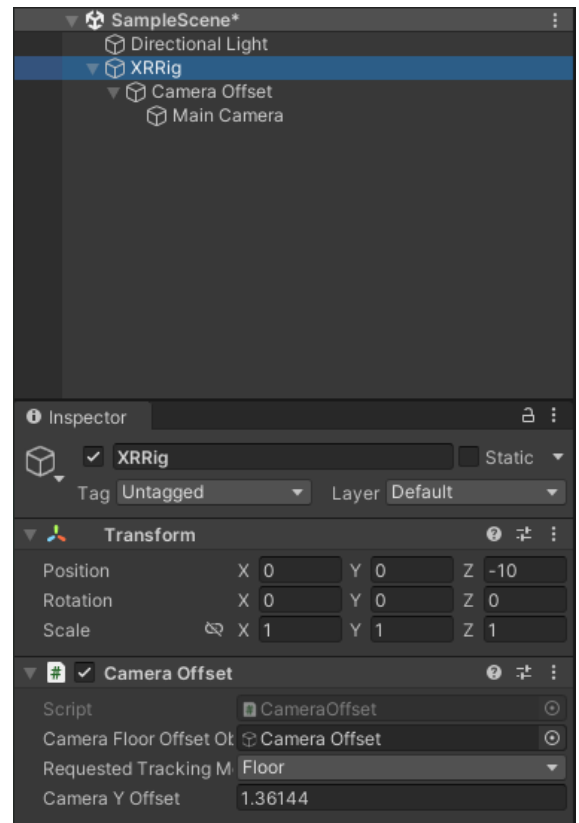
Anwendungen für an den PC gebundene VR-Headsets sind einfacher gestaltet als das Androidgegenstück für die Oculus Quest 2. Um von der standardmäßigen Kamera in einer leeren Unityszene zu einem VR Rig zu kommen, muss in der Regel nur unter **Window > Package Manager** das Paket „XR Plugin Management“ und die supportenden Oculus XR und OpenXR Plugins installiert werden, alle Pakete befinden sich in der Unity Registry.



In dem Untermenü **Edit > Project Settings > XR Plug-in Management** kann dann das jeweilige Plugin für das gewünschte VR-Headset installiert werden. Für die maximale Kompatibilität mit mehreren Headsets ist OpenXR am besten geeignet, gerätespezifische Features für Oculus VR Headsets oder Magic Leaps AR Headsets haben ihre eigenen Plugins. In OpenXR muss noch zusätzlich von mehreren Interaktionsprofilen ausgewählt werden, Unity weist bei der Auswahl auch noch einmal darauf hin.

Mit einem Rechtsklick auf die Main Camera in der Hierarchie (standardmäßig das Fenster auf der linken Seite im Unity Editor) lässt sich unter dem Punkt „XR“ die Main Camera in einen XR Rig umwandeln (**Convert Main Camera to XR Rig**).

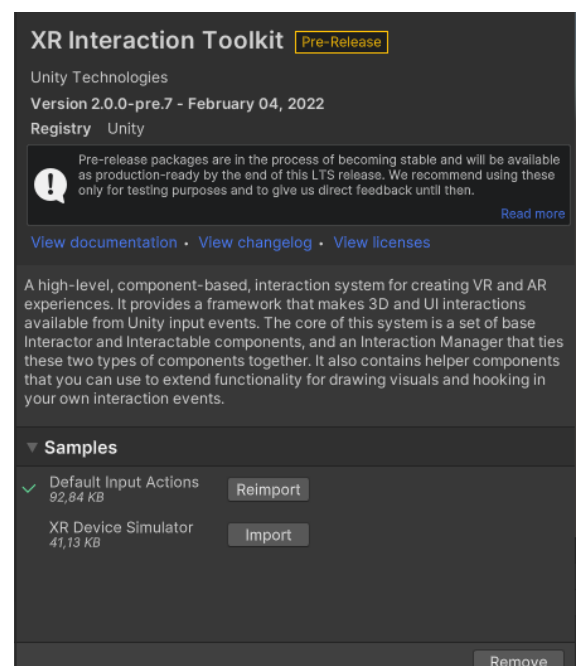
Der neu gesetzte XR Rig beinhaltet in der obersten Hierarchie ein Camera Offset Skript, hier muss die Variable „Requested Tracking Mode“ auf Floor gesetzt werden. Jetzt kann schon im Editortestlauf (  ) bei angeschlossener VR-Brille die Szene betrachtet werden.



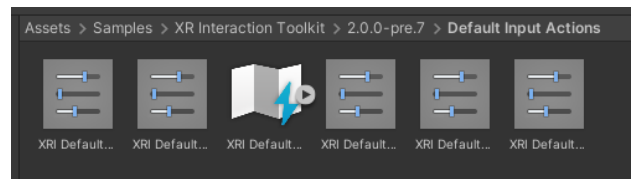
PC VR – die Controller

Um nun Controller mit in die Szene zu integrieren, müssen unter **Edit > Project Settings > Package Manager** sogenannte Pre-Release Packages erlaubt werden (**Enable Pre-Release Packages**). Dann kann im vorher geöffneten Package Manager das noch eventuell sich in Preview befindende Paket „**XR Interaction Toolkit**“ installiert werden.

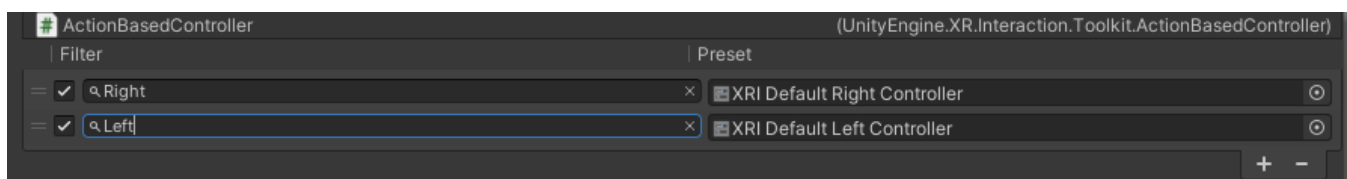
Hier sollten auch die zusätzlichen „Default Input Actions“ unter Samples mit importiert und die Version des Pakets geupdatet werden.




Die importierten Beispiele beinhalten im Projektverzeichnis (standardmäßig unteres Fenster im Unityeditor) unter **Assets > Samples > XR Interaction Toolkit > {Versionsnummer} > Default Input Actions** insgesamt fünf Presets (1x Move, 2x Controller und 2x Turn Varianten) und einen Importer. Die Presets lassen sich einsetzen, in dem man sie auswählt und im Inspector (standardmäßig rechte Seite) den jeweiligen Button „**Add to ActionBased[...] default**“ klickt.

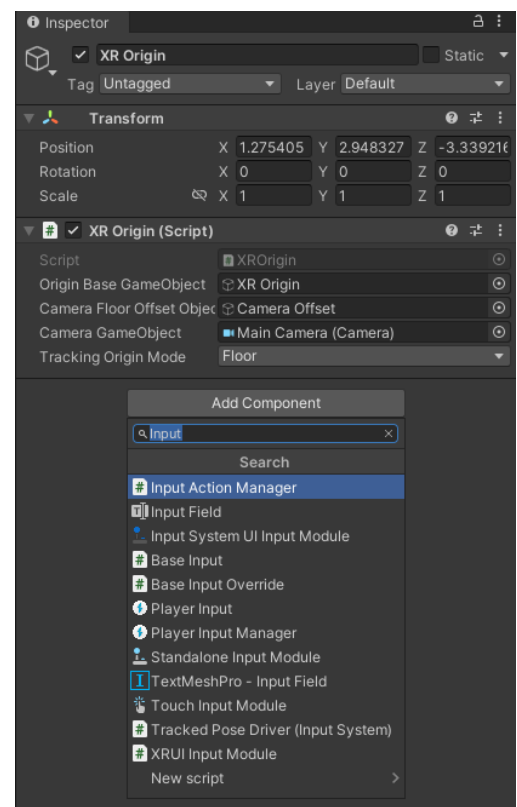


Diese Presets werden dann in **Edit > Project Settings > Preset Manager** geladen, dort müssen die Filter der Controller nur noch entsprechend „Left“ und „Right“ genannt werden, damit die Interaktionen korrekt zugeordnet werden können.



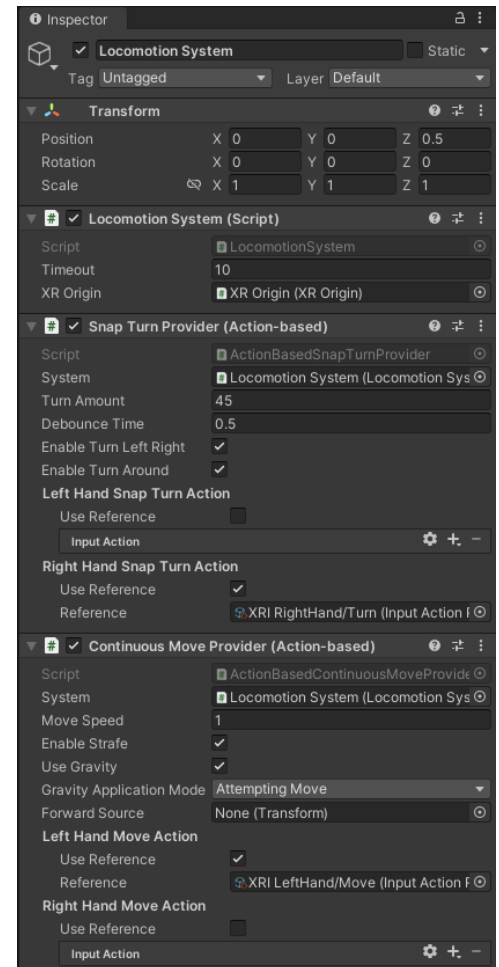
Jetzt kann der erste XR Rig, mit dem nur das Headset getestet wurde, aus der Hierarchie entfernt werden und mit der unter dem Punkt „XR“ neu verfügbaren **XR Origin (Action-based)** (in alten Toolkitversionen auch XR Rig (Action-based) genannt) ersetzt werden. Diese ähnelt sich im Aufbau des ersten XR Rigs und fügt unter dem Camera Offset zwei Controllerobjekte hinzu. Auch hier muss wieder die Variable „Tracking Origin Mode“ im Inspector auf Floor gesetzt werden, um den Interaktionsraum auf den Boden auszurichten. Die Controllerinputs müssen dem XR Origin Objekt als Komponente mit dem Skriptnamen „**Input Action Manager**“ hinzugefügt werden.

Wenn man jetzt den Editortestlauf () startet, kann man sowohl Headset als auch Controller in der Szene bedienen und es befinden sich auf der Position der Controller zwei steuerbare Strahlen, sogenannte **XR Ray Interactor**.



Um ein einfach greifbares Objekt zu erzeugen kann man ein Objekt in der Szene erstellen, mit der Komponente „**XR Grab Interactable**“ ausstatten und dann im Spiel mit dem XR Ray Interactor auf das Objekt zeigen und auf Knopfdruck festhalten.

Für Bewegung via Joysticks auf den Controllern muss in der Hierarchie noch das wieder unter dem Punkt „XR“ zu findende „**Locomotion System (Action-based)**“ hinzugefügt werden. Standardmäßig befinden sich drei Skripte in dem Objekt, das wichtigste ist hier das **Locomotion System** selbst, in das man das eben erstellte XR Origin Objekt von der Hierarchie in den Inspector zieht. Um Motion Sickness vorzubeugen ist für die Fortbewegung ein **Teleportation Provider** gegeben, der per Zeigen auf den Boden und Knopfdruck den Nutzer teleportiert. Die Komponente kann durch einen **Continuous Move Provider** für etwas erfahrenere VR-User ersetzt werden, hier wird sich wie in einem klassischen Videospiel durchgängig bewegt. Der **Snap Turn Provider** ist für Rotationen auf der Stelle in allen Bewegungsarten geeignet, falls die ruckartigen Snap-Rotationen nicht gewünscht sind, gibt es auch einen **Continuous Turn Provider**. Die Move und Turn Provider besitzen die Möglichkeit die Aktionen auf eine bestimmte Hand zuzuordnen, empfohlen ist hier die Rotation und Bewegung jeweils nur einem Controller zuzuweisen.



Jetzt kann man sich in der Szene mithilfe von einem VR-Headset sowohl bewegen, als auch mit Objekten interagieren.

Android VR (Oculus Quest / Quest 2)

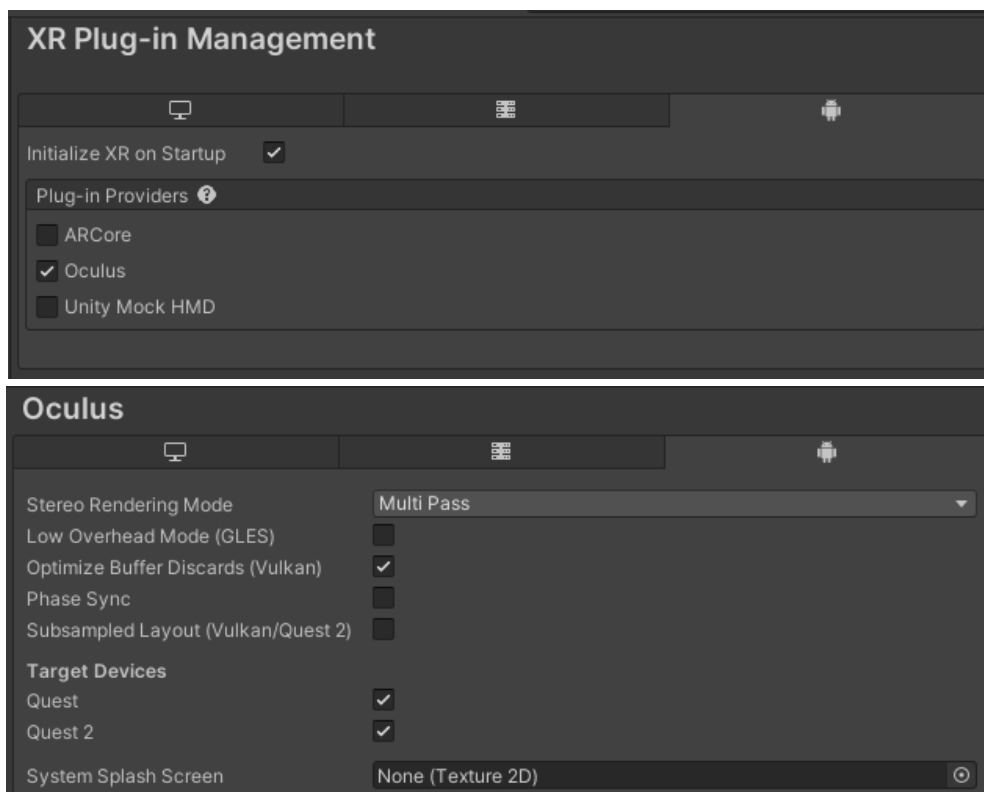
Die Erstellung von Unityanwendungen für die Gerätefamilie der Oculus Quest unterscheidet sich bereits bei der Installation, da hierfür der Android Build Support mit der Unityversion zusammen installiert und das Projekt selbst auf Android umgeschaltet werden muss.

Vorsicht: Es ist möglich, dass Unity bei der Installation von der Android SDK nicht alles vollständig installieren kann! Es muss unter **Edit > Preferences > External Tools** überprüft werden, ob die angegebenen Verzeichnisse von OpenJDK, Android SDK, NDK und Gradle auch Dateien enthalten und diese selbstständig installiert werden, wenn dies nicht der Fall ist.

Nach der Erstellung des Projektes müssen ähnlich zu den PC-Anwendungen unter **Window > Package Manager** folgende Pakete aus der Unity Registry installiert werden:

- XR Plugin Management
- Oculus XR Plugin
- XR Interaction Toolkit (evtl. Pre-Release, Installationsanleitung s. Seite 6)

Nach der Installation der Pakete muss unter **Edit > Project Settings > XR Plug-in Management** der Tab auf Android gestellt und in diesem Fall Oculus ausgewählt werden. Im Untermenü des Plug-in Managements namens Oculus kann im Tab für Android das Zielgerät (Quest / Quest 2) ausgewählt werden.



Der von Unity gestellte XR Rig, der vor allem seinen Einsatz in VR-Anwendungen für PC findet, ist hier nicht ausreichend ausgestattet und es wird das Importieren der **Oculus Integration** aus dem Asset Store empfohlen.

(<https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022>)

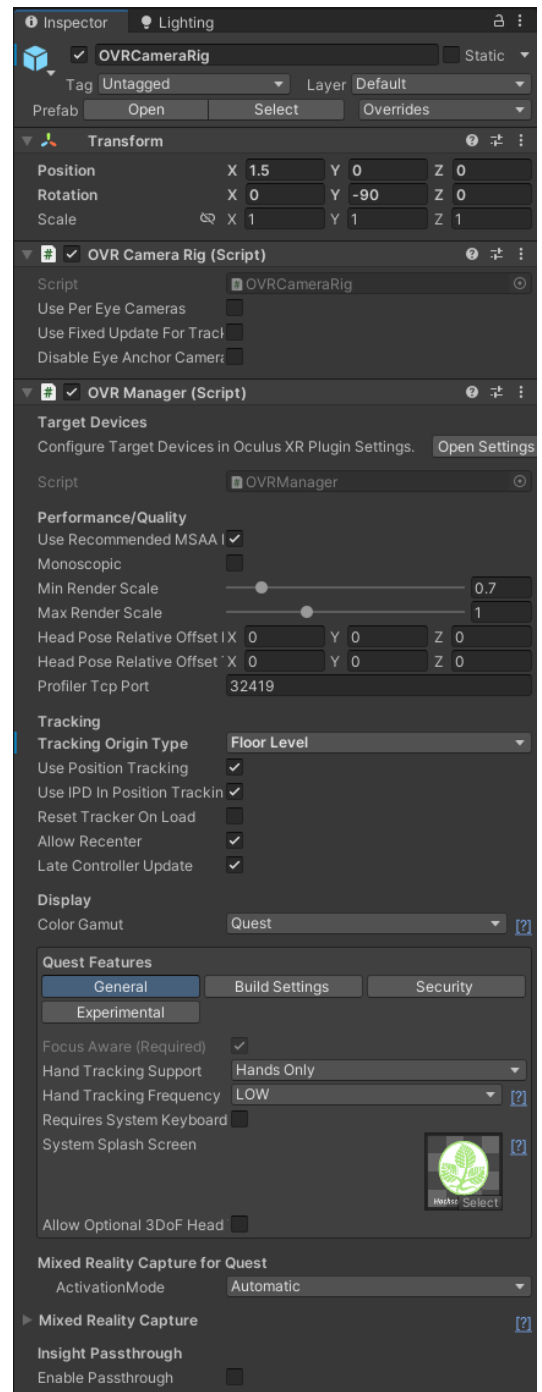
Bei der Installation wird gefragt, ob auf das **OpenXR Backend** und den **Oculus Spatializer** geupgradet werden soll. Beide Fenster sollten bestätigt werden, die entsprechenden Module sind in der Entwicklung bereits ausreichend fortgeschritten, um auf diese in einem Forschungssetting zurückzugreifen (Die Stabilität ist in diesem Fall nur für kommerzielle Produkte noch nicht ausreichend). Bei einer erfolgreichen Installation erscheint in der oberen Werkzeugleiste ein neues Untermenü mit dem Titel „Oculus“.

Die Oculus Integration bietet viele Komfortfunktionen zum Entwickeln in virtueller Realität, so kann man zum Beispiel Avatare (Handbewegungen, LipSync) und Audiolösungen für VR (Audio Manager, Spatializer) verwenden. Verschiedene Szenen zum Ausprobieren aller Funktionen befinden sich im Projektverzeichnis unter Assets > Oculus > VR > Scenes. Wichtige Komponenten sind hier die neuen vorgefertigten XR Rigs und der Export zu einer APK-Datei (s. Seite 12).

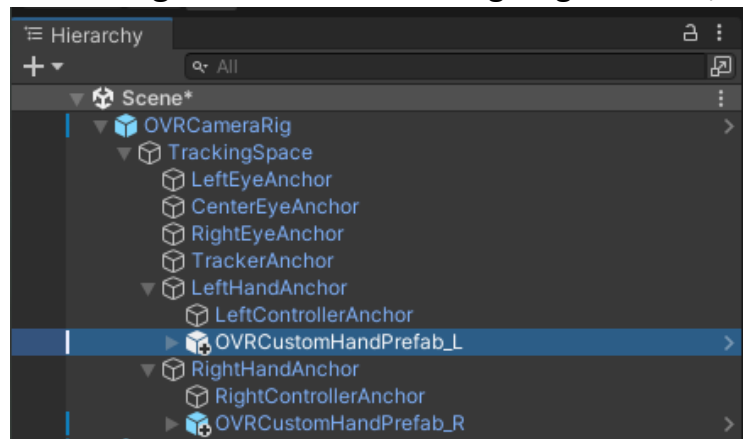
Der neue XR Rig für Oculus Quest Geräte befindet sich im Projektverzeichnis unter **Assets > Oculus > VR > Prefabs > OVRCameraRig.prefab**, in einer neuen leeren Szene kann man die Main Camera entfernen und das Prefab per Drag-and-Drop aus dem Verzeichnis in die Szene ziehen. Der **OVRCameraRig** besteht aus einem TrackingSpace mit sechs Ankern (linkes Auge, rechtes Auge, Headset (Center Eye), Tracker, linke Hand, rechte Hand) und einem OVRManager als Komponente.

Im **OVRManager** muss wieder die Variable „Tracking Origin Type“ auf Floor Level gesetzt werden. Unter dem Fenster „Quest Features“ lässt sich bei der Variable „Hand Tracking Support“ einstellen, ob zur Steuerung Controller oder Hände eingesetzt werden sollen oder ob dynamisch zwischen beiden gewechselt wird. In demselben Fenster lässt sich unter dem „Experimental“-Tab auch die Passthrough-Funktionalität einschalten, diese kann unter Umständen nicht funktionieren, da die Funktion noch experimentell ist.

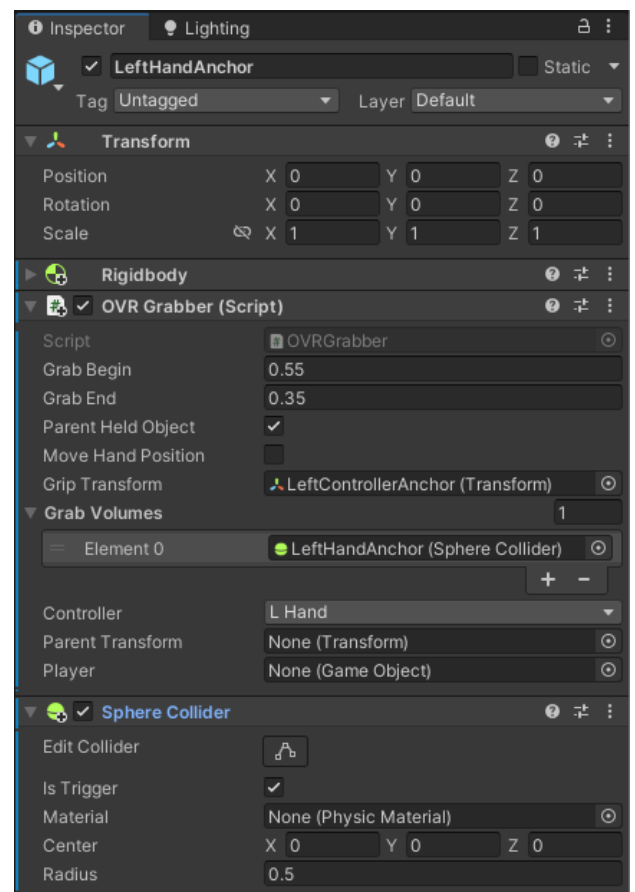
Für das Einsetzen von **Hand Tracking** ist neben dem Einschalten von dem Support im OVRManager auch das Hinzufügen von Handmodellen notwendig, um die gescannten Hände visuell in der Unityszene zu repräsentieren. Um Änderungen am OVRCameraRig vornehmen zu können, muss in der Hierarchie mit Rechtsklick auf das Objekt unter **Prefab > Unpack Completely** der Rig entpackt werden.



Das vorgefertigte Handmodell befindet sich im Verzeichnis unter **Assets > Oculus > VR > Prefabs > OVRCustomHandPrefab(_L / _R).prefab**. Diese können im TrackingSpace unter die beiden Objekte **LeftHandAnchor** und **RightHandAnchor** hinzugefügt werden, um Position und Rotation der jeweiligen Hände zu übernehmen. In dem OVRCustomHandPrefab muss dann unter dem Skript OVR Hand die korrekte Hand und unter OVR Custom Skeleton der korrekte Skeletttyp zugeordnet werden. Um alle Finger korrekt zu tracken muss auf den Button „**Auto Map Bones**“ geklickt werden.



Um Objekte in Unity mit Händen greifbar zu machen, muss an den beiden Objekten **LeftHandAnchor** und **RightHandAnchor** ein **OVRGrabber** Skript als Komponente hinzugefügt und ein Sphere Collider als Interaktionsraum zugeordnet werden. Die Checkbox mit dem Namen „Parent Held Object“ im OVRGrabber und die Checkbox „is Trigger“ im Sphere Collider müssen bestätigt werden. Das Kindobjekt **LeftControllerAnchor** bzw. **RightControllerAnchor** muss hier jeweils in das Feld „Grip Transform“ gezogen und der eben erstellte Sphere Collider in den Array „Grab Volume“ hinzugefügt werden. Im Dropdownmenü für die Controller wählt man noch die korrekte Hand aus und setzt diese Optionen genauso auf der anderen Hand um.



Objekte, die gegriffen werden sollen, bekommen eine Rigidbody-Komponente und das Skript **OVRGrabbable** zugeordnet.

Für Fortbewegung mithilfe eines OVRCameraRigs sollte dieser mit einem **OVRPlayerController** unter dem Verzeichnis **Assets > Oculus > VR > Prefabs > OVRPlayerController.prefab** ersetzt und dem Objekt eine Komponente mit dem Skript „Locomotion Controller“ hinzugefügt werden. Hier kann man dann Player Controller und Camera Rig miteinander kombinieren, um Controllerinputs als Bewegung zu

registrieren. Hier sind auch wieder mehrere Fortbewegungsmöglichkeiten gegeben, so zum Beispiel „Locomotion Teleport“ oder „Walk Only“.

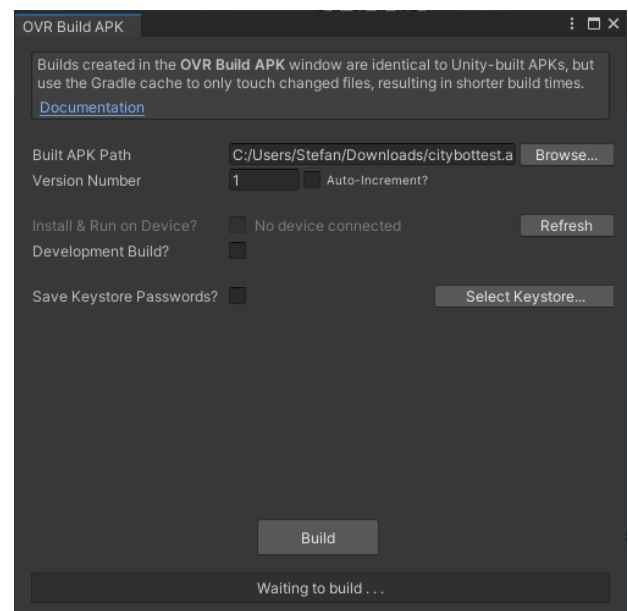
Export

Zum Exportieren der Anwendungen ist im Fall der PC-Anwendungen der Aufwand ähnlich zu traditionellen Videospielen, hier kann einfach direkt in die ausführbaren Dateien exportiert werden, ohne auf VR-spezifische Parameter achten zu müssen. Falls die Performance der Anwendung zu schwach ist, lassen sich unter **Edit > Project Settings > Quality** verschiedene grafische Einstellungen anpassen, in VR-Anwendungen ist erfahrungsgemäß das Anti Aliasing meist mit mehr Rechenintensität verbunden.


Für Exporte von Anwendungen, die die auf Android basierende Gerätefamilie von Oculus Quest verwenden, wird das OVR Build Tool empfohlen, das sich in Unity auf der oberen Werkzeugleiste unter **Oculus > OVR Build > OVR Build APK...** finden lässt.

Wichtige grafische Einstellungen für Oculus Quest Exporte sind folgende:

- **Edit > Project Settings > Player > Other Settings:**
 - Color Space: Linear
 - Texture compression format: ASTC
 - Minimum API Level: Android 6.0 ‚Marshmallow‘ (API Level 23)
 - Scripting Backend: IL2CPP
 - Target Architectures: ARM64



Setup Debugging

Für das Testen und Debuggen von VR-Anwendungen sind an den kabelgebundenen Headsets der Editortestlauf () von Unity bestens geeignet, hier hat man Zugriff auf die Unity Console und kann dort gemeldete Fehler beobachten und Variablen loggen. Unter dem Button „Stats“ lässt sich während des Tests die grafische Performance der Anwendung auslesen.

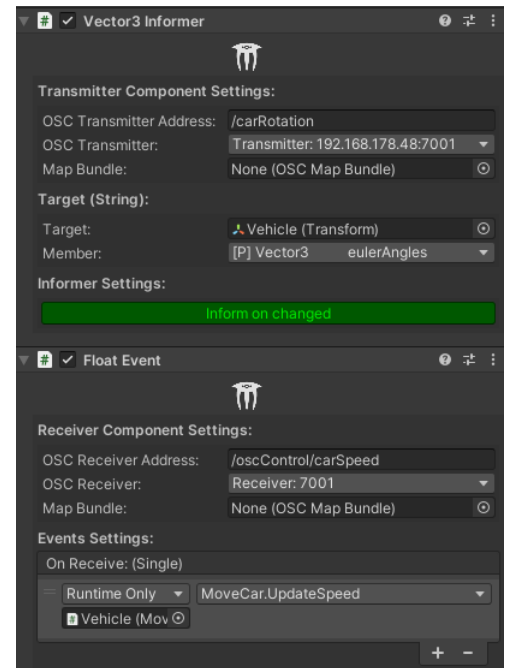
Bei der Entwicklung mit der Oculus Quest empfehle ich für Debugging das Oculus Developer Hub (<https://developer.oculus.com/documentation/unity/ts-odh/>), hier können am Computer per USB oder ADB over WiFi die exportierten APK-Dateien installiert und unter dem Performance Analyzer die Leistung des Headsets gemessen werden.

Setup Open Sound Control (optional)

Mithilfe von externen Tools aus dem Asset Store besteht die Möglichkeit in Unity über ein lokales Netzwerk das Protokoll Open Sound Control zur Fernsteuerung der Anwendung zu verwenden. Ursprünglich aus dem Audibereich kommend kann man per OSC Daten sehr effizient übertragen und empfangen.

In meiner Anwendung fand ich den Einsatz des Tools **extOSC** von Vladimir Sigalkin am einfachsten (<https://assetstore.unity.com/packages/tools/input-management/extosc-open-sound-control-72005>). Nach dem Importieren des Assets kann man in der Hierarchie per Rechtsklick unter **extOSC > OSC Manager** einen OSC Manager erstellen, der einen Receiver und einen Transmitter beinhaltet.

Hat man diese erstellt, dann kann man anderen Objekten in der Szene eine Komponente zuweisen, die entweder ein sogenanntes „Event“-Skript, bei der empfangene OSC-Daten Variablen in Unity verändern können, oder ein sogenanntes „Informer“-Skript ist, bei dem man bei sich ändernden Variablen in Unity ein OSC-Paket absenden kann.



Zusätzliche Ressourcen

- Oculus Developer Documentation:
<https://developer.oculus.com/documentation/unity/unity-overview/>
- YouTube Tutorials für VR-Entwicklung:
<https://www.youtube.com/c/ValemVR>
<https://www.youtube.com/c/JustinPBarnett>
- Unity Documentation:
<https://docs.unity.com/>