

Unity 3D Server for CAVE Rendering

Bachelorthesis

Studiengang:	Informatik
Autor:	Julien Villiger, Daniel Inversini
Betreuer:	Prof. Urs Künzler
Auftraggeber:	cpvr Lab BFH
Experten:	Harald Studer
Datum:	19.01.2016

Management Summary

Die Erstellung einer Applikation für den CAVE¹ der BFH ist ein relativ aufwändiger Prozess und setzt grosses Wissen über die Infrastruktur sowie fundierte C++ Kenntnisse voraus. Nur wenige Studenten realisieren, meist in Form einer Bachelor oder Master Thesis, eine Implementierung. Demzufolge ist das Zielpublikum klein und die Möglichkeiten des CAVEs werden kaum genutzt.

Im Rahmen der Thesis „Unity² 3D Server for CAVE Rendering“ wurde erfolgreich eine Integration von Unity in den CAVE vorgenommen. Durch die Einfachheit des Plugins, einer simplen Schritt-für-Schritt Anleitung und Demoapplikationen können auch unerfahrene Anwender von der Infrastruktur Gebrauch machen und die Stärke von Unity mit dem CAVE verbinden. Somit kann der schnell wachsenden Verbreitung von Unity Rechnung getragen werden und der CAVE der BFH ist über eine neue Plattform ansprechbar.

Der Hauptbestandteil der Umsetzung ist die virtuelle Abbildung der Komponenten. Mit Hilfe von Unity wird die Weiterverarbeitung und Interpretation vereinfacht und ist somit Basis für sämtliche Manipulationen der Applikation. Die Erweiterung der Projektion auf mehrere Leinwände erfolgt mittels Generierung virtueller Kameras, die sich einerseits der Logik der Applikation anpassen und andererseits Inputs vom Benutzer über die verschiedenen VR-Eingabegeräte des vorhandenen Trackingsystems von WorldViz³ entgegennehmen. So kann beispielsweise eine Kopfbewegung eine Änderung der Ansicht im Spiel bewirken. Gleichzeitig wird mit Hilfe dieser erstellten Kameras, Beamern mit Polfilter und polgefilterten Brillen eine stereoskopische⁴ Projektion ermöglicht.

Die gesamte Funktionalität lässt sich per Drag & Drop in ein neues oder bereits bestehendes Unity Projekt übernehmen. Einstellungen und der Funktionsumfang eines Spiels werden vom Plugin nicht überschrieben, lediglich erweitert. Um eine generische Lösung anbieten zu können, stehen etliche Einstellungsmöglichkeiten zur Verfügung. So können beispielsweise zusätzliche Kameras individuell platziert oder die Darstellung der GUI-Elemente auf eine CAVE-Leinwand fixiert werden. Die Viewfrustum⁵-Transformation, welche basierend auf der Position des Benutzers im CAVE berechnet wird, gewährleistet eine realistische Perspektive im virtuellen Raum und lässt den Benutzer in die künstliche Welt eintauchen.

Abgerundet wurde die Arbeit mit extra erstellten Demoapplikationen, welche die Möglichkeiten des CAVEs zusammen mit Unity, dem Plugin und dem Trackingsystem demonstrieren. 3D Welten lassen sich innert kurzer Zeit hautnah erleben, was auch für andere Abteilungen der BFH von grossem Nutzen sein kann.

¹ CAVE – https://de.wikipedia.org/wiki/Cave_Automatic_Virtual_Environment

² Unity – <http://unity3d.com/>

³ WorldViz – <http://www.worldviz.com/>

⁴ Stereoskopie – <https://de.wikipedia.org/wiki/Stereoskopie>

⁵ Frustum – https://en.wikipedia.org/wiki/Viewing_frustum

Inhalt

Management Summary	2
1 Einführung	5
1.1 Vorarbeiten Projekt 2	5
2 Infrastruktur	8
2.1 Trackingsystem WorldViz	9
2.2 Devices	10
2.3 Unity Server	11
2.4 Audio	12
2.5 Video Matrix Switch	12
3 Architektur der Komponenten	13
3.1 Unterteilung Module	15
3.2 Simplifiziertes Paketdiagramm	15
3.3 Sequenzdiagramm	16
4 Realisation	18
4.1 Stereoskopie	18
4.1.1 Kameraeinstellungen	18
4.1.2 GUI Kameras	19
4.1.3 Cursor Kameras	19
4.1.4 Sekundäre Kameras	19
4.1.5 VR Namespace Unity	20
4.1.6 Mosaic Settings	21
4.2 Immersion	23
4.2.1 Frustum allgemein	23
4.2.2 CAVE XXL Frustum	24
4.2.3 General Projection Matrix Frustum	25
4.2.4 Vergleich CAVE XXL und General Projection Matrix	26
4.3 Devices	28
4.3.1 Wand	28
4.3.2 Eyes	33
4.3.3 Gamepad	33
4.4 VRPN	34
4.4.1 Verwendung mit PPT Studio WorldViz	34
4.4.2 Datenverarbeitung im Unity	35
4.4.3 Datenveredlung im Unity	36
4.5 Unity Plugin	38
4.5.1 Konfiguration	38
4.5.2 Aufgabenverteilung	41
4.5.3 Deployment	48
4.5.4 Troubleshooting	53
4.5.5 Performance Verbesserungen	54
4.6 Warping	57
5 Demo Apps	59
5.1 Shooting Gallery	59
5.2 Model-Viewer	65
5.3 App Drittpartei	66
6 Testing	67
6.1 PPT-Studio und VRPN	67
6.2 Wrapping VRPN in C# und Unity	67
6.3 Unity Plugin Projekt als Debugger	68
6.4 Performance Tests Unity Server	69
7 Projektmanagement	73
7.1 Aufgabenstellung	73

7.2 Zieldefinitionen	73
7.2.1 Funktionale Anforderungen	74
7.3 Projektorganisation	75
7.3.1 Projektteam	75
7.3.2 Betreuer	75
7.3.3 Experte	75
7.4 Projektplanung, Meilensteine	75
7.5 Qualitätssicherung	76
7.6 Risikoanalyse	77
8 Fazit	78
9 Abbildungsverzeichnis	79
10 Tabellenverzeichnis	81
11 Quellcodeverzeichnis	81
12 Glossar	82
13 Literaturverzeichnis	86
14 Anhang	87

1 Einführung

1.1 Vorarbeiten Projekt 2

Im Rahmen der vorgängigen Projekt 2 Arbeit wurden verschiedene Methoden geprüft, wie eine Integration von Unity in den CAVE erfolgen kann. In einem ersten Schritt erfolgte eine Analyse der bereits verwendeten Frameworks (Equalizer⁶, Chromium⁷) und der Software MiddleVR⁸, die den Einsatz von Unity in einem CAVE deutlich vereinfachen sollte, sowie einem eigenen Plugin. Die Frameworks Equalizer und Chromium konnten sich Aufgrund folgender Punkte nicht durchsetzen:

- Projekt wurde eingestellt (Chromium)
- Kein Unity Support (Chromium und Equalizer)
- Struktur durch Framework gegeben, nicht für Unity adaptierbar (Wrapperklassen von Equalizer)
- Keyfeatures wie KI, Tracking und Physik nicht ansprechbar (jeweils nur OpenGL, Grafik)

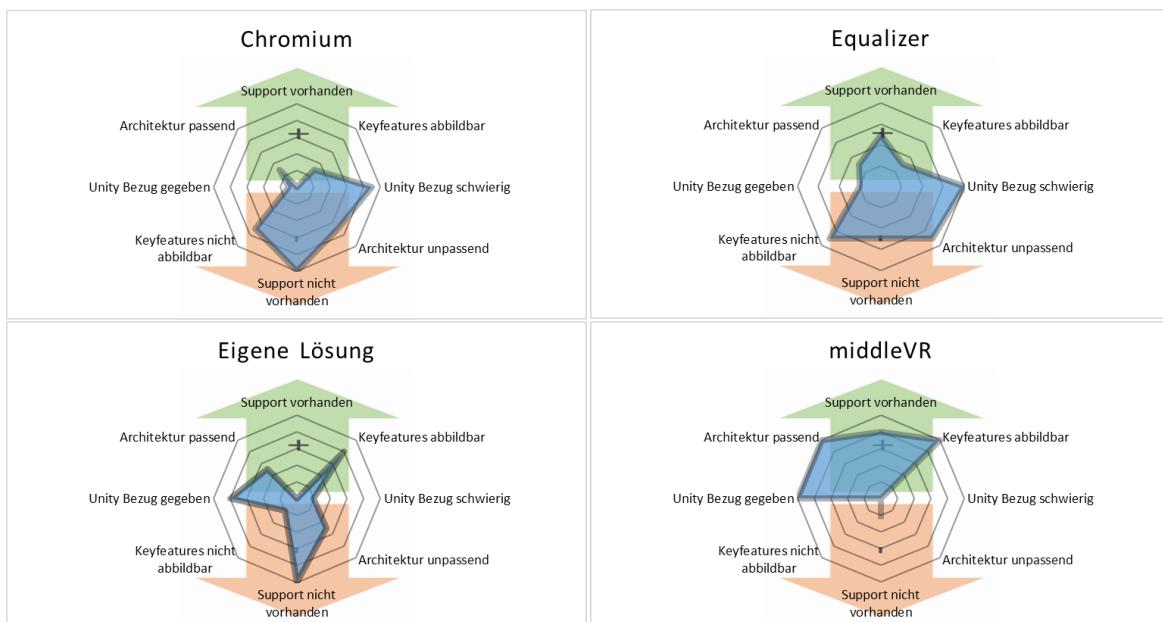


Abbildung 1: Vergleich Varianten Projekt 2

Weiter präsentierte sich die Software MiddleVR als günstiger Kandidat, hatte jedoch nach den vertieften Abklärungen signifikante Schwachstellen:

- Es wird nur das Transform⁹ des Unity Objekts über die Cluster (Position, Rotation und Massstab) synchronisiert
- Folglich fehlen Materials¹⁰, Farben und Partikel
- Die Flexibilität wird durch die proprietäre Software eingeschränkt
- Hohe Lizenzkosten durch Anzahl CPUs und GPUs

⁶ Equalizer – <http://www.equalizergraphics.com/>

⁷ Chromium – <http://chromium.sourceforge.net/>

⁸ MiddleVR – <http://www.middlevr.com/>

⁹ Transform Unity – <http://docs.unity3d.com/ScriptReference/Transform.html>

¹⁰ Material Unity – <http://docs.unity3d.com/ScriptReference/Material.html>

Die dadurch entstehende Abhängigkeit der Software, die Anforderungen an die Unity Applikationen (etliche Adaptionen am Code waren jeweils notwendig) und die nicht gebrauchte Fülle an Features waren der Grund, wieso eine eigene Umsetzung eines Unity Plugins erfolgen sollte.

Bezüglich der Infrastruktur wurden folgende Methoden analysiert und bewertet:

1. Mehrere Hosts / Mehrere GPUs / Mehrere Unity Instanzen
Basierend auf dem aktuellen CAVE Setting ist es möglich ein Clustering zu erstellen, wie es heute auch vom Equalizer-Framework verwendet wird. Die Unity Anwendung wird dann als komplette Netzwerkanwendung laufen, d.h. es kommt eine ähnliche Synchronisierung zum Zuge, welche auch bei Multiplayerspielen über das Internet verwendet wird.
2. Ein Host / Mehrere GPUs / Mehrere Unity Instanzen
Eine Möglichkeit wäre, eine Unity-Anwendung mittels mehreren GPUs auf einem Rechner parallel zu rechnen, um eine grösstmögliche Performance zu erreichen. Die Synchronisierung erfolgt demnach nicht über das Netzwerk, sondern würde auf dem gleichen Client stattfinden. Der mögliche Netzwerk-Flaschenhals wird vermieden. Obwohl alle Berechnungen auf einem Client laufen, gilt es trotzdem die Herausforderung der Synchronisierung zu meistern, analog dem Clustering über das Netzwerk.
3. Ein Host / Mehrere GPUs / Eine Unity Instanz mit Mosaic¹¹ Treiber
Mosaic ist eine Technologie von Nvidia, um mehrere Quadro¹²-Graphikkarten über den Treiber zu verlinken und auf einem Desktop darzustellen. Windows wird ein einzelner Output vorgegaukelt, auch wenn physisch mehrere GPUs mit multiplen Ausgängen angeschlossen sind.
4. Ein Host / Mehrere GPUs / Eine Unity Instanz
Ähnlich der Mosaic Variante ist ein Setting mit gängigeren, Nicht-Quadro-Grafikkarten denkbar. Der Einsatz vom Mosaic Treiber wird zwar verunmöglicht (fehlende Quadro Karten), die Anordnung der Screen erfolgt aber über die Windows-Einstellungen und die Unity Applikation kann in einer Fensteransicht über sämtliche Screens vergrössert werden.

Basierend auf Prototypen der verschiedenen Systeme, Prototypen und theoretischer Abklärungen konnte sich Variante 4 (Ein Host / Mehrere GPUs / Eine Unity Instanz mit Mosaic Treiber) klar hervorheben. Die dadurch erreichte hohe Flexibilität, die obsolete Synchronisierung und die gute Performance durch Verteilung der Last auf verschiedene GPUs waren ausschlaggebend.

¹¹ Nvidia Mosaic – <http://www.nvidia.com/object/nvidia-mosaic-technology.html>

¹² Nvidia Quadro – https://de.wikipedia.org/wiki/Nvidia_Qadro

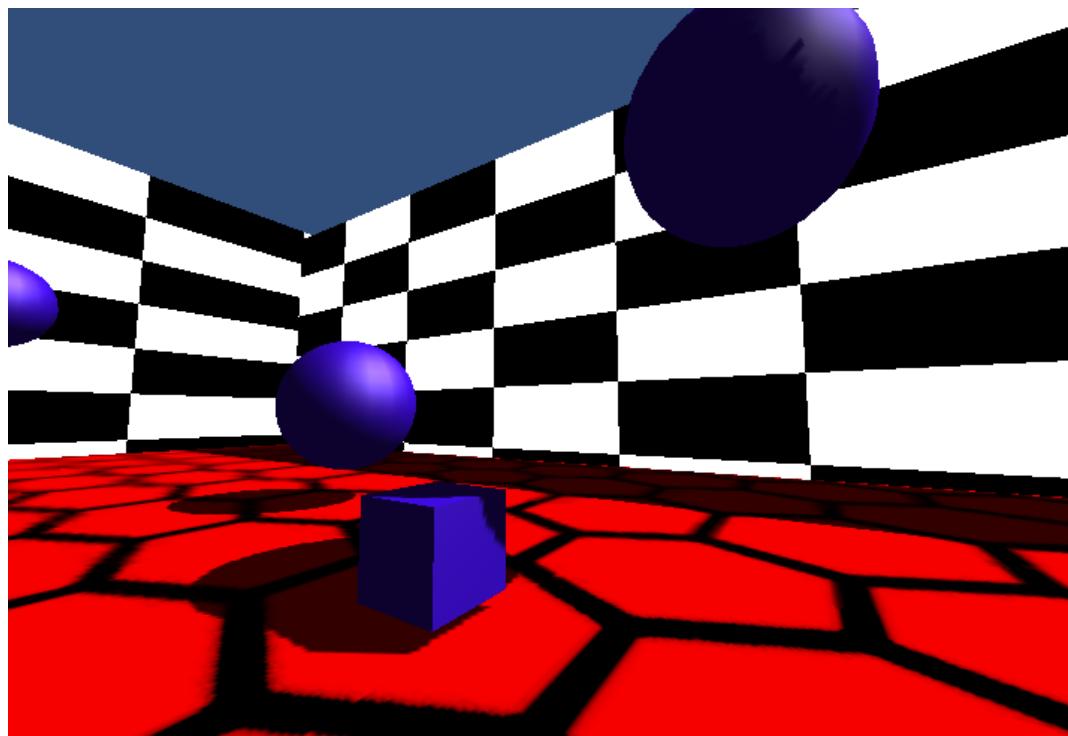


Abbildung 2: Prototyp 1 Projekt 2



Abbildung 3: Prototyp 2 Projekt 2

2 Infrastruktur

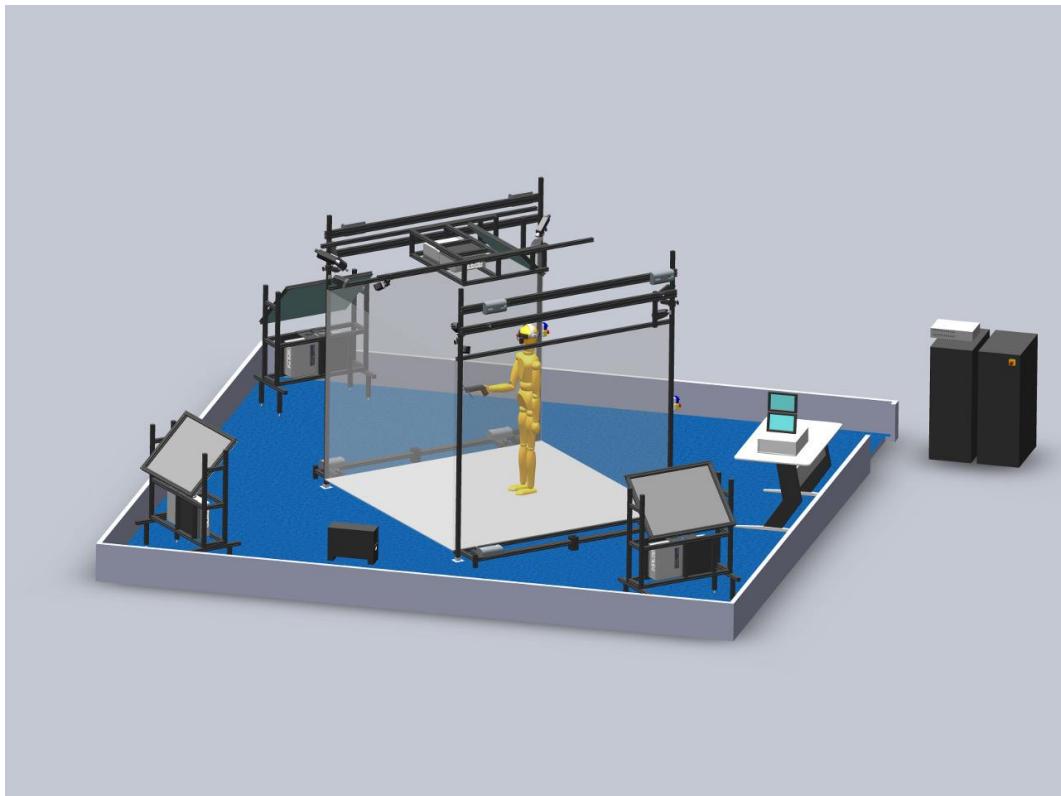


Abbildung 4: CAVE BFH

Der virtual reality haptic¹³ CAVE der BFH bietet Entwicklern und Forschern ein mächtiges Instrument, um hochrealistische immersive¹⁴ Applikationen zu bauen. Der CAVE ist eine kubische Konfiguration mit vier Leinwänden (Links, Vorne, Rechts, Boden) die von je 2 Projektoren bestrahlt werden. Um eine realistische 3D Stereoprojektion zu erschaffen, sind bei den Projektoren Polfilter angebracht und die Benutzer tragen entsprechend eine Brille mit Polfilter.

¹³ Haptic – <https://en.wikipedia.org/wiki/Haptics>

¹⁴ Immersion – [https://de.wikipedia.org/wiki/Immersion_\(virtuelle_Realit%C3%A4t\)](https://de.wikipedia.org/wiki/Immersion_(virtuelle_Realit%C3%A4t))

Alle Komponenten des CAVE und dessen Abhängigkeiten werden in der folgenden Grafik dargestellt:

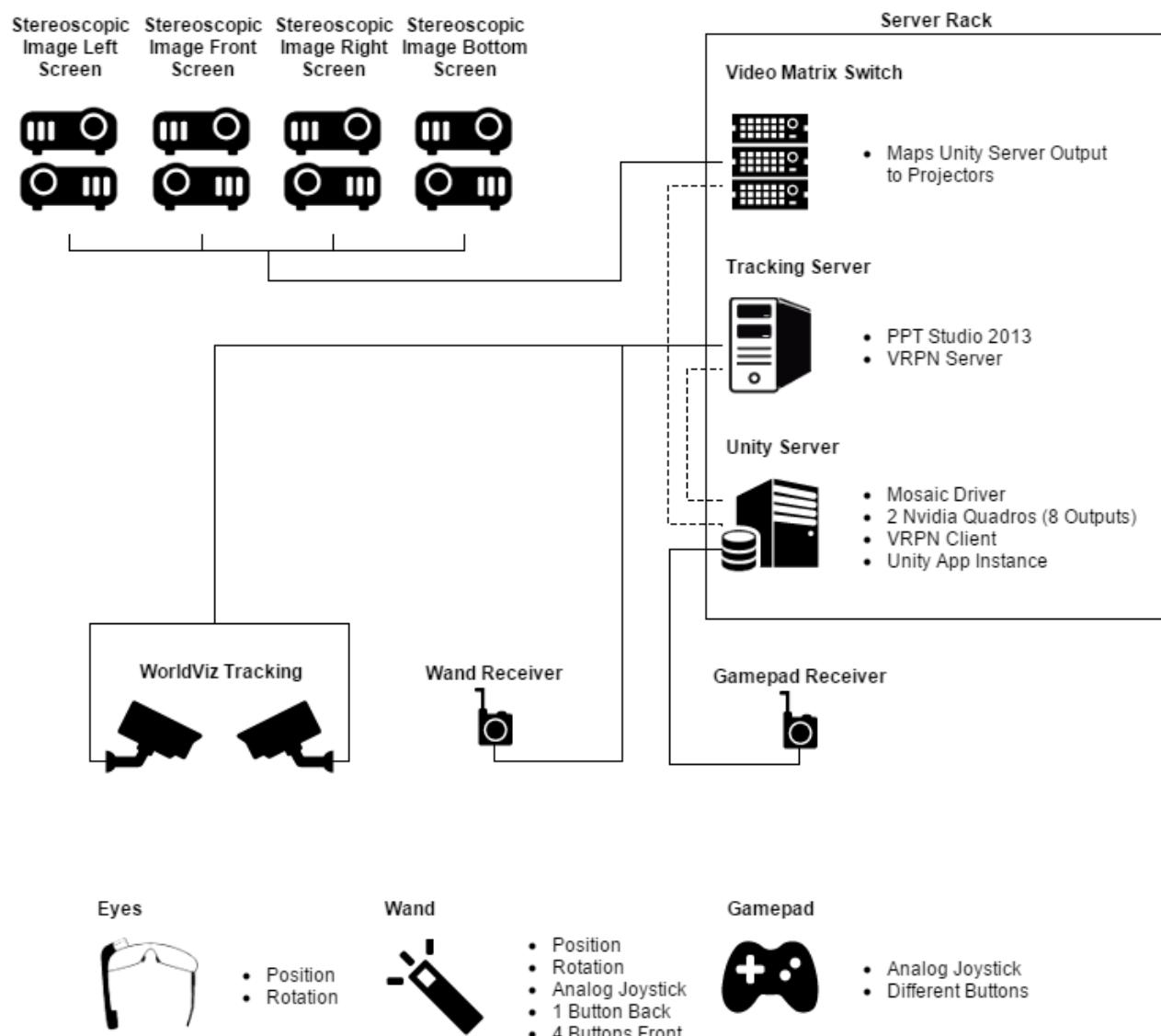


Abbildung 5: Infrastruktur CAVE

2.1 Trackingsystem WorldViz

Das Trackingsystem von WorldViz wurde integriert, um Positionen und Rotationen von verschiedenen Devices im CAVE zu erfassen. Die von den 10 WorldViz Kameras übermittelten Daten werden vom PPT Studio 2013¹⁵ zentral auf einem Server interpretiert, d.h. es werden Punkte im Raum und dessen Rotation der Devices berechnet, und können bei Bedarf über das VRPN¹⁶-Protokoll abgefragt werden.

¹⁵ PPT Studio 2013 – <http://www.worldviz.com/virtual-reality-motion-tracking/>

¹⁶ VRPN – <https://github.com/vrpn/vrpn/wiki>

2.2 Devices

- **Eyes**

Die Eyes von WorldViz sind Brillen mit Polfilter und zwei montierten Infrarot-Trackern. Somit lassen sich die Position des Kopfes und dessen Rotation auf zwei Achsen (Yaw und Roll¹⁷) bestimmen. Die dritte Achse (Pitch) kann mit lediglich zwei Trackern nicht ermittelt werden. Dazu wären mindestens drei LEDs oder ein Gyrometer¹⁸ notwendig.



Abbildung 6: PPT Eyes, Quelle: www.worldviz.com

- **Wand**

Der Wand von WorldViz ist das primäre Eingabegerät. Nebst zwei Infrarot-Tackern, welche für die Positions- und Rotationsbestimmung verwendet werden, ist ein Gyrometer integriert, um noch präziser Drehungen feststellen zu können. Dadurch wird auch die fehlende Rotationsachse (fehlende bei den Eyes) kompensiert und es können Yaw, Roll und Pitch ermittelt werden.

Als Inputs dienen ein analoger Joystick, vier Buttons auf der Vorderseite sowie ein Button auf der Rückseite.

¹⁷ Yaw, Pitch, Roll – https://en.wikipedia.org/wiki/Six_degrees_of_freedom

¹⁸ Gyrometer – <https://de.wikipedia.org/wiki/Gyrometer>



Abbildung 7: PPT Wand, Quelle: www.worldviz.com

- **Gamepad**

Ein weiteres Inputgerät ist ein handelsübliches Gamepad. Frei von jeglichem Tracking wird einzig die Unity Applikation gesteuert.



Abbildung 8: Gamepad, Quelle: www.androidrundown.com

2.3 Unity Server

Der leistungsstarke Unity Server ist der Knotenpunkt des gesamten Systems. Auf diesem Rechner läuft die Unity Applikation mit dem konfigurierten Unity Plugin, welches die Trackingdaten vom Trackingserver abgreift und das korrekte Rendering in der Unity Applikation für die Projektoren Aufteilung sicherstellt. Mittels Mosaic, einem speziellen Treiber von Nvidia der die Aufteilung auf mehrere Grafikkartenausgänge übernimmt, werden alle Projektoren korrekt für die stereoskopische Projektion angesprochen.

2.4 Audio

Um die Immersion weiter zu steigern, ist ein 3D Soundsystem mit vier Lautsprechern in Betrieb.

2.5 Video Matrix Switch

Weil parallel weitere Clients in Betrieb sind und Bilddaten für den CAVE liefern können, wird ein Video Matrix Switch eingesetzt, um die verschiedenen Inputquellen auf die 8 bestehenden Projektoren abilden zu können.

3 Architektur der Komponenten

Damit das Unity Plugin einwandfrei läuft, ist eine enge Zusammenarbeit zwischen realen und virtuellen Komponenten vonnöten. Beispielsweise haben die Inputdevices Wand und Eyes ein virtuelles Pendant, um dessen Eigenschaften besser verwerten und weiterverarbeiten zu können. Zusätzlich ermöglicht dieses Konzept ein sauberes Debugging und eine ansprechende Visualisierung.

Die folgende Grafik zeigt die Verschmelzung der realen und virtuellen Welt mit je einem halben CAVE und deren spezifischen Komponenten auf.

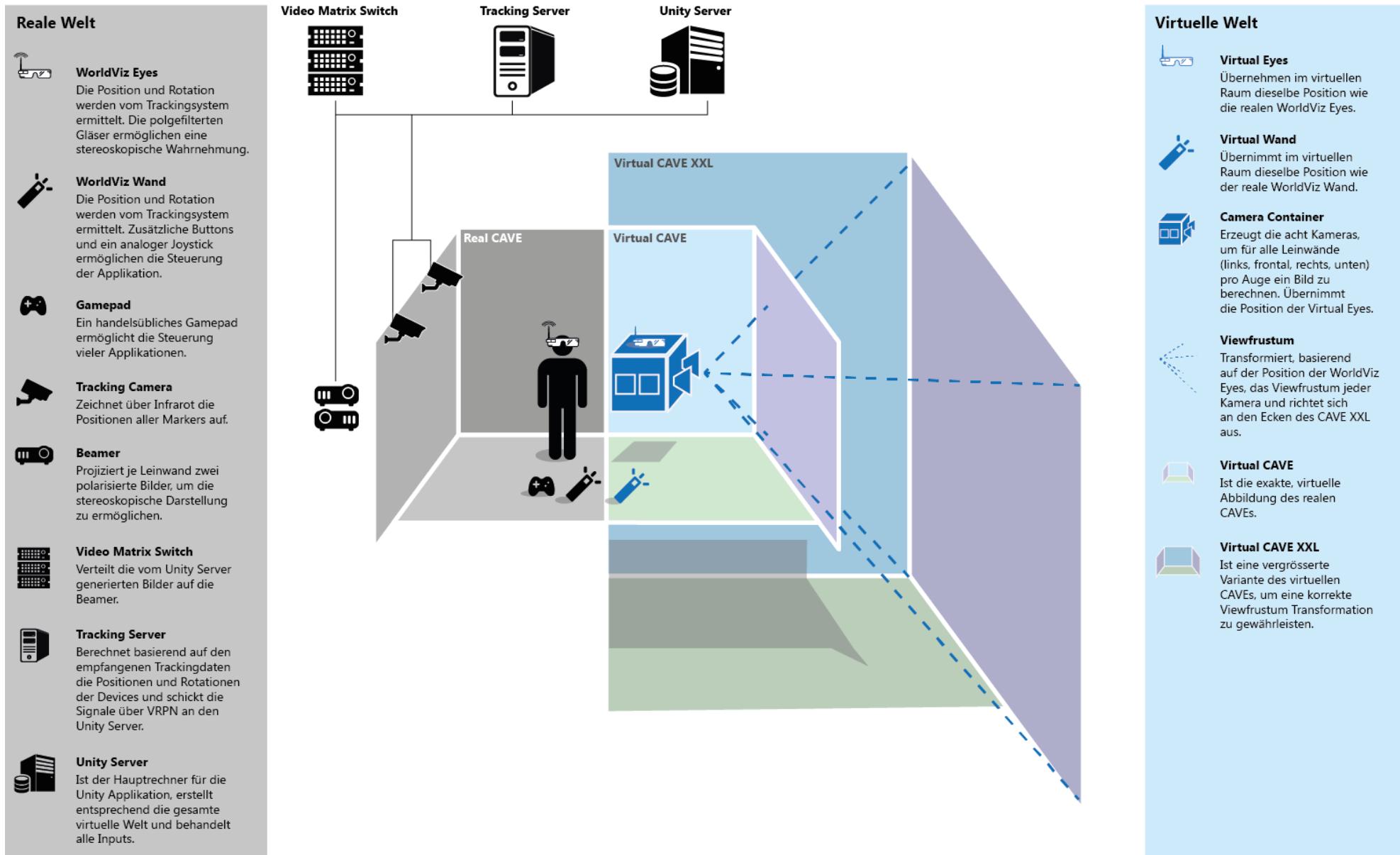


Abbildung 9: Übersicht der Komponenten

3.1 Unterteilung Module

Das modulare Konzept der objektorientierten Programmierung wurde streng eingehalten. Jede einzelne Komponente verfügt über einen klar abgegrenzten Aufgabenbereich und hat möglichst wenige Verknüpfungen zu anderen Modulen. Diese für das menschliche Denken intuitive Aufteilung erleichtert die Programmierung enorm und vereinfacht den Einstieg und die Einarbeitung in den Code für Außenstehende.

3.2 Simplifiziertes Paketdiagramm

Folgendes Diagramm zeigt das Zusammenspiel zwischen den Komponenten Unity, Unity-MonoBehaviour, dem Plugin und externen Libraries. Aus Simplifizierungsgründen wurde bei Unity nur die Mainkamera angegeben. Dort könnte auch ein Fahrzeug, «First Person Controller» oder ähnliches sein. Komponenten welche von MonoBehaviour abhängig sind, verwenden die Update, FixedUpdate und weitere Routinen.

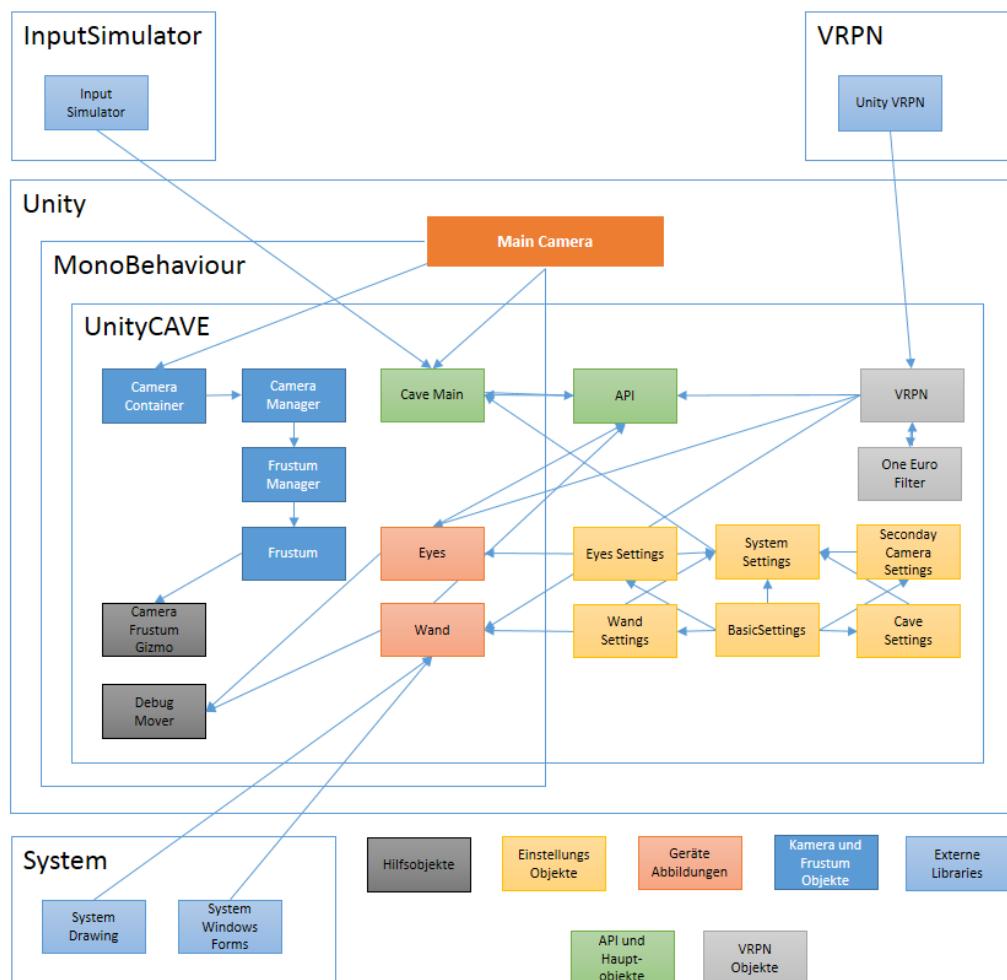


Abbildung 10: Paketdiagramm simplifiziert

3.3 Sequenzdiagramm

Folgendes Diagramm zeigt ein Update, also eine Abfolge für die Berechnung eines Frames, welches Unity durchführt. Die physikalischen Komponenten Wand und Eyes werden nicht dargestellt, jedoch das virtuelle Pendant in Unity. Der Benutzer macht im CAVE eine Bewegung, die vom Trackingsystem wahrgenommen und mittels Unity Plugin weiterverarbeitet wird um schlussendlich ein Bild auf den Projektoren anzuzeigen.

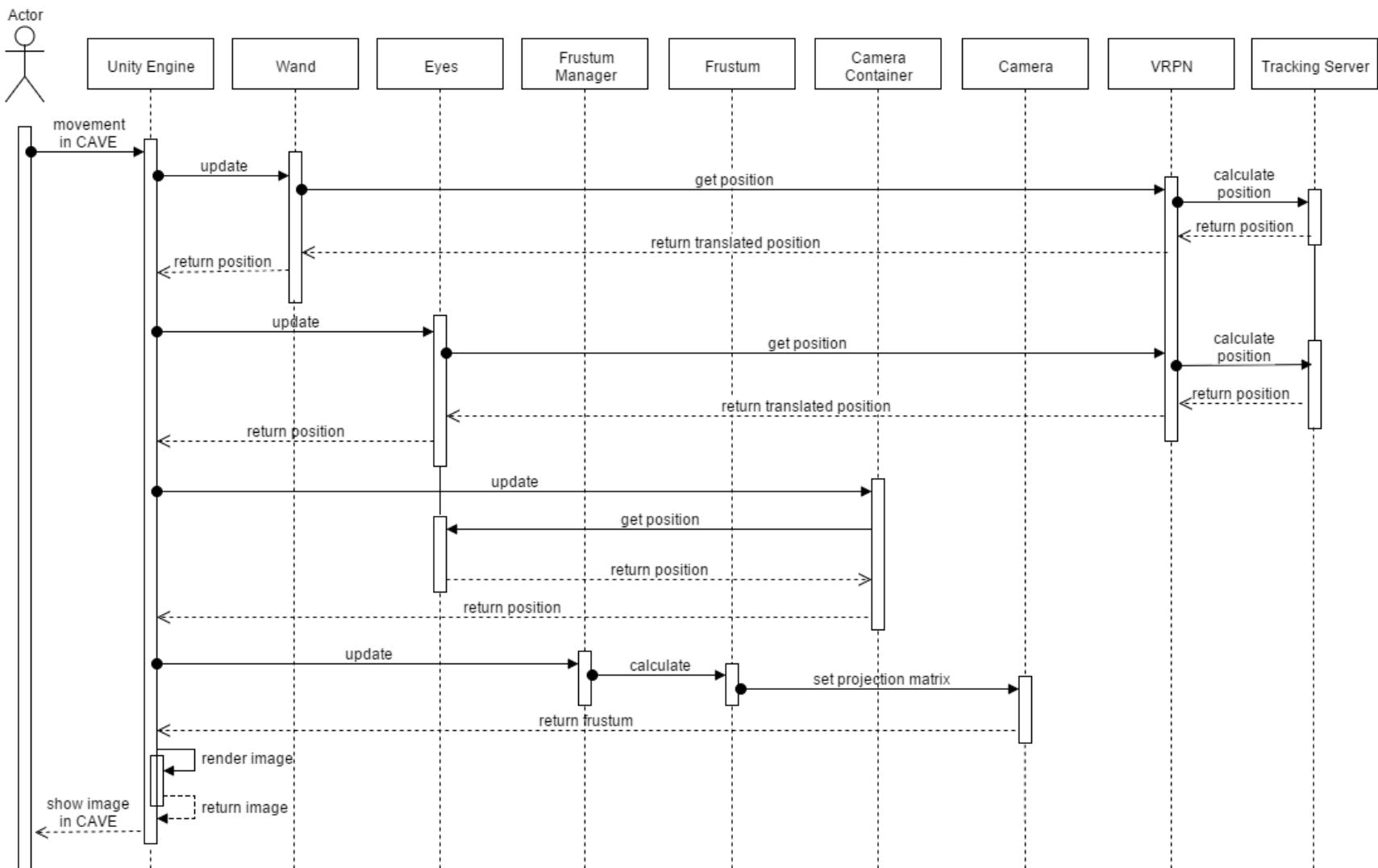


Abbildung 11: Sequenzdiagramm

4 Realisation

Basierend auf der gegebenen Infrastruktur, der Aufgabenstellung und der konzeptuellen Erarbeitung erfolgt die Realisation. Auch hier wurden die zentralen Aspekte wie eine gute Immersion und die Einfachheit der Bedienung in den Mittelpunkt gerückt.

Das folgende Kapitel zeigt die Behandlung der vier Hauptstandbeine Stereoskopie, Immersion, Devices und VRPN. Zum Schluss wird das Plugin und dessen Einstellungsmöglichkeiten erläutert. Als zusätzliche Anforderung wurde eine Analyse vorgenommen, wie der von den Beamern erzeugten Verzerrung der Bilder entgegengewirkt werden kann.

4.1 Stereoskopie

Damit der Benutzer die Applikation in der dritten Dimension erleben und eine Polfilterung der Bilder erfolgen kann, muss das Plugin etliche Kameras zur Verfügung stellen.

4.1.1 Kameraeinstellungen

Für jede Seitenwand des CAVEs werden mehrere Kameras instanziert und dem CameraContainer hinzugefügt:

- 3D Render Kamera für das linke Auge
- 3D Render Kamera für das rechte Auge
- GUI Render Kamera, aktiv falls sich GUI Elemente auf dieser Seite befinden
- Cursor Kamera, aktiv falls sich der Cursor auf dieser Seite befindet

Die 3D Render Kameras werden wie folgt instanziert:

- An der Position der Hauptkamera (übernommen von der Applikation)
- Rotiert um jeweils 90, 0, -90 Grad um Y für die Seitenwände
- Rotiert um 90 Grad um X für den Boden
- Verschoben um die halbe Augendistanz für den passiven Stereoeffekt



Abbildung 12: Einstellungen Augendistanz

```

Left = new CameraInfo
{
    Cam = _cameraLeftLeft,
    CamGUI = API.Instance.Cave.CaveSettings.GUILocation == BasicSettings.Sides.Left ?
        Instantiate(_cameraLeftLeft) : null,
    CamCursor = Instantiate(_cameraLeftLeft),
    Offset = new Vector3(0f, 0f, -(API.Instance.Cave.CaveSettings.EyeDistance / 2))
},

Right = new CameraInfo
{
    Cam = _cameraLeftRight,
    CamGUI = API.Instance.Cave.CaveSettings.GUILocation == BasicSettings.Sides.Left ?
        Instantiate(_cameraLeftRight) : null,
    CamCursor = Instantiate(_cameraLeftRight),
    Offset = new Vector3(0f, 0f, +(API.Instance.Cave.CaveSettings.EyeDistance / 2))
}

```

Quellcode 1: Kamerainstanziierung

4.1.2 GUI Kameras

Der Benutzer kann die Position der GUI-Elemente auf eine CAVE-Seite festlegen. Beim initialisieren der Applikation werden sämtliche 2D-Elemente gesucht und so im virtuellen Raum platziert, dass die Darstellung direkt auf der CAVE-Leinwand erfolgt und kein Tiefeneffekt entsteht. Gleichzeitig werden spezielle GUI-Kameras für das linke sowie das rechte Auge instanziert, die einzig dazu da sind, GUI-Elemente zu rendern. Diese zusätzlichen Kameras sind notwendig, damit keine 3D-Objekte die Sicht auf die UI-Elemente nehmen.

4.1.3 Cursor Kameras

Ähnlich wie für die GUI-Elemente werden auch für den Cursor spezielle Kameras erstellt, jedoch insgesamt 8 Stück (pro Seite je eine Kamera pro Auge). Je nachdem wohin der Wand zielt, also wo der Cursor dargestellt werden soll, aktivieren sich diese Cursor Kameras und stellen einen Windowscursor dar. Der Systemcursor kann nicht verwendet werden, weil der nur einmal vorhanden ist und somit nur für ein Auge angezeigt werden könnte. Deshalb wird die Position bestimmt und mit Hilfe von Unity zwei Kopien des Cursors dargestellt. Gleichzeitig wird der richtige Cursor ausgeblendet, damit kein Konflikt entsteht.

Falls ein Custom Cursor gewünscht ist, kann dieser dem Plugin angegeben werden und das entsprechende Sprite wird angezeigt.

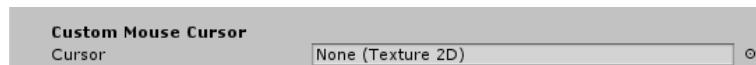


Abbildung 13: Custom Cursor

4.1.4 Sekundäre Kameras

Je nach Applikation kann es vorkommen, dass neben einer Hauptkamera (deren Viewport an die Wände des CAVEs projiziert werden), noch mehrere sekundäre Kameras vorhanden sind. Dies können eine Minimap, ein Querschnitt eines Körpers, eine Aussenkamera oder auch nur Rückspiegel bei einem Fahrzeug sein.

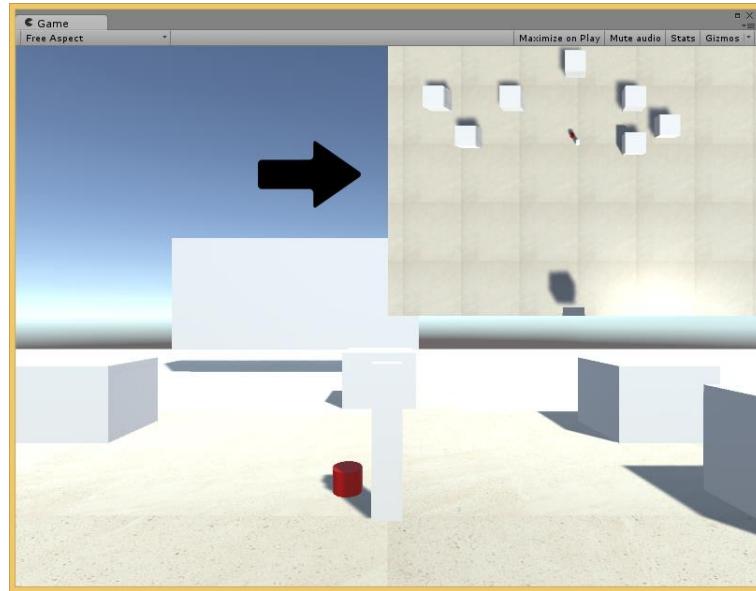


Abbildung 14: Minimap

Das Plugin löst diese Anforderung wie folgt:

- Der Benutzer kann entscheiden, auf welche Seite er die sekundäre Kamera zuordnen will.
- Es müssen lediglich all die gewünschten Kameras dem Plugin übergeben werden.

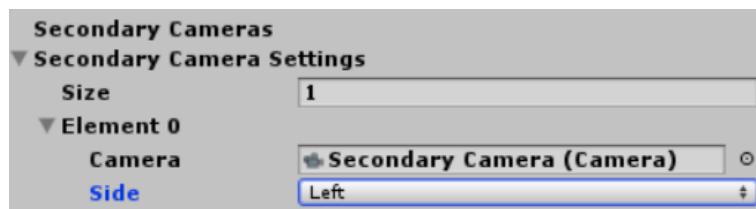


Abbildung 15: Sekundäre Kameras Einstellungen

4.1.5 VR Namespace Unity

Unity verfügt seit Version 5.1 über einen VR Namespace¹⁹. Dieser wird verwendet, um externe Plugins/ SDKs, wie das der Oculus Rift, abzugrenzen. Mit diesem Namespace möchte Unity folgende Ziele erreichen:

- Vermeiden von Konflikten der Plugins untereinander
- Mehrere VR Geräte brauchen nicht mehr mehrere Plugins (Reduzierung Aufwand)
- Neue SDKs der Hersteller können mit älteren Versionen der Spiele nicht kompatibel sein

¹⁹ VR Namespace Unity – <http://docs.unity3d.com/Manual/VROverview.html>

Das Basis-API unterstützt aktuell nur folgende Features:

- **Automatisches Stereodisplay**

Es ist nicht mehr nötig, wie bisher zwei Kameras zu instanziieren, dies wird von Unity übernommen. Das funktioniert jedoch nur für alle Kameras, die nicht auf eine Textur gerendert werden.

- **Head-Tracking als Input**

Dieser Input wird analog dem umgesetzten Unity Plugin behandelt.

Da dieses Feature erst mit Version 5.1 (Release Juni 2015) verfügbar war und es im Moment nur über Basisfähigkeiten verfügt, wird dies nicht verwendet.

(Unity, 2015)

4.1.6 Mosaic Settings

Mosaic ist eine Technologie von Nvidia, um mehrere Graphikkarten über den Treiber zu verlinken und auf einem Desktop darzustellen. Windows wird ein einzelner Output vorgegaukelt, auch wenn physisch mehrere GPUs mit multiplen Ausgängen angeschlossen sind.

(Nvidia, 2015)

Die Mosaic Einstellungen des Unity Servers sind wie folgt:

- 1280 Pixel auf 1024 Pixel Auflösung pro Ausgang
- Eine 2 auf 4 Verteilung der acht Bildschirme
- Gesamtauflösung von 5120 Pixel auf 2048 Pixel

Karte 1 Ausgang 1 Position 0;0	Karte 1 Ausgang 2 Position 0;1	Karte 2 Ausgang 1 Position 0;2	Karte 2 Ausgang 2 Position 0;3
Karte 1 Ausgang 3 Position 1;0	Karte 1 Ausgang 4 Position 1;1	Karte 2 Ausgang 3 Position 1;2	Karte 2 Ausgang 4 Position 1;3

Tabelle 1: Mosaic Setting schematisch

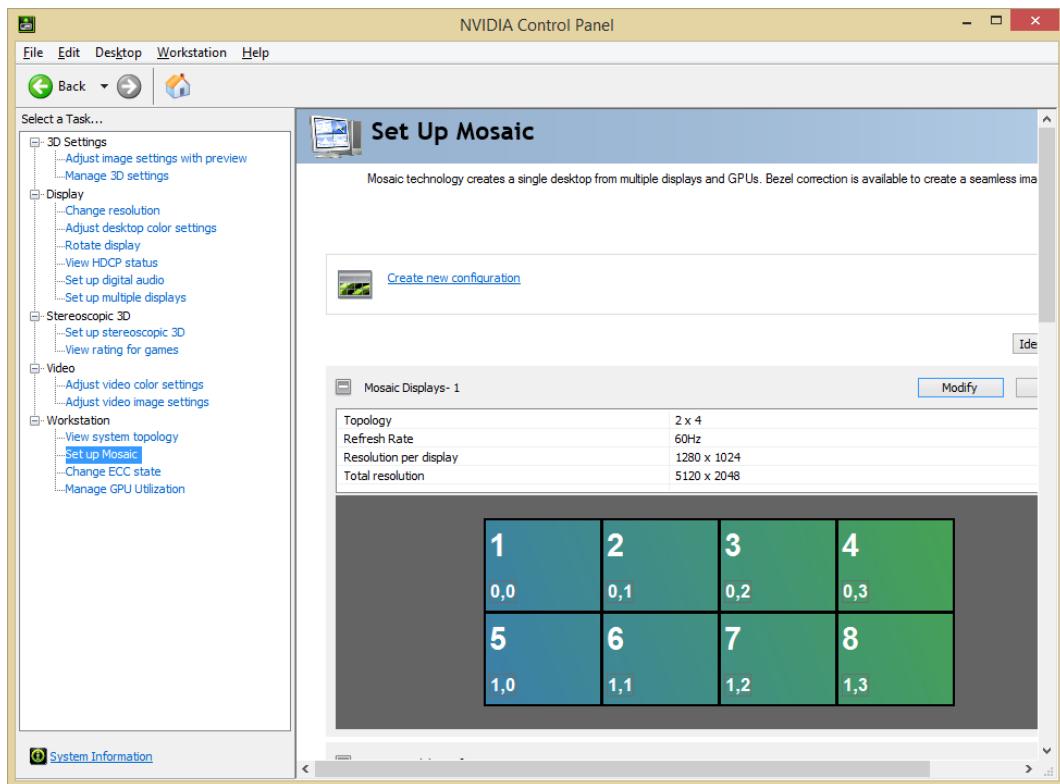


Abbildung 16: Mosaic Setting Unity Server

4.2 Immersion

Das Ziel ist, die Wahrnehmung der eigenen Person weitgehend zu verhindern und den Benutzer in die virtuelle Welt eintauchen zu lassen. Dazu gehört eine realistische Perspektive auf die künstliche Welt zu gewährleisten.

4.2.1 Frustum allgemein

Für eine optimale Projektion muss das bereits vorhandene Frustum der Applikation angepasst werden. Das Frustum wird im CAVE direkt bestimmt durch die geometrischen Eigenschaften der Wände des CAVES (initial) und wird durch die Position des Benutzers im CAVE aktualisiert (runtime). Ganz allgemein bildet das Frustum das 3D Bild der Computergraphik auf einen zweidimensionalen Bildschirm ab.

Es gibt mehrere Möglichkeiten ein Frustum aufzubauen, die folgenden zwei Methoden wurden umgesetzt:

- Frustum mit einer Projektionsfläche XXL (Seitenwand des CAVE virtuell ~30m entfernt, Dimensionen beibehalten)
- Frustum mit der genauen Projektionsfläche der CAVE-Wand und einer Off-Axis Projektion

Diese unterschiedlichen Möglichkeiten haben sich aus dem Adaptieren des bereits vorhandenen Frustums, den eigenen Überlegungen und der Adaptierung der generalisierten Projektionsmatrix des Ur-CAVEs ergeben.

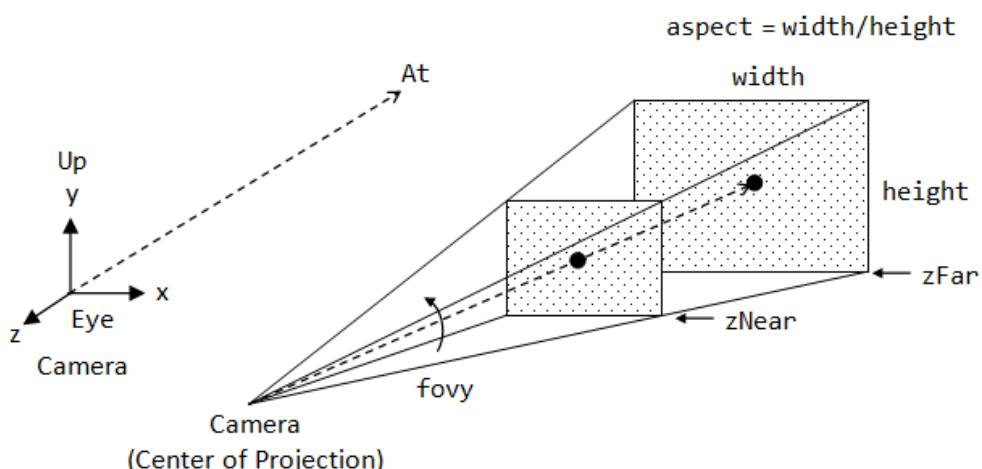


Abbildung 17: Frustum, Quelle: www.stackoverflow.com

Ein Frustum wird in Unity wie folgt dargestellt. Sichtbar hier ist, dass die Seitenwand des CAVE XXL die Plane bildet, woran das Frustum aufgespannt wird.

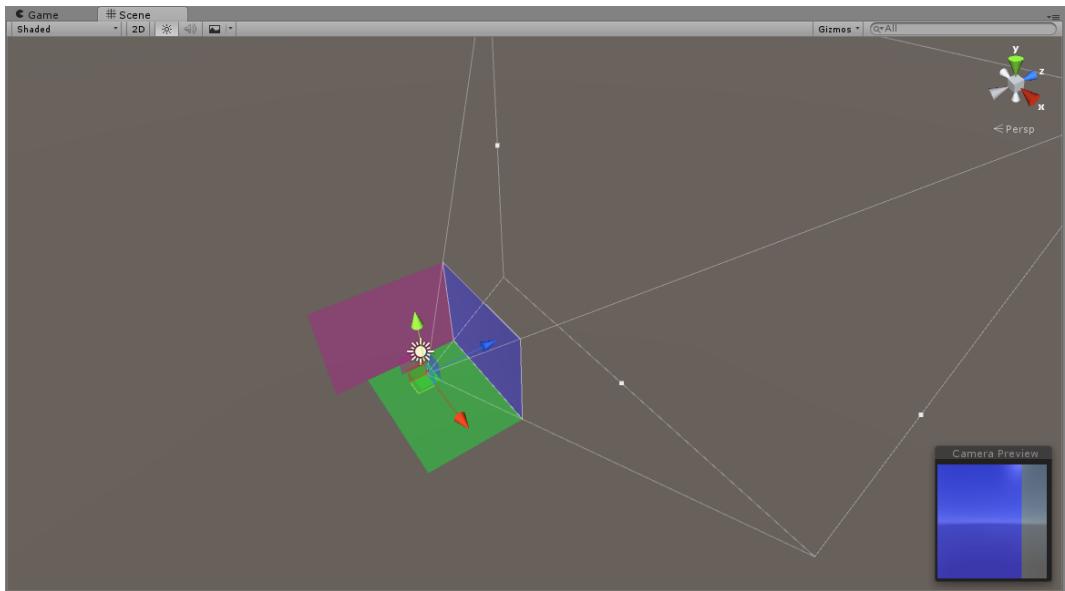


Abbildung 18: Unity Plugin Frustum 1

Mit allen 8 Kameras ergibt sich folgendes Bild:

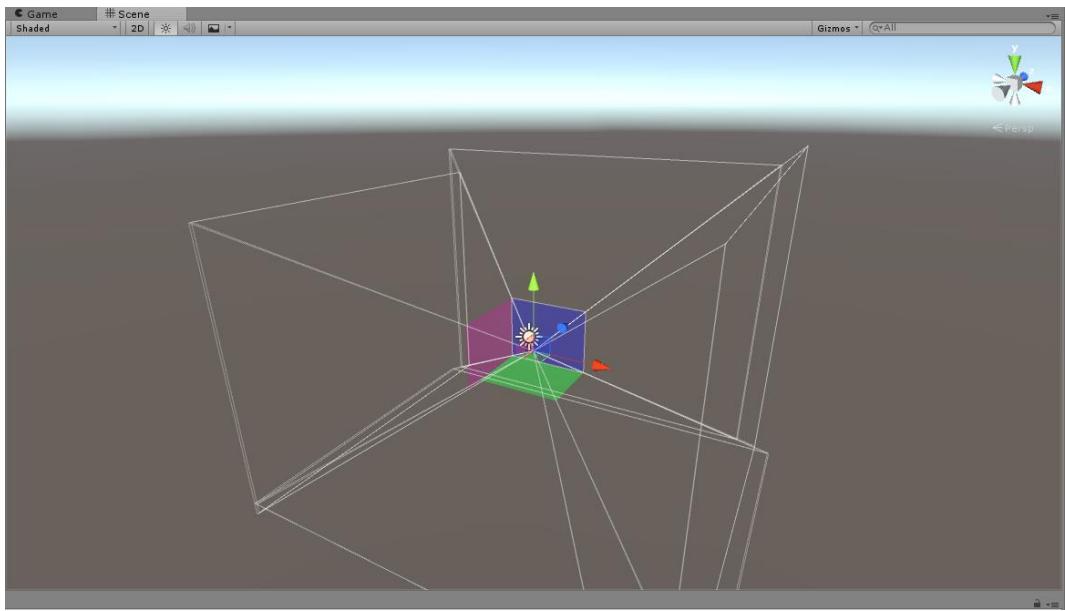


Abbildung 19: Unity Plugin Frustum 2

4.2.2 CAVE XXL Frustum

Falls das Frustum auf die realen CAVE Seitenwand projiziert wird, vergrössert sich das Field of View²⁰ (FoV) wenn man sich der Leinwand nähert. Bei einem FoV von beispielsweise 120 Grad werden die Objekte verzogen, obwohl die Projektionsfläche dieselbe bleibt.

²⁰ Field of View – https://en.wikipedia.org/wiki/Field_of_view

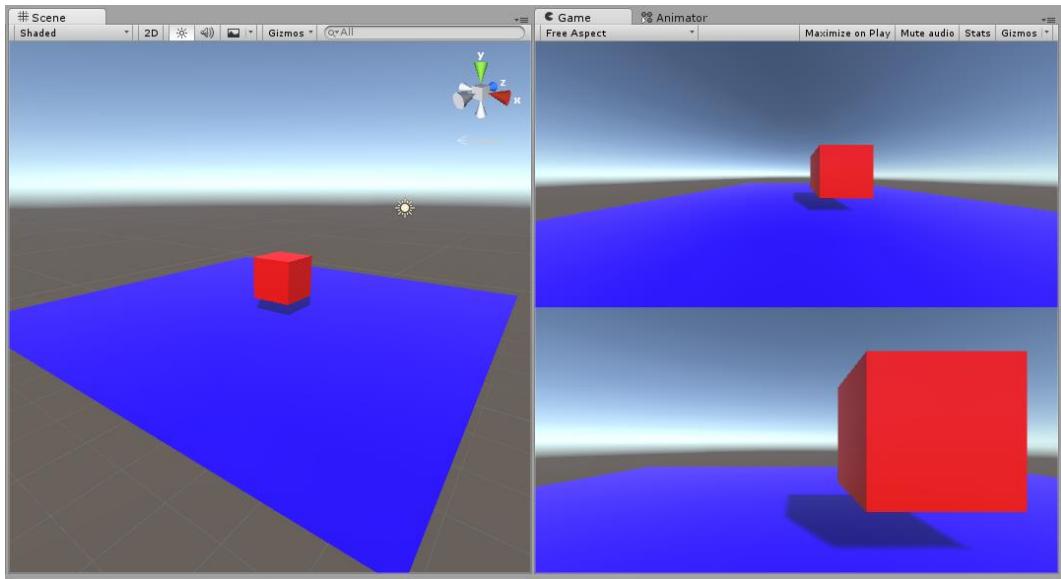


Abbildung 20: Vergleich FoV 120 & 60 Grad

Um diesen Effekt abzuschwächen, wird in diesem Betriebsmodus nicht die Seitenwand des CAVEs verwendet, um das Frustum aufzuspannen, sondern die Seitenwand des CAVE XXL.

- Es kann die «Generalized Perspective Projection» von Robert Kooima verwendet werden, welche 2008 publiziert wurde. (Kooima, 2008)
- Da sich der Betrachter der Seitenwand des CAVE XXL nie gross nähert, tritt der oben genannte Effekt nicht auf.
- Die Frustums überschneiden sich nicht, noch haben sie Lücken.

4.2.3 General Projection Matrix Frustum

In diesem Betriebsmodus wird das FoV nach der Berechnung des Frustums noch angeglichen.

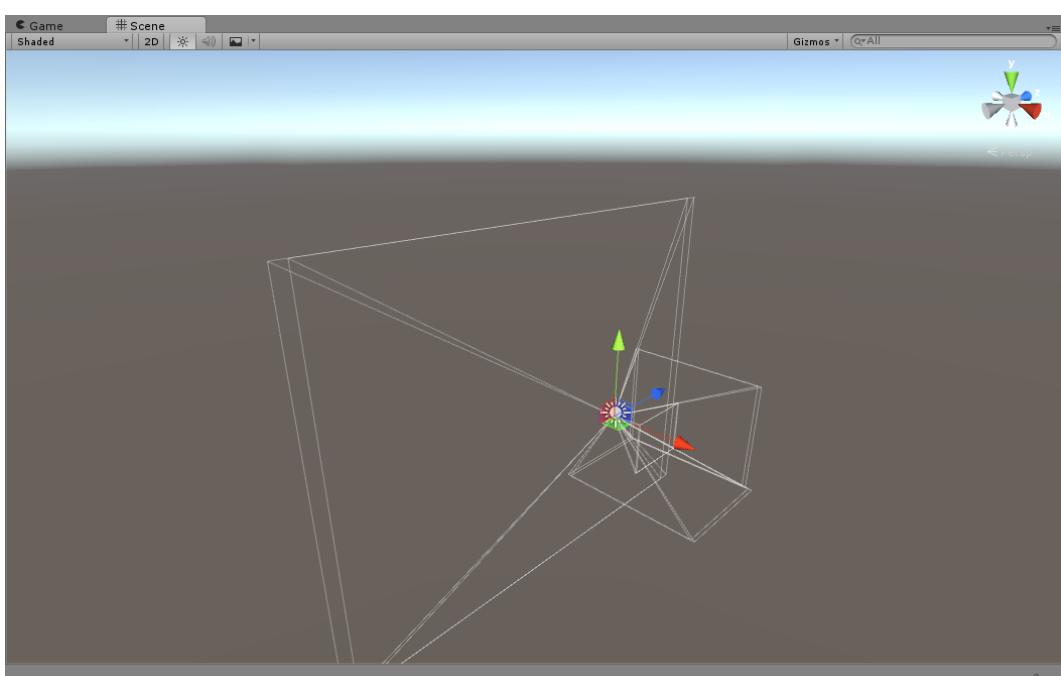


Abbildung 21: Unity Plugin Frustum GGM

Es ist sichtbar, dass hier unterschiedlich grosse Frustums in Verwendung sind, je nachdem wie nahe an der CAVE Wand man sich befindet. Die FoVs werden aufgrund des Verhältnisses von Höhe und Breite des Frustums nachgeglichen.

4.2.4 Vergleich CAVE XXL und General Projection Matrix

Grundlegend basieren beide Berechnungen auf der «GPP» (Kooima, 2008). Das Frustum wird anhand einer Plane im 3D Raum aufgespannt.

Vorteile CAVE XXL

- Der Benutzer nähert sich der Seitenwand XXL nie so stark, dass ein unrealistisches FoV erzeugt wird
- Die Vergrösserung aller Distanzen erzeugt eine grössere Genauigkeit
- Weniger Berechnung sind nötig, da nicht noch das FoV nachberechnet werden muss und dies dann die bereits berechnete Projektionsmatrix adjustiert

Vorteile General Projection Matrix

- Konstrukt CAVE XXL nicht notwendig
- Anpassung FoV nach verbreiteter und akzeptierter Methode

Da der CAVE XXL für weitere Features (Raycast²¹, 2D GUI) verwendet wird, halbieren sich die Vorteile der General Projection Matrix. Zusätzlich ergibt das Frustum nach CAVE XXL ein besseres 3D Bild. Somit wird standardmässig diese Methode verwendet. Zu Demonstrationszwecken sind aber beide Methoden vorhanden.

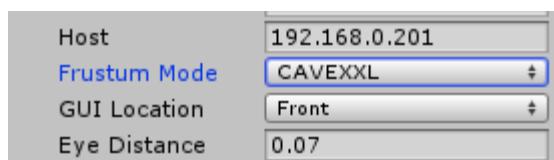


Abbildung 22: Frustum Mode

²¹ Raycast – https://en.wikipedia.org/wiki/Ray_casting

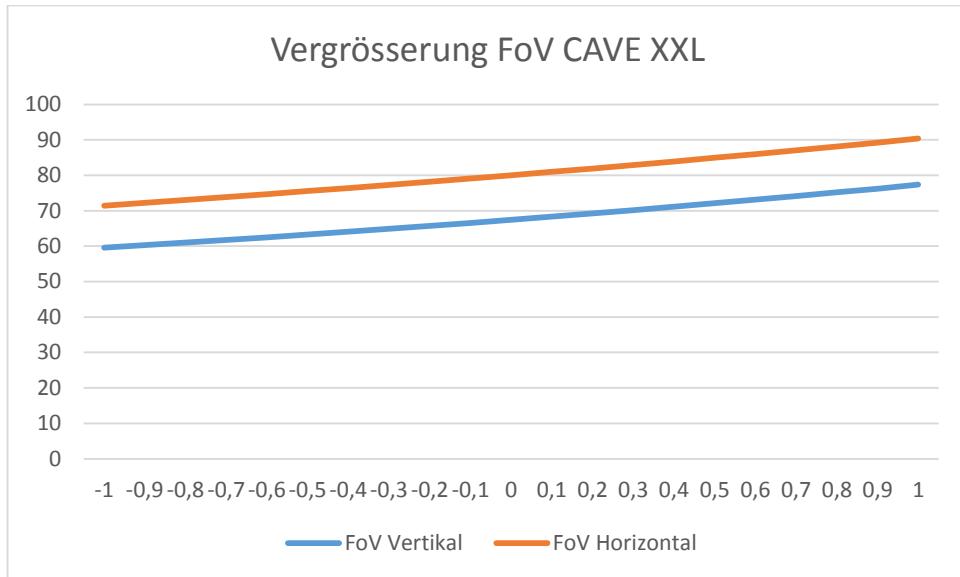


Abbildung 23: Frustum CAVE XXL FoV Anpassung

Die obige Abbildung zeigt die Vergrösserung des FoVs bei einer Bewegung im CAVE (-1 Ausserhalb des CAVEs, 1 direkt an der Projektionswand, Unity Einheiten).

4.3 Devices

Anstelle einer Maus oder Tastatur stehen drei Input-Devices zur Verfügung, welche das Plugin interpretieren muss. Dazu gehören der WorldViz Wand, die WorldViz Eyes sowie ein handelsübliches Gamepad. Alle Geräte haben eigene Anforderungen und Eigenschaften, die es zu verarbeiten gilt.

4.3.1 Wand

Die Interpretation des Wands ist ein zentrales Element für die Verwendung des CAVEs mit Unity. Er dient dazu, Objekte in der virtuellen Welt zu bewegen oder zu rotieren.

- **Virtueller CAVE**

Damit die Verarbeitung der vom Wand gelieferten Informationen vereinfacht werden können und eine visuelle Darstellung möglich ist, wurde ein virtueller CAVE in Unity erstellt, welcher dieselben Dimensionen wie der reale CAVE der BFH hat. Diese genaue Adaptierung ist möglich, weil in Unity die verwendete Grösseneinheit einem Meter in der realen Welt entspricht.

Implementiert wurde das mittels einem Prefab²², welches einmalig in der Hierarchie liegt.

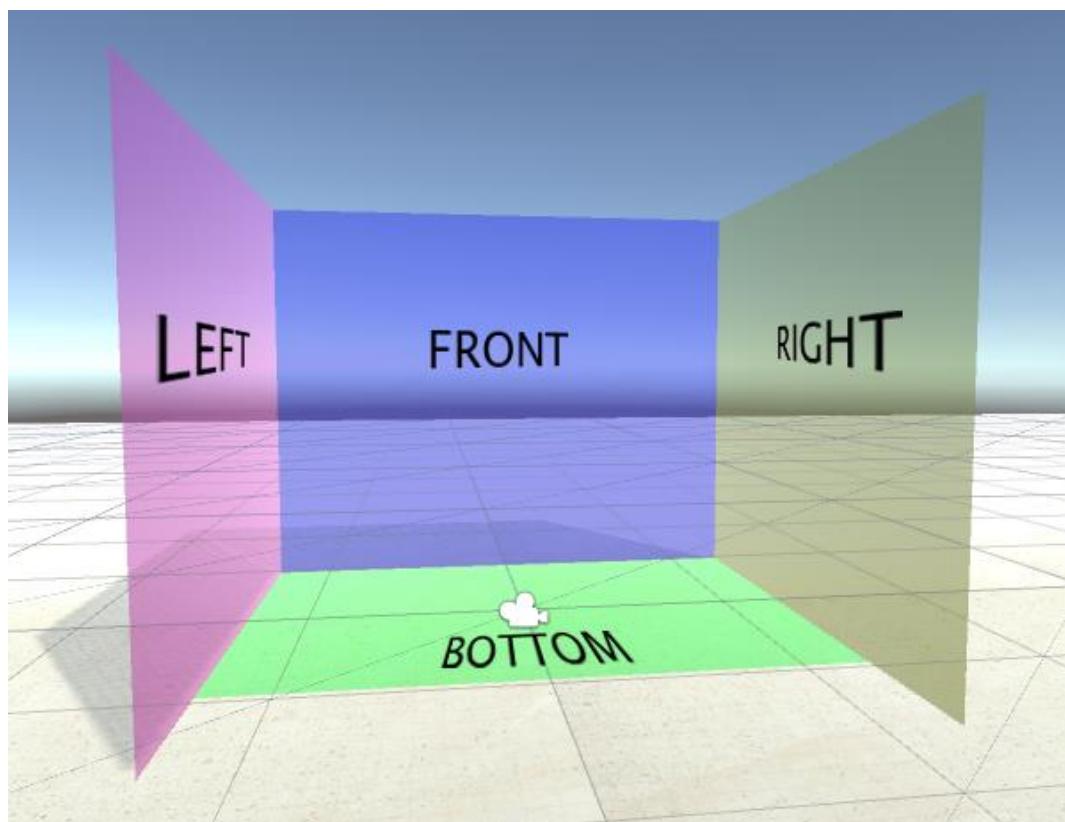


Abbildung 24: Unity Plugin, virtueller CAVE

Zusätzlich zum CAVE sind auch die beiden Devices Eyes und Wand als virtuelle Objekte in der Hierarchie der Applikation.

Der virtuelle CAVE übernimmt die Position und Rotation der Hauptkamera der Applikation. Die Korrelation zwischen CAVE und Wand / Eyes kann nur bestehen, wenn sich der Spieler in der

²² Unity Prefab – <http://docs.unity3d.com/Manual/Prefabs.html>

virtuellen Welt auch im virtuellen CAVE befindet. Die Hauptkamera wird rein durch die Unity Applikation gesteuert und das Plugin hat keinerlei Einfluss darauf, um den Spielmechanismus nicht zu stören. Damit aber die Checks, wie sich die Devices im CAVE befinden, nach wie vor gemacht werden können, verändert der CAVE die Position und Rotation analog der Hauptkamera.

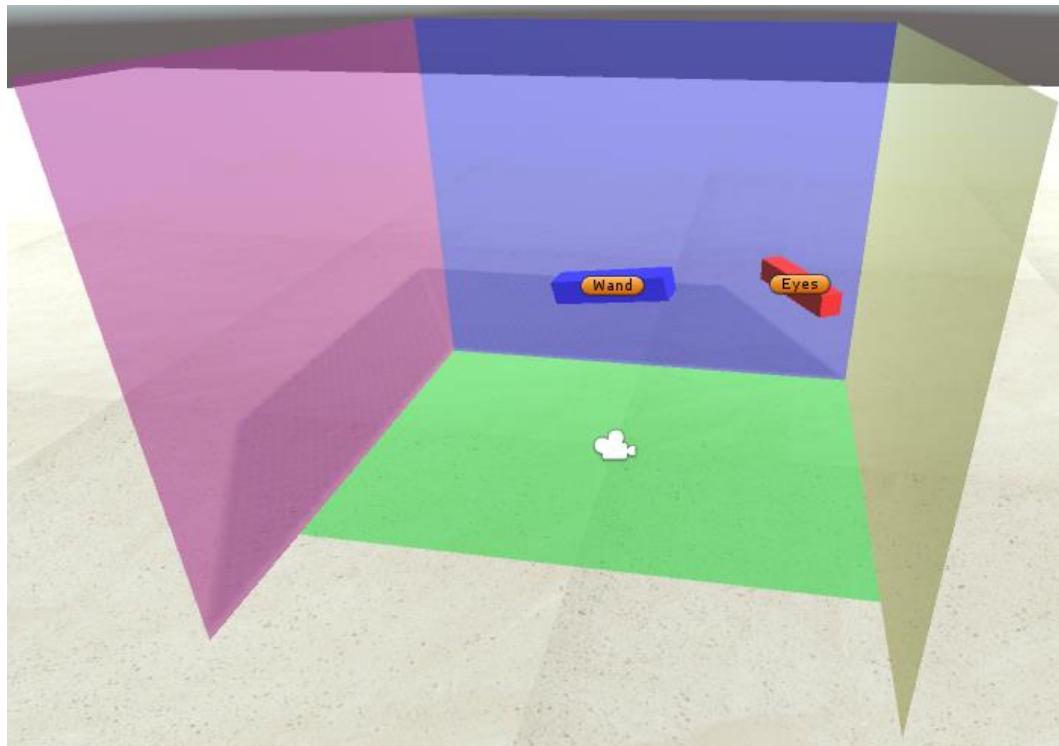


Abbildung 25: Virtueller Wand und Eyes

- **Position und Rotation**

Die realen Positions- und Rotationsveränderungen werden direkt über VRPN auf die Objekte abgebildet und ermöglichen somit ein einfaches Auslesen dieser Daten über das API.

```
var wandRotation = API.Instance.Wand.transform.rotation;
```

Quellcode 2: Zugriff über das API

Das Plugin lässt jedoch zu, einzelne Achsen bei der Positions- und Rotationsfestlegung auszuschliessen. Hierzu wird die Position, bzw. Rotation, vor der Berechnung des neuen Frames zwischengespeichert und auf die blockierten Achsen auf den ursprünglichen Wert zurückgesetzt.

Statt eine Achse komplett zu blockieren, lässt sich auch die Sensibilität adaptieren. Der Standardwert ist 1, was bedeutet, dass im realen Cave eine Verschiebung von einem Meter genau einem Meter in der virtuellen Welt entspricht. Wird die Sensibilität jedoch unter 1 gesetzt, bewegt sich der Wand langsamer und es wird eine kleinere Bewegung virtuell ausgeführt.

- **Mouse Cursor**

Um möglichst generisch die Steuerung der Applikationen übernehmen zu können, ist es unerlässlich, die Mausposition mittels Wand setzen zu können, weil das häufig das primäre Inputgerät ist. Wird auf dieser Ebene des Betriebssystems bereits Hand angelegt, entfallen spezifische Mappings auf Applikationslevel um die Steuerung übernehmen zu können.

Dazu wird ermittelt, wohin der Wand zielt. Die Verlängerung der x-Achse, also eine Gerade definiert durch die Rotation des Wands, kann einen Schnittpunkt mit einer Leinwand des CAVEs haben.

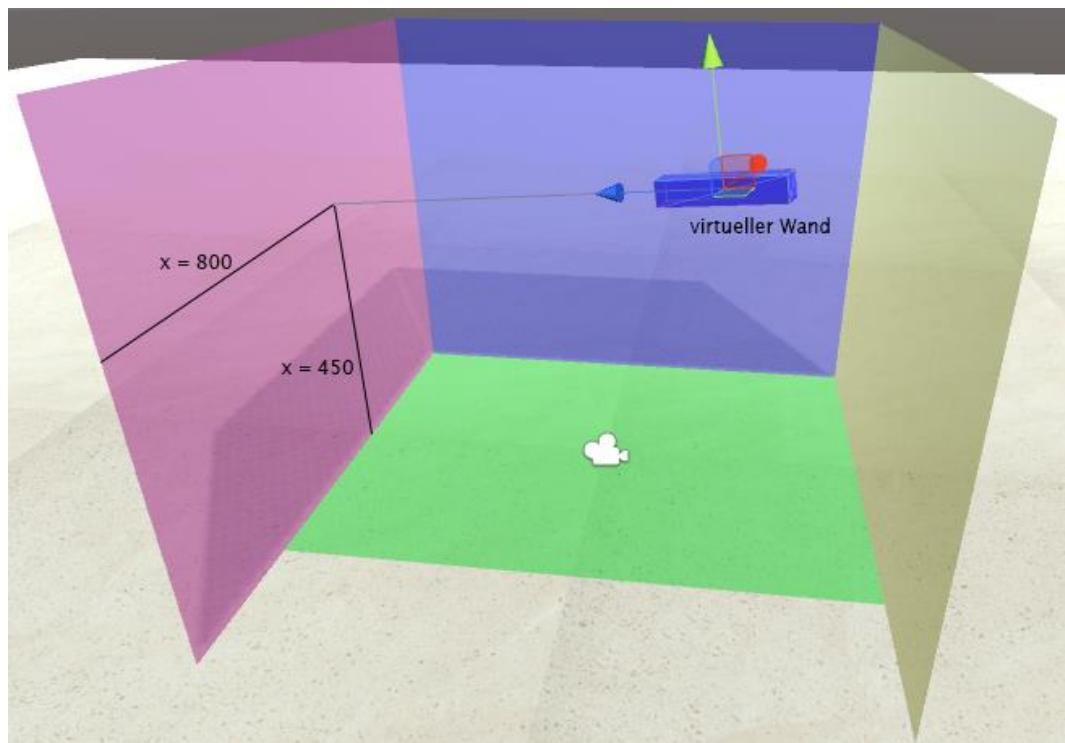


Abbildung 26: Unity Plugin, Schnittpunkt Wand / CAVE

Der Schnittpunkt wird von Unity mit einem Raycast ermittelt.

```
var fwd = transform.TransformDirection(Vector3.forward);

Ray ray = new Ray(transform.position, fwd);
RaycastHit hit;

if (Physics.Raycast(ray, out hit, 100))
{
    Vector3 localSpaceHitPoint = hit.transform.worldToLocalMatrix.MultiplyPoint(hit.point);

    ...
}
```

Quellcode 3: Raycast

Ausgehend vom virtuellen Wand wird ein sogenannter Ray geschossen, welcher wahlweise nach der ersten Kollision abbricht und das getroffene Hit-Objekt zurückgibt oder durch sämtliche Colliders weiterfliegt und alle Ergebnisse als Rückgabewert liefert.

Der exakte Schnittpunkt auf der getroffenen Fläche liefert nun die benötigten Informationen, um den Mouse Cursor auf Betriebssystemebene festzulegen. Zunächst muss aber noch unterschieden werden, welche Leinwand betroffen ist. Das Mapping funktioniert so, dass bei einem Auftreffen auf die linke Leinwand der Cursor im ersten, oberen Achtel des Bildschirms festgelegt wird und der Cursor auf den zweiten, oberen Achtel mit Hilfe des GUI-Systems dupliziert wird, damit bei der stereoskopischen Projektion beide Augen den Cursor sehen. Bei der Front-Leinwand dasselbe mit dem 3. und 4. Achtel. Findet der Schnittpunkt auf der rechten oder unteren Leinwand statt, wird der Cursor in der unteren Hälfte des Betriebssystems platziert. Weitere Informationen zur Aufteilung der Viewports auf dem Unity-Server mittels Unity Plugin werden im Kapitel „Unity Plugin“ behandelt.

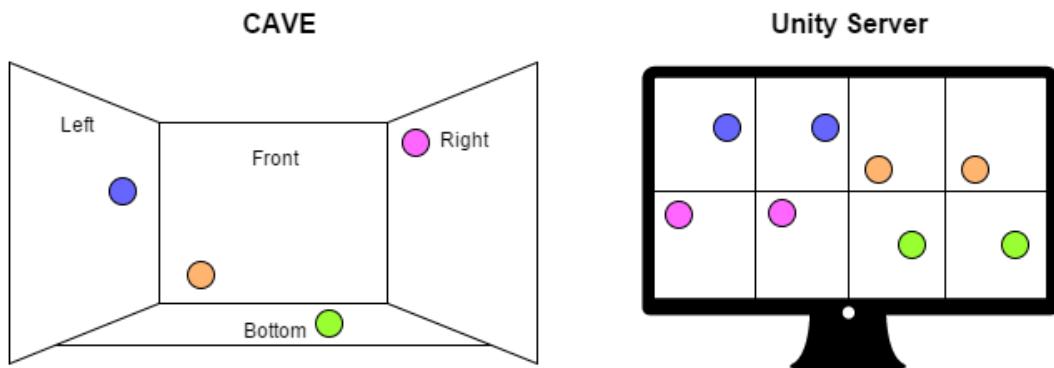


Abbildung 27: Zuordnung Schnittpunkt CAVE und Unity Server

Auf dem Unity Server ist somit jeweils der erste Punkt die reale Position des Cursors und der zweite, gleichfarbige Punkt eine Duplikation für das rechte Auge. Der reale Cursor wird jedoch nie dargestellt, weil die Kopie, welche mittels GUI-System dargestellt wird, immer einen zeitlichen Versatz aufweist und für den Benutzer ein Störfaktor darstellt. Aus diesem Grund wird der Cursor zwar platziert, damit sämtliche über den Cursor laufenden Inputs nach wie vor funktionieren, jedoch ausgeblendet und angezeigt wird eine Cursor-Grafik.

- **Buttons**

Der Wand verfügt über verschiedene Inputs. Nebst dem analogen Joystick sind ein Button auf der Rückseite und vier weitere Buttons auf der Vorderseite angebracht. Zusätzlich lässt sich der Joystick nach unten drücken.

Dem Benutzer soll die Freiheit haben zu entscheiden, welche Aktionen beim Drücken dieser Buttons ausgeführt werden und damit wiederum eine möglichst grosse Bandbreite von Applikationen abgedeckt werden können, ist für jeder Wand-Input eine Keyboard- oder Mauseingabe auswählbar.

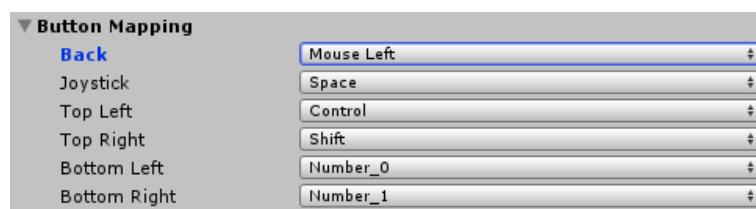


Abbildung 28: Wand Button Zuordnung

Die Auswahlmöglichkeiten beschränken sich auf eine erstellte Liste von Enums, die direkt an einen virtuellen Keycode der InputSimulator²³-Bibliothek geknüpft sind. Wird nun also beim Wand ein Button betätigt, wird diese Information über das VRPN-Protokoll ans Unity Plugin geliefert und mittels dem InputSimulator ein Tastendruck simuliert. Der grosse Vorteil hier ist wiederum, dass Unity nicht unterscheiden kann, ob dieser Tastendruck von einer Tastatur kommt oder im Unity Plugin initialisiert wurde. Die Abfrage in der Applikation, ob eine Taste gedrückt wurde, kann also wie gewohnt über die Input-Klasse gemacht werden und braucht keine spezielle, vom Unity Plugin abhängige Implementierung. Das untenstehende Codesnippet zeigt, wie auf herkömmliche Weise in Unity das betätigen der Leertaste abgefragt wird und auch weiterhin mit dem umgesetzten Unity Plugin Gültigkeit hat.

```
if (Input.GetKeyDown(KeyCode.Space))
{
    // Applikationsspezifischer Code beim Betätigen der Leertaste
}
```

Quellcode 4: Unity Tastaturabfrage

Eine Ausnahme bilden die Mauscicks. Aus Sicherheitsgründen ist es nicht gestattet, mittels InputSimulator Maus-Inputs zu simulieren. Deshalb musste die user32.dll²⁴ zugezogen werden, welche erlaubt, Maus-Events zu senden. Dies geschieht wiederum auf Betriebssystem-Level und hat entsprechend keinen Einfluss auf die Interpretation in der Unity Applikation.

Weil, basierend auf der Unity-Architektur, die Button-Abfrage bei jedem Frame geschieht, würden die Inputs einmal pro gerendertem Frame simuliert werden. Das hätte zur Folge, dass selbst bei einem kurzen Klick von weniger als einer Sekunde, der Input mehrfach ausgeführt werden würde. Deshalb werden beim Auslösen desselben Buttons mittels Coroutinen mehrfache Ausführungen blockiert.

- **Joystick**

Weiter verfügt der Wand über einen analogen Joystick, welcher sich stufenlos auf zwei Achsen bewegen kann. Über die API des Unity Plugins lassen sich diese Werte einfach mittels Delegates auslesen. Es muss lediglich eine Funktion definiert werden, welche bei einem Joystick-Update aufgerufen wird. Der folgende Beispielcode zeigt, wie ein Objekt basierend auf den Joystick-Werten bewegt wird.

²³ Inputsimulator – <http://inputsimulator.codeplex.com/>

²⁴ User32.dll – https://en.wikipedia.org/wiki/Microsoft_Windows_library_files

```

void Start()
{
    // Register
    API.Instance.Wand.OnJoystickAnalogUpdate += OnJoystickUpdate;
}

private void OnJoystickUpdate(float x, float y)
{
    Vector3 posNew = transform.position;
    posNew.x += x / 10f;
    posNew.z += y / 10f;

    transform.position = posNew;
}

```

Quellcode 5: Joystick Handling

Die Joystick-Werte sind Teil der VRPN-Daten, welche der Trackingserver über das PPT Studio 2013 liefert.

4.3.2 Eyes

Die Eyes von WorldViz sind notwendig für das stereoskopische Sehen und die Positions- sowie Rotationsbestimmung des Anwenders.

- **Position und Rotation**

Mit Hilfe zweier Infrarot-Trackern werden die Position im Raum und die Rotation auf zwei Achsen (Yaw und Roll) im PPT Studio 2013 aufbereitet und über VRPN an das Unity Plugin übermittelt. Eine Erfassung der dritten Rotationsachse (Pitch) kann ohne weiteren Tracker nicht erfolgen. Im Gegensatz zum Wand verfügen die Eyes auch nicht über ein Gyrometer.

Im CAVE-Prefab des Plugins sind die Eyes ebenfalls als Objekt in der Hierarchie und übernehmen die Werte vom Trackingsystem. Über das API ist somit ein einfacher Zugriff möglich. Die selektive Fixierung und Adaption der Sensibilität funktionieren analog dem Wand.

4.3.3 Gamepad

Das direkt am Unity Server angeschlossene Gamepad ist nicht konfigurierbar und wird bewusst als Standard-Input belassen, um gängige Applikationen steuern zu können. Der Input-Manager von Unity deckt diese Art von Devices bereits sehr gut ab.



Abbildung 29: Inputmanager Unity

4.4 VRPN

Virtual–Reality Peripheral Network (VRPN) ist eine Klassenbibliothek und ein Server–Interface für Client–Programme und physikalische Geräte, welche in einem VR–System verwendet werden. Das Ziel von VRPN ist es, ein allgemeines Interface für Devices wie Motion–Tracking, Joysticks und weitere Geräte bereitzustellen.

4.4.1 Verwendung mit PPT Studio WorldViz

Das PPT Studio stellt seine Geräte über verschiedene Ausgänge zur Verfügung und verfügt über einen eingebauten VRPN Server. Dieser liefert über verschiedene Trackernamen die gewünschten Informationen.

```

#define Machine Address of PPT machine
hostname = 'localhost'
#define markerID of Wand in PPT
markerID = 3
#create a tracker object for the 6DOF data
tracker = vrpn.addTracker('PPT0@'+ hostname, markerID-1)
#create analog device for the joystick
analogDev = vrpn.addAnalog('PPT_WAND%d@%s:%d' % (markerid, 8945))
#create button device for the buttons
buttonDev = vrpn.addButton('PPT_WAND%d@%s:%d' % (markerid, hostname, 8945))

```

Abbildung 30: Beispielverbindung VRPN, Quelle WorldViz Dokumentation

4.4.2 Datenverarbeitung im Unity

Folgende Wrapperbibliothek wurde verwendet, welche über das offizielle Git-Repository von VRPN erhältlich ist:

- Offizielles Git-Repository
<https://github.com/vrpn/vrpn/wiki>
- UnityWrapper
<https://github.com/arviceblot/unityVRPN>

Dieser Sourcecode wurde in das projekteigene Git-Repository verlinkt, mit CMake²⁵ kompiliert und als DLL ins Asset eingefügt. Im Unity wurde dann ein kurzer Wrapper geschrieben, welche den managed Code der Library zur Verfügung stellt:

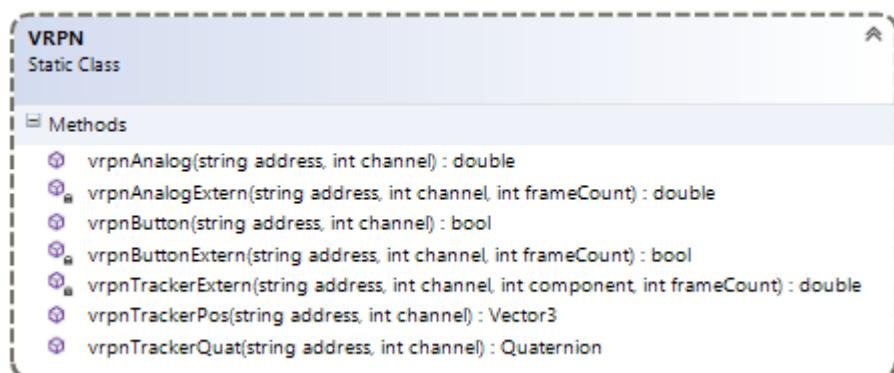


Abbildung 31: VRPN Wrapper C#

²⁵ CMake – <https://cmake.org/>

Die Weiterverarbeitung dieser Trackerdaten übernehmen dann die Klassen der Objekte, namentlich Wand und Eyes. Im Updatezyklus werden diese Daten ausgelesen, verarbeitet (Anwendung eines Smoothing-Filters) und zugewiesen (Rotation und Position).

```

private void HandlePosition()
{
    if (API.Instance.Cave.WandSettings.TrackPosition)
    {
        // Position
        var posOri = transform.localPosition;
        var pos = VRPN.vrpnTrackerPos(
            API.Instance.Cave.WandSettings.WorldVizObject + "@" +
            API.Instance.Cave.Host, API.Instance.Cave.WandSettings.Channel);

        if (_usePositionSmoothing)
        {
            Vector3 filteredPos = Vector3.zero;
            Vector3 filteredVelocity = Vector3.zero;
            OneEuroFilter.ApplyOneEuroFilter(pos, Vector3.zero, posOri,
                Vector3.zero, ref filteredPos, ref filteredVelocity,
                _posJitterReduction, _posLagReduction);
            pos = filteredPos;
        }

        // Block Axis
        if (API.Instance.Cave.WandSettings.PositionAxisConstraints.X)
            pos.x = posOri.x;
        if (API.Instance.Cave.WandSettings.PositionAxisConstraints.Y)
            pos.x = posOri.z;
        if (API.Instance.Cave.WandSettings.PositionAxisConstraints.Z)
            pos.x = posOri.z;

        transform.localPosition = pos;
    }
}

```

Quellcode 6: VRPN-Positionshandling

4.4.3 Datenveredlung im Unity

Da unter Umständen ein etwas unruhiger Input über das VRPN geliefert wird, wurde ein Lowpass²⁶ Filter hinzugefügt. Dieser kann aktiviert, deaktiviert und mit zwei Parametern eingestellt werden. Der „1€ Filter“, welcher zum Einsatz kommt, beschreibt sich wie folgt:

The 1€ filter (“one Euro filter”) is a simple algorithm to filter noisy signals for high precision and responsiveness. It uses a first order low-pass filter with an adaptive cutoff frequency: at low speeds, a low cutoff stabilizes the signal by reducing jitter, but as speed increases, the cutoff is increased to reduce lag. The algorithm is easy to implement, uses very few resources, and with two easily understood parameters, it is easy to tune. In a comparison with other filters, the 1e filter has less lag using a reference amount of jitter reduction. (Géry Casiez, 2012)

²⁶ Lowpass Filter – https://en.wikipedia.org/wiki/Low-pass_filter

Der 1€–Filter ist nur auf 2D ausgelegt, wurde jedoch um eine Dimension erweitert. Sowohl bei tiefen wie auch bei hohen Frequenzen und schnellen Bewegungen wurden optimale Resultate erzielt. Aufgrund dieser guten Resultate kam dieser Filter schlussendlich zum Einsatz.

Folgende Grafik zeigt für die verschiedenen Geschwindigkeitsintervalle die durchschnittliche Distanz zwischen dem gefilterten und der aktuellen Cursorposition. (Géry Casiez, 2012)

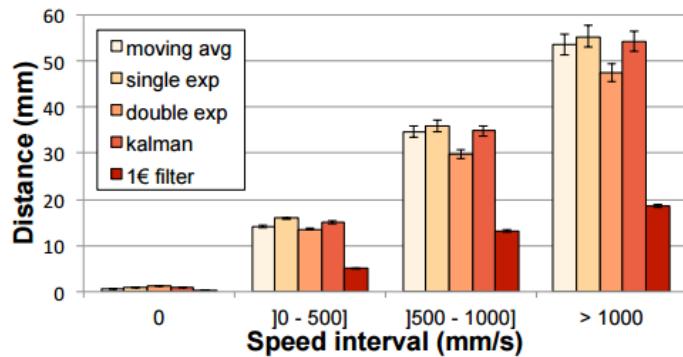


Abbildung 32: 1€ Smoothing Vergleich

4.5 Unity Plugin

Das resultierende Produkt ist ein Unity Plugin, welches sich in wenigen Augenblicken in die eigene Applikation integrieren lässt.

4.5.1 Konfiguration

Um eine möglichst weite Bandbreite von Unity Applikationen abdecken zu können, werden etliche Einstellungsmöglichkeiten zur Verfügung gestellt. Diese gliedern sich in fünf relevante Sektionen.

- **Wand**

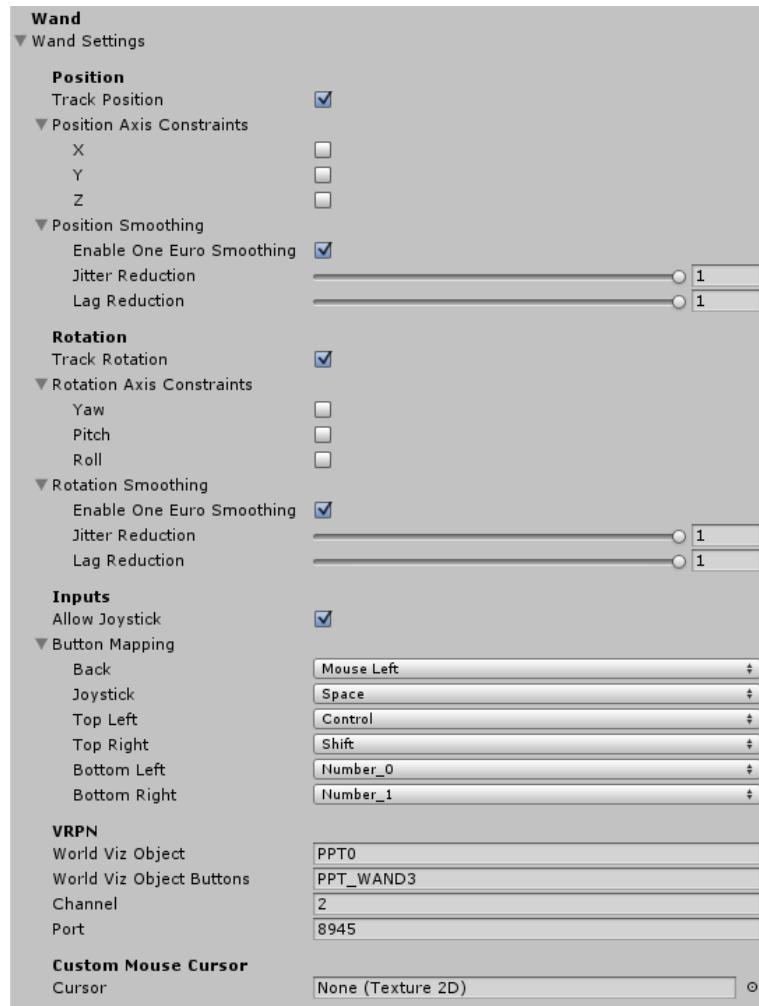


Abbildung 33: Unity Plugin, Settings Wand

Position

Falls die Position des Wands in der aktuellen Applikation unerheblich ist, kann die an dieser Stelle deaktiviert werden. Somit übernimmt der virtuelle Wand, welcher im Prefab liegt, keine Translationen vom realen Wand. Ist diese Option aber aktiviert, besteht die Möglichkeit, achsenabhängig die Sensibilität einzustellen. Das heisst, bei einer hohen Sensibilität auf der y-Achse vollführt der virtuelle Wand eine grosse Bewegung im Vergleich zu der realen Bewegung im CAVE. Gegenteilig, wird der Regler unter 1 gestellt, ist die virtuelle Bewegung kleiner als die reale Bewegung. Zusätzlich können einzelne Achsen auch komplett deaktiviert werden.

Weiter kann auf Wunsch die Interpolation (Smoothing) deaktiviert werden, es wird aber empfohlen, diese Option aktiv zu halten.

Rotation

Auch die Rotation kann auf Wunsch komplett deaktiviert werden. Ebenfalls ein Smoothing ist standardmäßig aktiv.

Inputs

Der Wand verfügt über mehrere Buttons, die aus einer umfassenden Auswahlliste auf die Applikation abgebildet werden können. Zusätzlich kann der Joystick aktiviert oder deaktiviert werden.

Custom Mouse Cursor

Falls im Spiel ein spezifischer Cursor Verwendung findet, kann die Textur hier angegeben werden, damit sie für beide Augen gerendert wird. Ansonsten wird der normale Windows-Cursor angezeigt.

VRPN

Der VRPN-Block wird für die Anmeldeinformationen beim VRPN-Server verwendet. Diese müssen nur bei einer Umstellung des PPT-Studios (auf dem Tracking-Server) adaptiert werden.

- **Eyes**

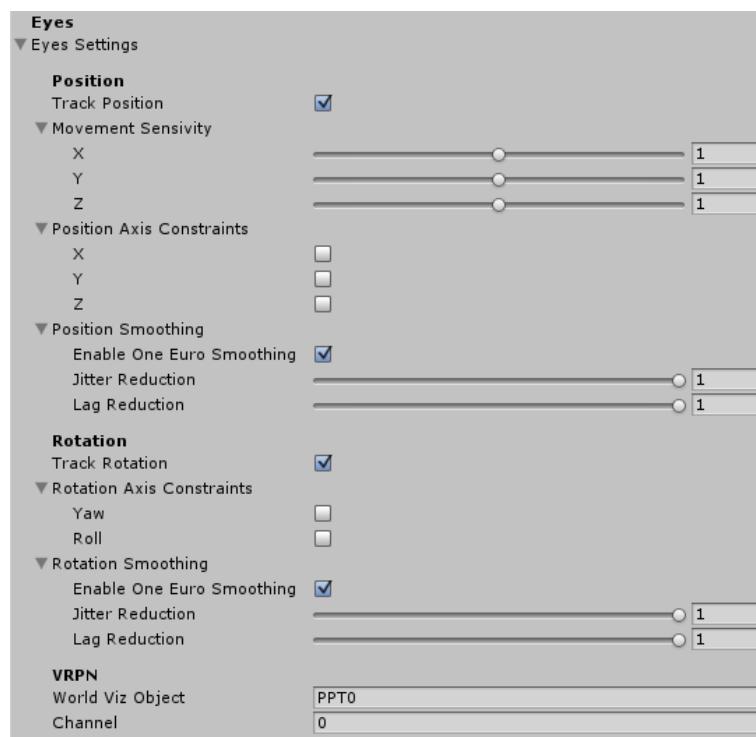


Abbildung 34: Unity Plugin, Settings Eyes

Position

Falls die Position der Eyes in der aktuellen Applikation unerheblich ist, kann die an dieser Stelle deaktiviert werden. Somit übernehmen die virtuellen Eyes, welche im Prefab liegen, keine Translationen von den realen Eyes. Ist diese Option aber aktiviert, besteht die Möglichkeit, achsenabhängig die Sensibilität einzustellen. Das heisst, bei einer hohen Sensibilität auf der y-Achse vollführen die virtuellen Eyes eine grosse Bewegung im Vergleich zu der realen Bewegung im CAVE. Gegenteilig, wird der Regler unter 1 gestellt, ist die virtuelle Bewegung kleiner als die reale Bewegung. Zusätzlich können einzelne Achsen auch komplett deaktiviert werden.

Weiter kann auf Wunsch die Interpolation (Smoothing) deaktiviert werden, es wird aber empfohlen, diese Option aktiv zu halten.

Rotation

Auch die Rotation kann auf Wunsch komplett deaktiviert werden. Ebenfalls ein Smoothing ist standardmäßig aktiv.

VRPN

Der VRPN-Block wird für die Anmeldeinformationen beim VRPN-Server verwendet. Diese müssen nur bei einer Umstellung des PPT-Studios (auf dem Tracking-Server) adaptiert werden.

- **Sekundäre Kameras**

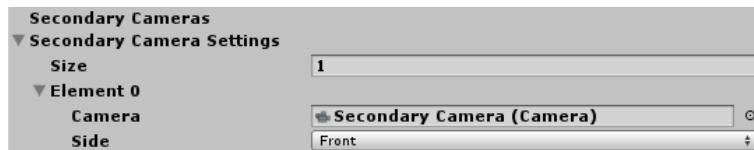


Abbildung 35: Unity Plugin, Settings Sekundäre Kameras

Möglicherweise werden für die Applikation nicht nur eine Hauptkamera, sondern auch eine oder mehrere sekundäre Kameras parallel gerendert. Eine beliebige Anzahl an Kameras können hier angegeben werden und auf welcher Seite der Leinwand die Kamera erscheinen soll.

- **Cave**

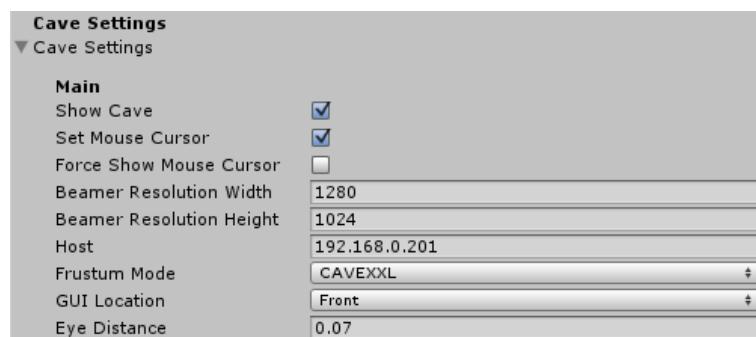


Abbildung 36: Unity Plugin, Settings Cave

In dieser Sektion muss in den meisten Fällen keine Einstellungen vorgenommen werden. Möglicherweise will der Benutzer aber die GUI-Elemente auf einer anderen Leinwand darstellen.

- **System**

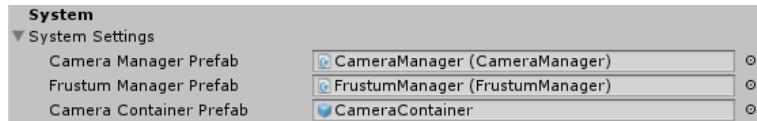


Abbildung 37: Unity Plugin, Settings System

Die zugewiesenen Prefabs werden beim Initialisieren des Plugins instanziert und müssen nicht adaptiert werden.

Das CameraContainer Gameobject beinhaltet alle Kameras, die sich jeweils der Hauptkamera unterordnen und die Bildaufteilung für die verschiedenen Beamer übernimmt.

Einstellungen müssen nur bei grundlegenden Veränderungen gemacht werden und empfehlen sich nur für erfahrene Benutzer.

All diese Einstellungen sind ebenfalls über das bereitgestellte API manipulierbar. Folgender Beispielcode zeigt eine einfache Einstellungsänderung in den Wand Settings.

```
API.Instance.Cave.WandSettings.TrackPosition = false;
```

Quellcode 7: Einstellungsänderung über API

4.5.2 Aufgabenverteilung

Der zentrale Knoten des Plugins ist die Klasse „CaveMain“. Hier sind alle Einstellungsmöglichkeiten, die Referenzen auf sämtliche Objekte und die Geometrie der virtuellen CAVEs gespeichert.

Ein direkter Zugriff auf verschiedene Komponenten sollte grundsätzlich nicht erfolgen, dafür wird ein API als Schnittstelle für Unity Applikationsprogrammierer zur Verfügung gestellt.

Das untenstehende Klassendiagramm zeigt sämtliche Klassen und deren wichtigsten Assoziationen auf.

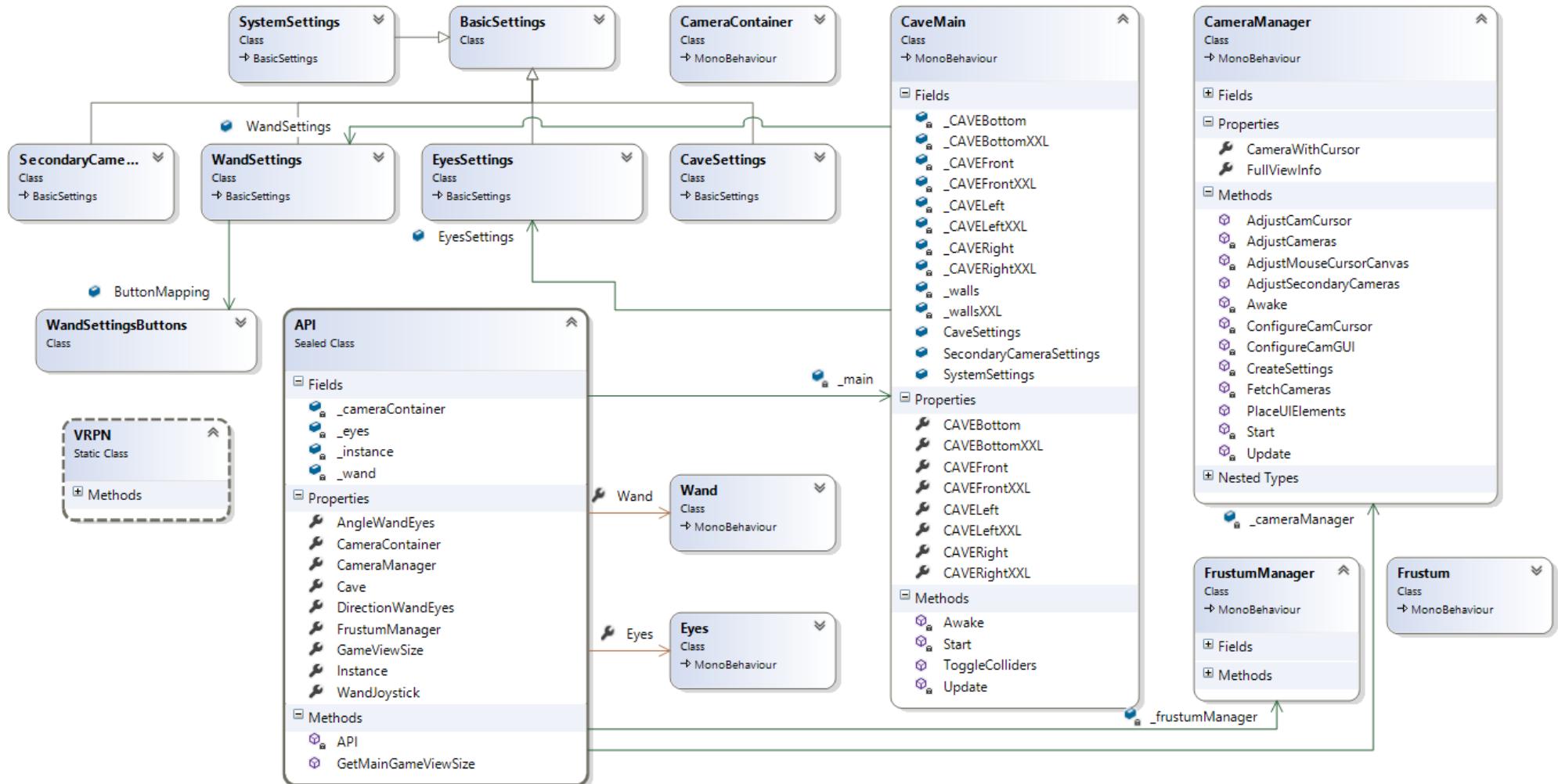


Abbildung 38: Unity Plugin, Klassendiagramm

Die wichtigsten Klassen und deren Methoden werden folgend kurz beschrieben.

- **CaveMain.cs**

Das ist der zentrale Knoten des Plugins. In der Unity-Hierarchie sind sämtliche Plugin-relevanten Objekte Child-Elemente dieser Instanz. Wichtige Management-Klassen werden während des Startvorgangs vom CaveMain instanziert und die Geometrie des virtuellen CAVEs wird hier zusammengetragen. Der Zugriff auf die benutzerspezifischen Einstellungen, welche in Model-Klassen ausgelagert sind, erfolgt über diese Stelle. Je nach Einstellung, ob der CAVE zu Debug-Zwecken dargestellt werden soll, deaktiviert diese Klasse sämtliche Renderers.

Methode	Aufgabe
Awake()	Instanziert CameraManager, FrustumManager und CameraContainer
Start()	<ul style="list-style-type: none"> • Erstellt eine Kollektion der Cave-Transforms • Deaktiviert auf Wunsch sämtliche Renderer • Deaktiviert die Collider des virtuellen Caves
Update()	Übernimmt die Position sowie Rotation der Hauptkamera.
ToggleColliders()	Schaltet die Collider der CAVE-Wände ein, aus.

Tabelle 2: CaveMain Methoden

Enum	Aufgabe
FrustumMode	Auswahl für CAVE XXL und GPP_Kooima. Steuert, wie das Frustum berechnet wird.

Tabelle 3: CaveMain Enums

- **Eyes.cs**

Die virtuellen Eyes interpretieren die Daten des Trackingservers über VRPN und beachten die benutzerspezifischen Einstellungen des Plugins. Nach dieser Bearbeitung wird das in der Hierarchie liegende Eyes-Objekte aktualisiert, damit mittels API dessen Werte ausgelesen werden können.

Methode	Aufgabe
Start()	Speichert etliche über das API erreichbare Daten in privaten Variablen.
Update()	Führt in jedem Frame die Methoden „HandlePosition“ und „HandleRotation“ aus.
HandlePosition()	<ul style="list-style-type: none"> • Position von VRPN übernehmen • Smoothing durchführen • Blockierte Achsen zurücksetzen • Neue Position setzen
HandleRotation()	<ul style="list-style-type: none"> • Rotation von VRPN übernehmen • Smoothing durchführen • Blockierte Achsen zurücksetzen • Neue Rotation setzen

Tabelle 4: Eyes Methoden

- **Wand.cs**

Der Wand interpretiert analog den Eyes die Daten des Trackingservers über VRPN und aktualisiert unter Berücksichtigung der benutzerspezifischen Einstellungen das Wand-Objekt. Der Wand ist aber noch für etliche weitere Aufgaben zuständig. Alle Inputs, die der Wand liefert, werden ebenfalls an dieser Stelle bearbeitet. Das umfasst das Setzen der Cursor-Position und dessen eventuell vorhandenen Cursor-Textur, die Bereitstellung der Joystick-Daten und die Input-Simulation der verschiedenen Buttons.

Methode	Aufgabe
Start()	Speichert etliche über das API erreichbare Daten in privaten Variablen und setzt, falls vom Benutzer gewünscht, eine spezifische Cursor-Textur.
Update()	Führt in jedem Frame die Methoden „HandlePosition“, „HandleRotation“, „HandleButtons“, „HandleJoystick“ und „SetCursor“ aus.
HandlePosition()	<ul style="list-style-type: none"> • Position von VRPN übernehmen • Smoothing durchführen • Blockierte Achsen zurücksetzen • Neue Position setzen
HandleRotation()	<ul style="list-style-type: none"> • Rotation von VRPN übernehmen • Smoothing durchführen • Blockierte Achsen zurücksetzen • Neue Rotation setzen
HandleButtons()	<ul style="list-style-type: none"> • Empfängt die Button-Inputs über VRPN • Simuliert die im Inspector zugewiesenen Inputs (Tastatur sowie Maus) • Stellt sicher, dass derselbe Input nicht versehentlich mehrmals hintereinander ausgeführt wird

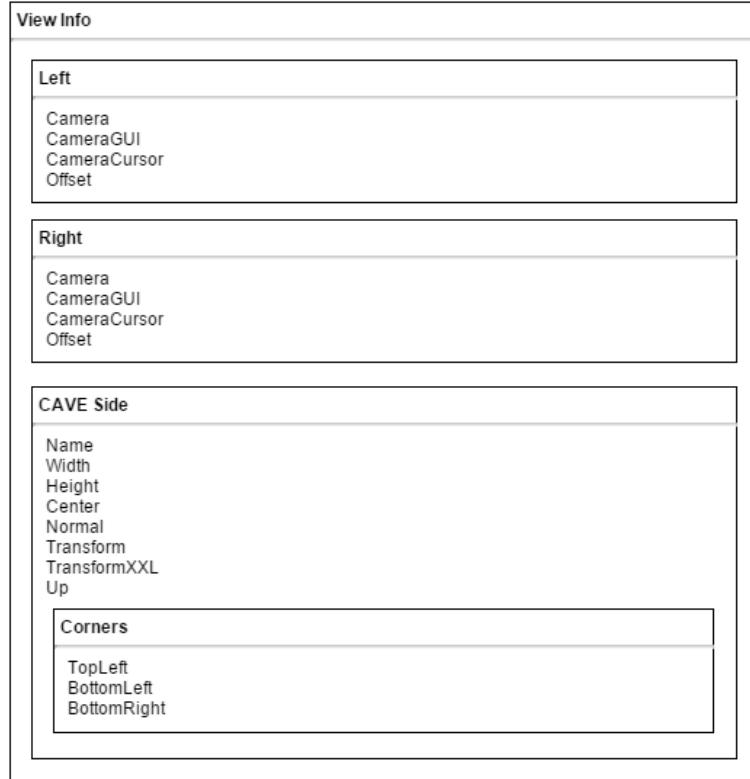
HandleJoystick()	<ul style="list-style-type: none"> • Empfängt die Joystick-Inputs über VRPN • Führt die registrierten Delegates aus mit der aktuellen Joystick-Position
SetCursor()	<ul style="list-style-type: none"> • Raycast auf den virtuellen Cave • Leinwand ermitteln • Cursor-Kamera adaptieren • 2D-Position auf Screen ermitteln • Cursor-Position setzen • Cursor-Position des Duplikats setzen

Tabelle 5: Wand Methoden

- **CameraManager.cs**

Das Aufgabenspektrum des CameraManagers befasst sich mit allen Tasks, die in Verbindung mit einer Kamera stehen. Vorgängig wird hier das grundlegende Setting der multiplen Kamera-Konstruktion für das stereoskopische Sehen vorgenommen und alle relevanten Informationen zu den CAVE-Seiten werden in Structs gespeichert.

Jede Seite des CAVEs, also 4 Stück, enthält Daten über die Kameras je Auge, die Abmessungen der Leinwand, die Eckpunkte der Leinwand usw. Diese Daten finden während dem gesamten Prozess Verwendung, hauptsächlich jedoch bei der Frustumberechnung. Folgend sind die verwendeten Structs.



Quellcode 8: CameraManager DataStruct

Methode	Aufgabe
Awake()	Setzt CaveMain als Parent.
Start()	Verhindert das Rendern von UI-Elementen auf der Hauptkamera und führt die Methoden „FetchCameras“, „CreateSettings“, „AdjustCameras“, „AdjustSecondaryCameras“ und „PlaceUIElements“ aus.
Update()	Ermittelt, auf welcher Kamera sich der Systemcursor befindet.
FetchCameras()	Speichert die Referenzen auf die verschiedenen Kameras.
CreateSettings()	Füllt die oben abgebildeten Structs mit Daten ab und speichert sie in einem Dictionary.
AdjustCameras()	Übernimmt die Grundeinstellungen der Hauptkamera, macht spezifische Einstellungen je Seite und positioniert die Viewports.
AdjustSecondaryCameras()	Konfiguriert die vorhandenen sekundären Kameras.
PlaceUIElements()	Konfiguriert die GUI-Kameras und platziert die Canvas.
AdjustCamCursor()	Die für den Cursor zuständigen Kameras werden je nach Cursorposition ausgerichtet und ein- / ausgeschaltet.

Tabelle 6: CameraManager Methoden

Enum	Aufgabe
CameraDepths	Steuert, auf welchem Layer sich die Kamera befindet. Sodass beispielsweise das GUI nicht durch das Spiel verdeckt wird.

Tabelle 7: CameraManager Enums

- **CameraContainer.cs**

Dieser Container beinhaltet alle dynamisch, vom Unity Plugin generierten Kameras, um sie zentral verschieben zu können. Zur Laufzeit wird geprüft, ob eine andere Kamera als Hauptkamera definiert wurde und verschiebt sich entsprechend dorthin.

Die Position CameraContainer-Objects ist direkt an die Position der Eyes gebunden. Und weil die Sensibilität, also in welchem Verhältnis sich die virtuellen Eyes zu den realen Eyes verschieben, eingestellt werden können, muss diese Berechnung hier geschehen. Beim Setzen der Position wird eine einfache Vektormultiplikation durchgeführt.

Methode	Aufgabe
Start()	Die Bewegungssensibilität wird zwischengespeichert.
Update()	<ul style="list-style-type: none"> • Prüfung, ob der Container noch an der Hauptkamera angehängt ist und eventueller Parentwechsel. • Setzen der Position basierend auf den Eyes sowie Sensibilität beachten.

Tabelle 8: CameraContainer Methoden

- **FrustumManager.cs**

Der FrustumManager sammelt die benötigten geometrischen CAVE-Daten und die dazugehörigen Kameras, um mittels Frustum-Klasse das auf die Kameras abzubildende Viewfrustum zu berechnen.

Methode	Aufgabe
Awake()	Setzt CaveMain als parent.
Update()	Nimmt die vom CameraManager zusammengetragenen Daten und bereitet sie für die Frustum-Berechnung auf.

Tabelle 9: FrustumManager Methoden

- **Frustum.cs**

Berechnet für die ihm angegebene Kamera das Viewfrustum, basierend auf der Eyes-Position, den drei Eckpunkten einer CAVE-Seite sowie der Near- und Farplane. Siehe dazu Kapitel 4.2.1

- **VRPN.cs**

Diese Klasse übernimmt die Schnittstelle zwischen C# (Unity) und C++ (VRPN). Alle Daten, die vom Trackingserver aufbereitet und übers VRPN-Protokoll verschickt werden, werden hier empfangen und in C# Datentypen umgewandelt, um die Verwendung zu vereinfachen.

- **API.cs**

Grundsätzlich kann der gesamte Sourcecode des Plugins eingesehen und verändert werden. Um die Verwendung jedoch zu vereinfachen und die eigene Anwendung applikationsspezifisch mit dem Plugin zu verknüpfen, werden gewissen Werte, Berechnungen und Objekte in der API zur Verfügung gestellt. Mittels Singleton-Pattern wird sichergestellt, dass die Verwaltung der besagten Properties zentral an einem Ort geschieht und dort abgegriffen werden können.

Der untenstehende Code der Klasse API zeigt, wie beispielsweise der Wand gesucht, referenziert und zurückgegeben wird. Ebenfalls werden gewisse Berechnungen, in diesem Beispiel der relative Winkel zwischen dem Wand und den Eyes, direkt berechnet und können abgefragt werden.

```

public sealed class API
{
    static readonly API _instance = new API();

    public Wand Wand
    {
        Get
        {
            if(_wand == null)
            {
                return _wand = GameObject.Find("WorldVizWand")
                    .GetComponent<Wand>();
            }

            return _wand;
        }
    }

    ...

    public Quaternion AngleWandEyes { get { return Quaternion
        .Inverse(Eyes.transform.rotation) * Wand.transform.rotation; } }

    public static API Instance
    {
        get
        {
            return _instance;
        }
    }

    ...
}

}

```

Quellcode 9: API

Der Zugriff auf das API erfolgt, weil es sich um ein Singleton-Pattern handelt, über die selber erstellte Instanz.

```

var angle = API.Instance.AngleWandEyes;

```

Quellcode 10: Zugriff über das API

4.5.3 Deployment

Das entwickelte Unity Plugin muss möglichst unkompliziert und rasch in die gewünschte Applikation integriert werden können. Um das zu erreichen, wird der gleiche Ansatz wie das Deployment über den integrierten Unity Asset Store gewählt. Dazu werden sämtliche Verzeichnisse und Dateien in eine .unitypackage-Datei gepackt und können in jedes beliebige Projekt importiert werden.

Um diese Bündelung der Dateien zu erreichen, gibt es in Unity den Befehl, ein Package zu exportieren.

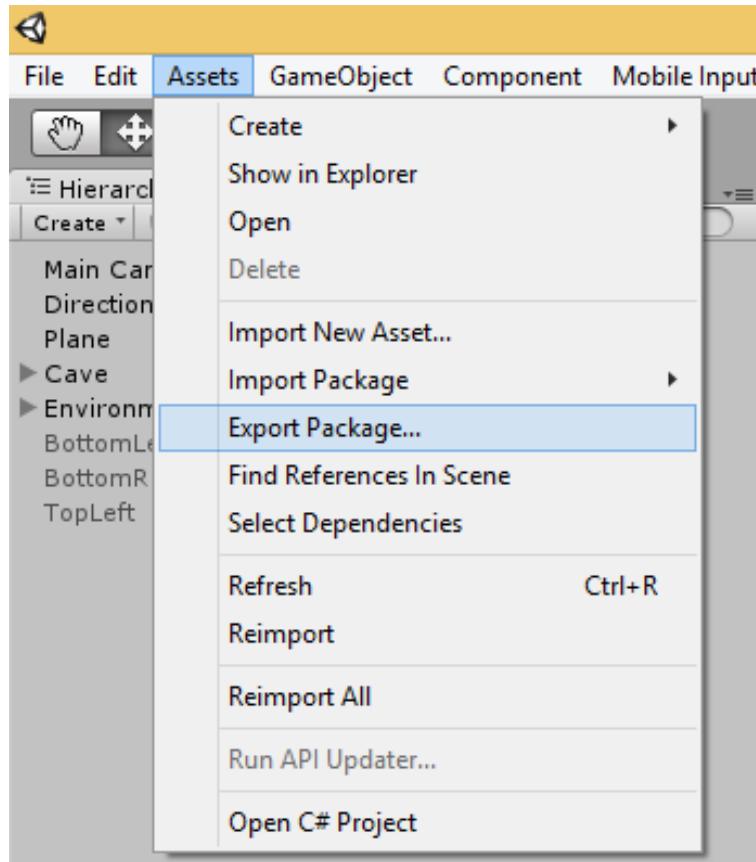


Abbildung 39: Unity Plugin, Export Package

Beim anschliessenden Popupmenu alle Assets auswählen und den Export starten.

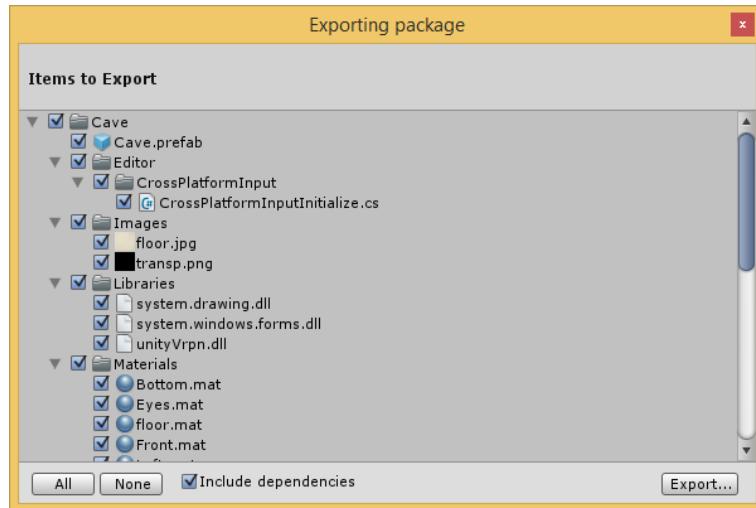


Abbildung 40: Unity Plugin, Export Package Selection

Die Verwendung des Plugins mit sämtlichen Abhängigkeiten erfolgt in wenigen Schritten:

1. Import des Plugins

Abhängig der Unity-Version kann der Import zu Problemen führen. Sind gewisse, bereits existierende Files in Verwendung, werden die Klassen / Prefabs / usw. nicht überschrieben, sondern lediglich hinzugefügt (CaveMain 1.cs, CaveMain 2.cs usw.). Darum empfiehlt sich, den

Import manuell über den Explorer vorzunehmen und nicht die exportierte .unitypackage-Datei zu verwenden.

Als erstes muss das Unity-Projekt geschlossen werden, damit alle Dateien und Verzeichnisse über Schreibrechte verfügen. Falls bereits eine andere Version des Plugins ins Projekt integriert wurde, den Ordner „Cave“ im Assets-Verzeichnis löschen und die neue Version reinkopieren.

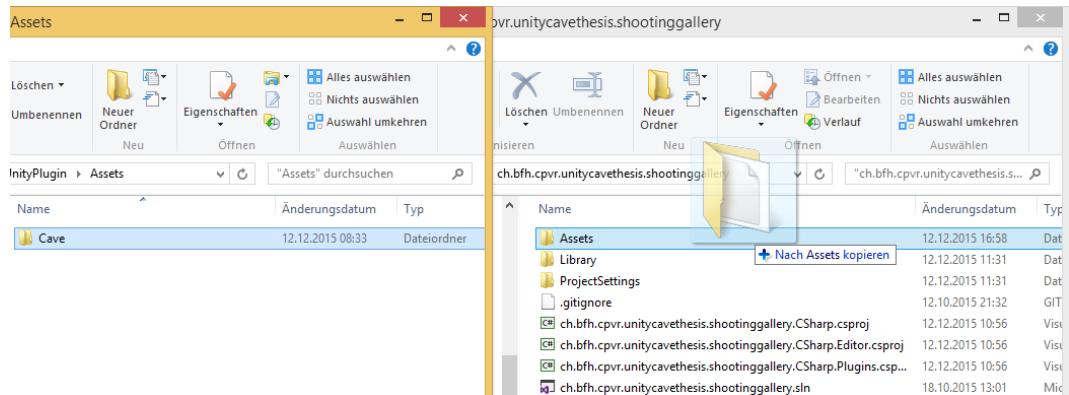


Abbildung 41: Kopieren des Unity Plugins

Alternativ kann der Asset Import-Mechanismus von Unity verwendet werden. Diese Methode, auch wenn sie elegant erscheinen mag und ursprünglich so angedacht war, führt leider oft zu einem Fehlverhalten und es wird an dieser Stelle abgeraten, das Asset auf diese Weise zu importieren.

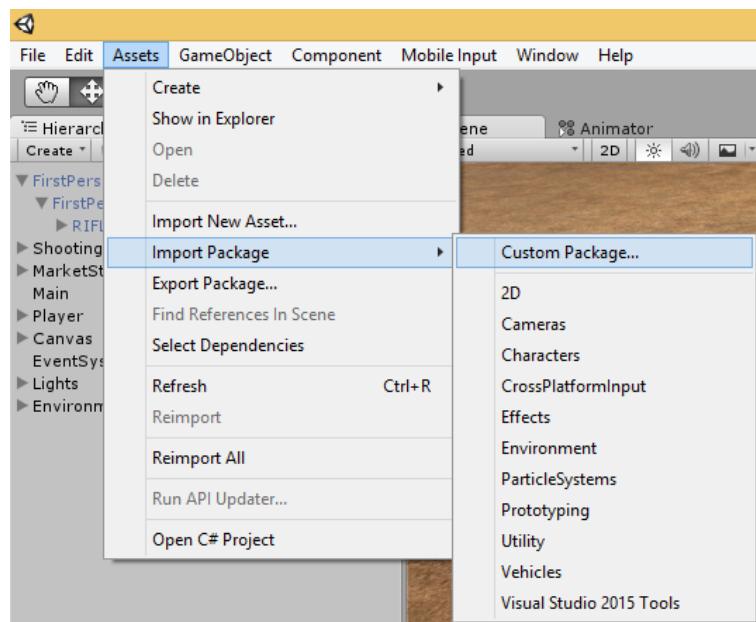


Abbildung 42: Unity Plugin, Import Package

Beim anschliessenden Popupmenu sind alle Assets auszuwählen und mit „Import“ zu bestätigen.

2. Cave-Prefab

In einem zweiten Schritt gilt es, das Cave-Prefab, welches sich direkt im Verzeichnis „Cave“ befindet, an einer beliebigen Stelle in der Szene zu platzieren.

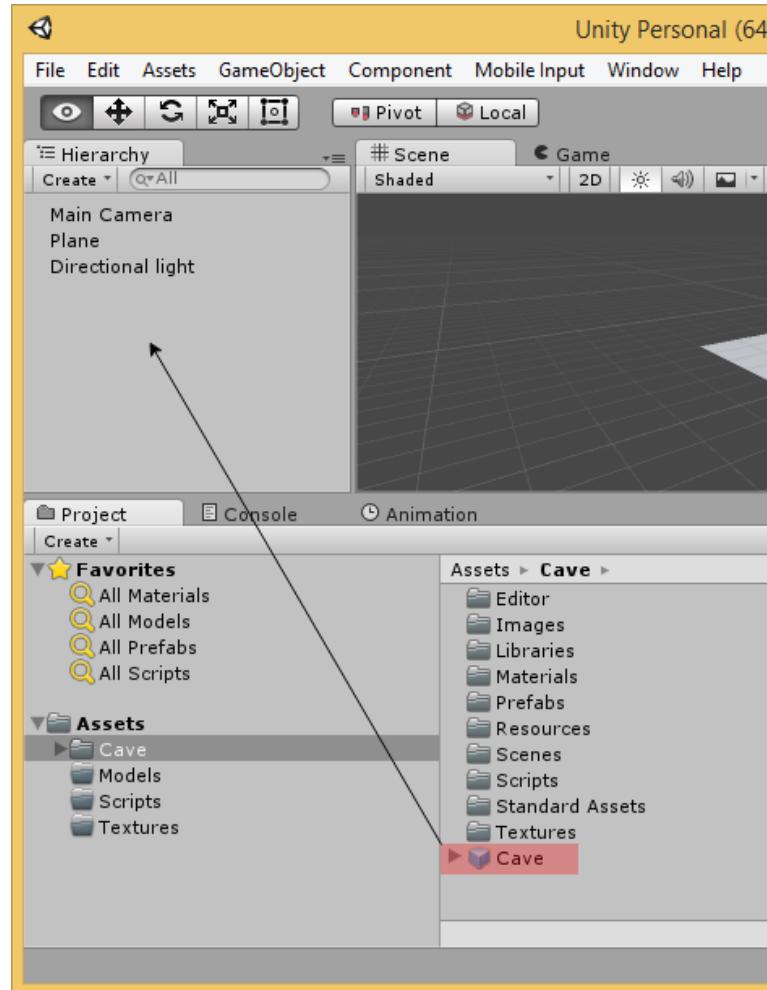


Abbildung 43: Hinzufügen Cave Prefab

3. API Kompatibilitätslevel

Weil das Unity Plugin sich weiterer DLLs bedient, muss bei den Player-Settings das „Api Compatibility Level“ auf „.NET 2.0“ (ohne Subset) gesetzt werden. Damit wird die Verwendung externer DLLs ermöglicht.

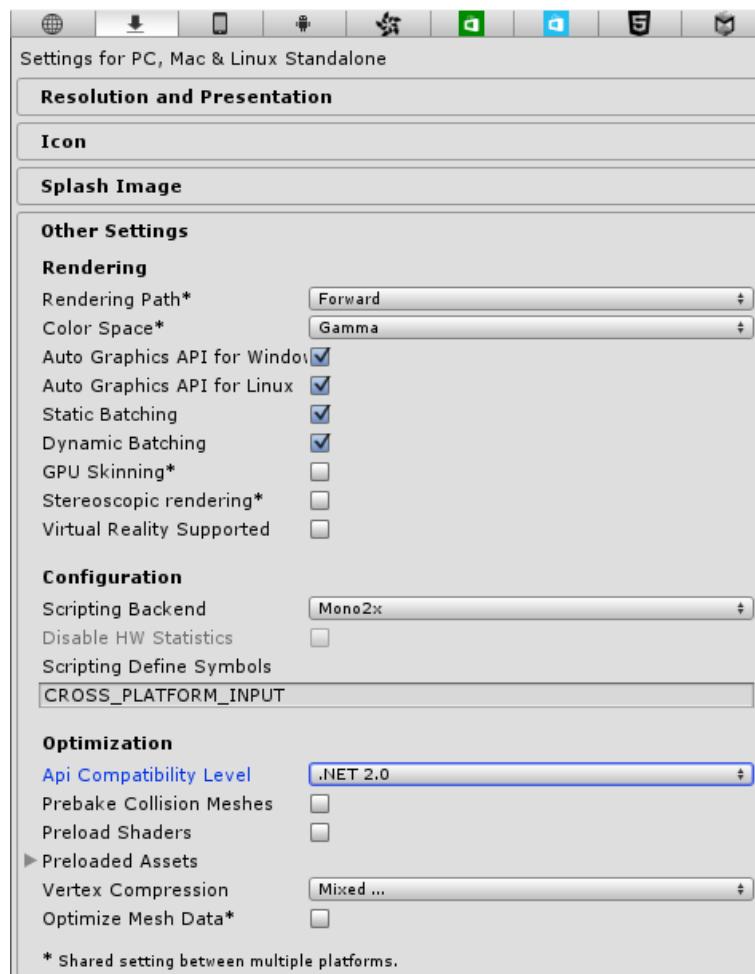


Abbildung 44: Player Settings, Api Compatibility Level

4. Plattform Einstellungen

Um sicherzustellen, dass die zusätzlichen DLLs für die Zielplattform exportiert werden, im Projekt zu den DLLs navigieren (Cave -> Libraries) und bei allen DLLs die entsprechenden Häkchen setzen.

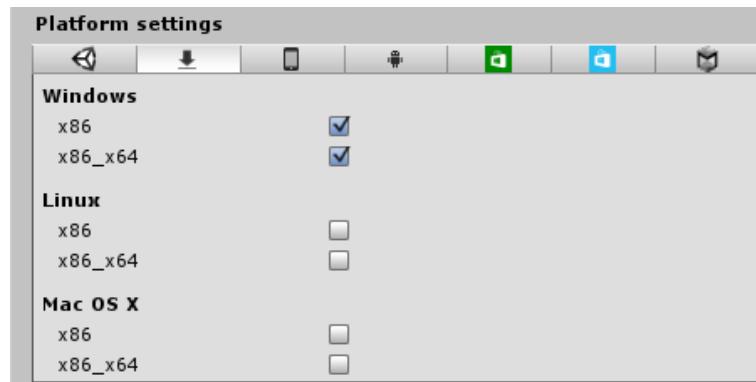


Abbildung 45: Platform Settings

5. Einsatzbereit

Nach diesen einfachen Schritten wurde das Plugin erfolgreich in die Applikation eingebettet und kann verwendet werden.

4.5.4 Troubleshooting

• Fehlende Verlinkung Prefabs

Es besteht die Möglichkeit, dass die Verbindung zu den Prefabs vom CameraManager, FrustumManager und CameraContainer unter den System Settings beim Kopieren verloren geht. In diesem Fall direkt in Unity das entsprechende Prefab aus dem Ordner „Cave / Prefabs“ an das dafür vorgesehene Property hängen.

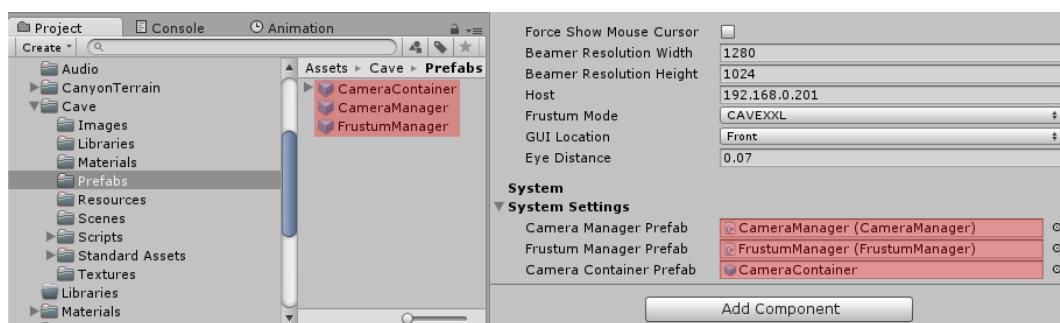


Abbildung 46: Zuweisung Prefabs

- **Falsche Architektur**

Falls beim Exportieren Fehler auftreten sollten, muss die Architektur möglicherweise auf 64bit angepasst werden.

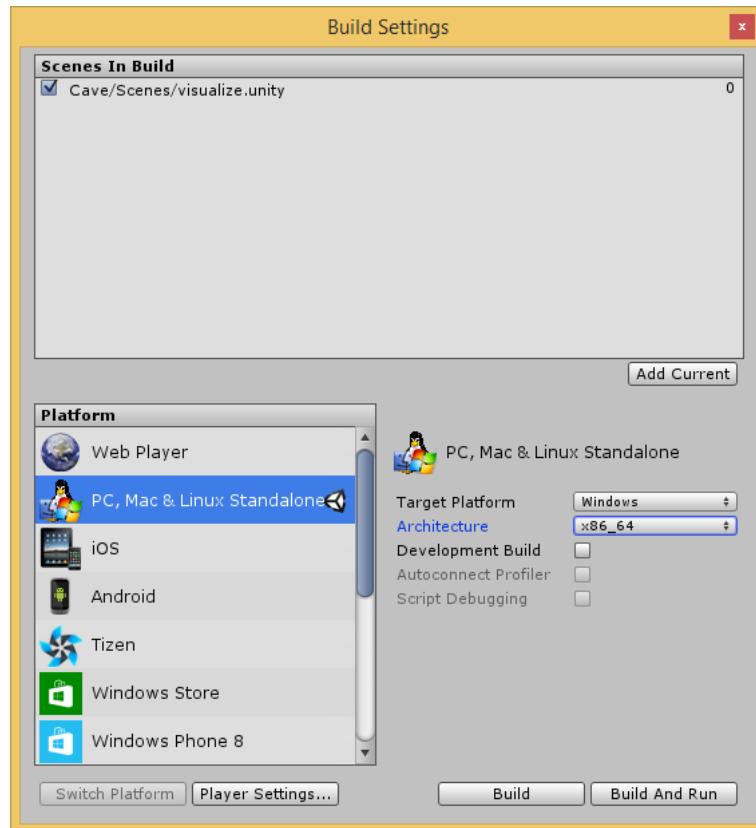


Abbildung 47: Export 64bit

4.5.5 Performance Verbesserungen

Um eine möglichst gute Performance zu erreichen, wurde bereits während der Entwicklung darauf geachtet, Scripts nicht unnötig auszuführen. Dennoch wurde zum Schluss der Code nochmals analysiert und geprüft, wo Verbesserungen durchgeführt werden können.

Im folgenden Profiling-Screenshot ist deutlich zu sehen, dass fast sämtliche Ressourcen nur noch für die Berechnung der Kameras verwendet werden. Alle anderen Aufgaben sind weitgehend optimiert und haben keinen markanten Einfluss auf den Spielfluss. Das relativ kostspielige Rendering aller Kameras lässt sich nicht umgehen und ist ein wichtiger Bestandteil des Plugins.

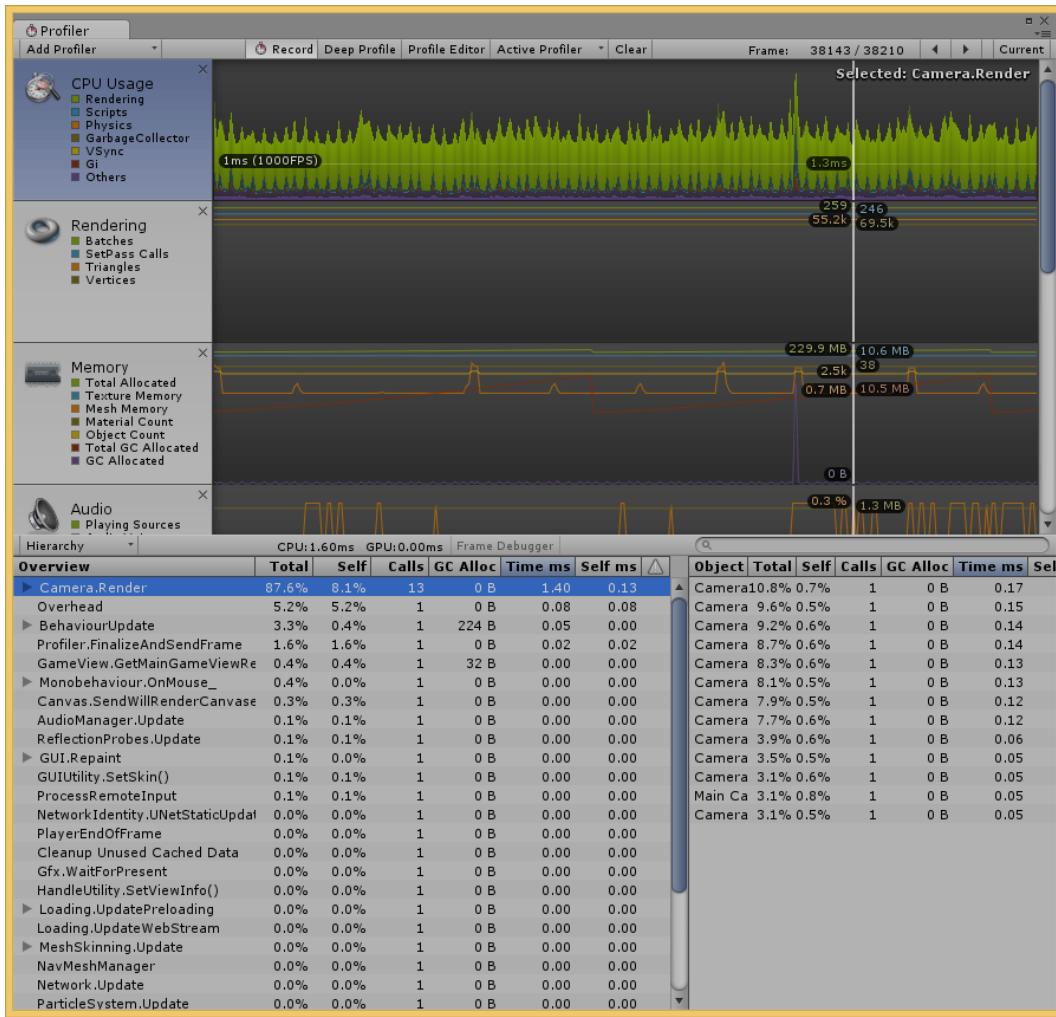


Abbildung 48: Performance analysieren mit Unity Profiler

Die folgenden Codeänderungen wurden im Zuge der Performanceanalyse durchgeführt.

- **Komponenten suchen**

Bei etlichen Modulen bestehen Verbindungen zu anderen Objekten. Es wurde sichergestellt, dass sämtliche Referenzen nicht mehrmals neu geholt werden, sondern alle Verknüpfungen in der Startphase gemacht werden.

- **Deaktivieren Rendering Hauptkamera**

Weil das Plugin eigene Kameras erstellt und die Darstellung der ursprünglichen Kamera obsolet macht, indem die neuen Viewports den alten Viewport verdecken, wird das Rendering dieser Kamera nach der Konfigurationsphase deaktiviert.

- **UI-Kameras orthografische Projektion**

Weil die Berechnung einer orthografischen im Vergleich zu einer perspektivischen Projektion einfacher ist, wird die Projektion für alle UI-Kameras auf orthografisch gesetzt. Dies ist möglich, weil die Perspektive keinen Einfluss auf die Darstellung der 2D-UI-Elemente hat.

- **FixedUpdate²⁷ statt Update**

Unity stellt eine Methode namens FixedUpdate() zur Verfügung, die im Vergleich zu Update() nicht so häufig wie möglich (also bei jedem berechneten Frame) ausgeführt wird, sondern nur zu fix definierten Zeitpunkten. Standardmäßig sind das 30 Ausführungen pro Sekunde. Alle Berechnungen vom Wand, den Eyes und dem Viewfrustum werden in dieser FixedUpdate–Methode gemacht. Der Unterschied ist für den Benutzer nicht spürbar und es lassen sich somit etliche Berechnungen vermeiden.

- **Wand Raycast**

Der Wand führt in gewissen Zeitabständen Raycasts aus, um den Punkt auf dem virtuellen CAVE, in welche er ausgerichtet ist, zu bestimmen. Die Distanz, wie weit dieser Raycast in die virtuelle Welt geschossen werden soll, kann bestimmt werden. Im Falle der CAVE-Schnittpunktsuche ist eine maximale Distanz von den räumlichen Gegebenheiten bereits gegeben und muss nur so weit erfolgen.

- **Clipping Planes**

Wie weit, bzw. nah, sich die Clipping Planes befinden, wird von der Unity Applikation festgelegt und darf vom Plugin nicht beeinflusst werden. Jedoch für die UI-Kameras ist die maximale Distanz nur bis zum CAVE XXL und lässt sich entsprechend verkleinern, um eine bessere Performance zu erreichen.

²⁷ FixedUpdate Unity – <http://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html>

4.6 Warping²⁸

Bei einer stereoskopischen Projektion mit zwei unterschiedlichen Beamern muss für eine vollständige Immersion die Kalibrierung dieser zwei Output Geräte sehr genau sein. Aufgrund verschiedener Umstände (Wärmeausdehnung, Freiheitsgrade durch Gelenke und Spiegel) wäre es wünschenswert, den gerenderten Output noch zu «warpen», um eine bessere Übereinstimmung beider Bilder auf einer Leinwand zu erreichen.

„Warping (von englisch warp = verformen, verzerren) von Bildern gehört in der Computergrafik zu den bildbasierten Techniken. Falls zu einem Bild die dazugehörigen Tiefenwerte existieren, ist es mittels der Warping-Gleichung möglich, das Bild von einem anderen Blickpunkt zu betrachten. Das Verfahren ist echtzeitfähig, bringt jedoch einige Artefakte wie beispielsweise Aufdeck- oder Verdeckungsfehler mit sich.“ (Warping, 2016)

In Unity ist dies durch einen Shader oder über die Projektionsmatrix der Kamera möglich. Da im CAVE die Projektionsmatrix bereits für die Frustumtransformation neu berechnet wird, empfiehlt sich hier die Verwendung des Shaders.

Im folgenden Beispiel ist ein einfacher Shader verwendet worden, um eine homographische Transformation der Kameraperspektive durchzuführen. (chiragraman, 2016)

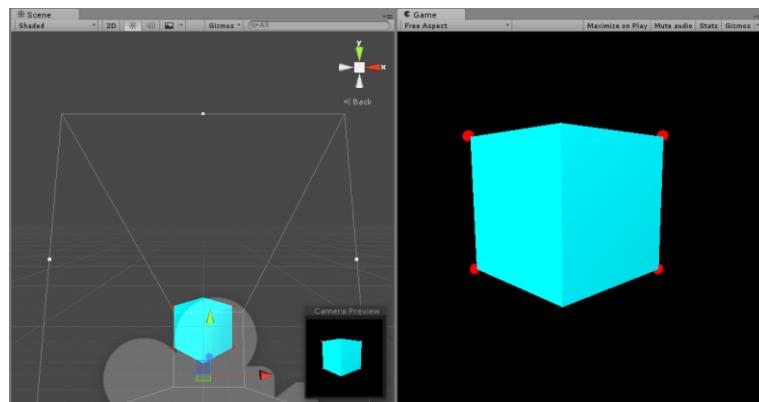


Abbildung 49: Initialposition Warpingbeispiel

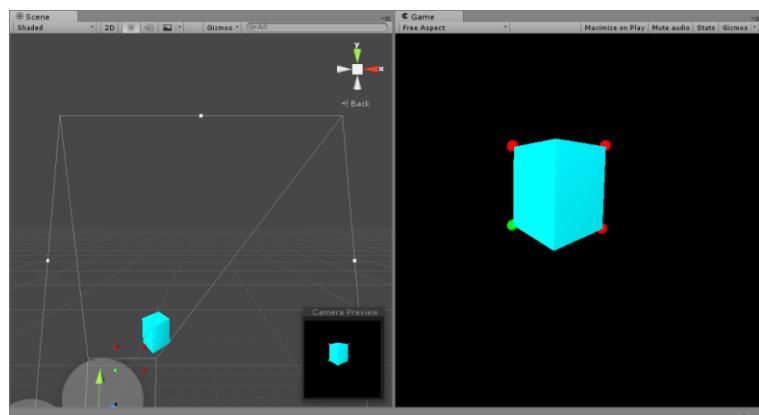


Abbildung 50: Verkleinerung Perspektive Warping

²⁸ Warping – https://en.wikipedia.org/wiki/Image_warping

Fazit

Das Warping ist prinzipiell möglich und es empfiehlt sich, dies über einen Shader durchzuführen. Dadurch wird die Projektionsmatrix der Unity Kamera nicht mehr verändert und der Shader kann für sich selbst behandelt werden. Es ist möglich, bereits vorhandene GLSL²⁹-Shader in Unity zu verwenden. Eine Manipulation der Projektionsmatrix kann dann u.U. ähnliche Verzerrungseffekte hervorrufen wie in Kapitel 4.2.2 erwähnt wird.

Offene Punkte, Abgrenzung

Folgende Arbeiten wurden in diesem Thema durchgeführt:

- Recherche Warping Machbarkeit in Unity
- Prototyp Variante Shader (Fisheye von Unity)
- Prototyp Projektionsmatrix (chiragraman, 2016)

Adaptionen direkt im Plugin und adjustierte Shader wurden aus Zeit- und Komplexitätsgründen weggelassen. Hier empfiehlt es sich, die Verzerrungen zuerst zu erfassen, in einem zweiten Schritt diese durch Vektoren abzubilden um damit einen Shader aufzubauen. Weiter ist nicht bewiesen, ob die Verzerrungen sich während des Betriebes noch durch zusätzliche Temperaturausdehnungen verändern. Falls dies der Fall wäre, müsste der Shader durch eine Nachkonfiguration während des Betriebes verändert werden.

²⁹ GLSL Shader in Unity – <http://docs.unity3d.com/Manual/SL-GLSLShaderPrograms.html>

5 Demo Apps

5.1 Shooting Gallery



Abbildung 51: Shooting Gallery Ingame

Das Setting dieses Demospiels ist eine Schiessbude im Wilder Westen Stil, wie sie auf einem Jahrmarkt anzutreffen ist. Die Galerien mit den abzuschiessenden Zielen verteilen sich rund um den Spieler. Mit Hilfe des Head Trackings kann sich der Spieler in der gesamten Szenerie umschauen, Bewegungen ausführen und die Objekte aus verschiedenen Perspektiven betrachten. Das Wand-Device steuert das Gewehr, um die Zielobjekte anzuvisieren und abzuschiessen. Die Buttons des Wands werden gebraucht um das Gewehr abzufeuern.

- **Umgebung**

Die Landschaft ist ein Model aus dem Asset Store von Unity mit riesigen Ausmessungen. Um die abzuschiessenden Ziele platzieren zu können, Hindernisse zu schaffen und der Szenerie Leben einzuhauen, wurden verschiedene Marktstände, Heukarren, Büsche und Zäune eingefügt.

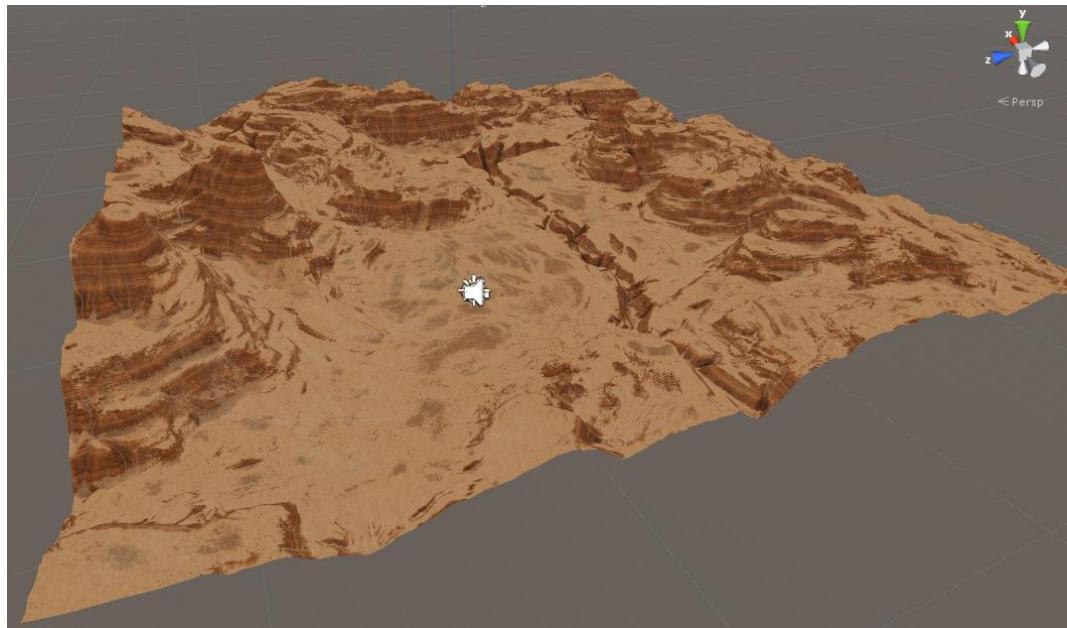


Abbildung 52: Desert Model, Quelle: Unity Asset Store

- **Gewehr**

Das Model des Gewehrs ist ebenfalls aus dem Unity Asset Store. Die Enten und Zielscheiben werden mit der Cursorposition anvisiert, welche durch das Unity Plugin vom Wand gesetzt wird.

Rotation

Das Gewehr dreht sich entsprechend dessen Position. Die Rotation wird zweierlei beeinflusst.

a) Rotation der Eyes

Basierend auf der Rotation der Eyes auf der y-Achse (Yaw) und der z-Achse (Roll) dreht sich auch das Gewehr im Spiel. Somit wird ermöglicht, dass sich der Benutzer des CAVEs drehen kann und das Gewehr immer in seine Blickrichtung zielt. Neigt er den Kopf leicht auf eine Seite, übernimmt die Flinte ebenfalls diese Manipulation rollt sich auf die Seite.

b) Relativer Winkel zwischen Eyes und Wand

Zusätzlich zu der Blickrichtung, welche mittels Eyes festgestellt wird, folgt das Gewehr der aktuellen Cursorposition. Dazu wird der relative Winkel zwischen dem Wand und den Eyes berechnet und darauf basierend erfolgt eine zusätzliche Rotation des Gewehrs.

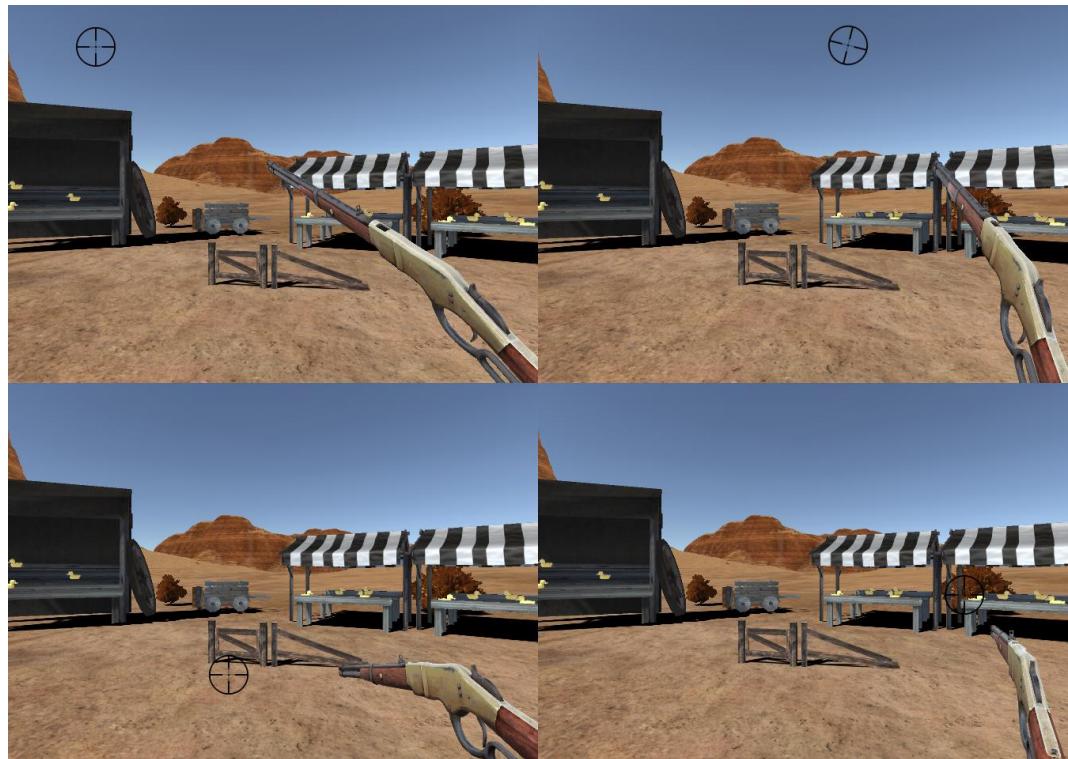


Abbildung 53: Shooting Gallery, Rotation des Gewehrs

Schuss

Dort wo sich der Cursor auf der 2D-Ebene befindet, wird auch präzise der Schuss in der 3D-Welt auftreffen. Dazu wird ein Raycast mit der Ausrichtung der Kamera an der Position des Cursors in die Umgebung geschossen und geschaut, welches Objekt als erstes im Wege steht.

Beim Auftreffen dieses virtuellen Schusses wird dem Ziel mitgeteilt, ob nun eine Interaktion erfolgen soll oder nicht. Zusätzlich wird ein Partikeleffekt, welcher den Einschuss verdeutlich, an der getroffenen Stelle erzeugt. Ein weiterer Raucheffekt wird bei der Flinte direkt gezeigt.



Abbildung 54: Shooting Gallery, Rauch

Audio

Verschiedene Audiofiles wurden integriert, um das Spielerlebnis zu verbessern. Folgende Ereignisse provozieren einen Audioeffekt:

- Feuern des Gewehrs
- Treffen einer Ente
- Treffen der Zielscheibe

- **Zielscheibe**

Eines der beiden abzuschiessenden Ziele ist eine Holzkonstruktion mit drei Zielscheiben, welche sich zu zufälligen Zeitpunkten nach oben, bzw. nach unten klappen und somit angreifbar, bzw. nicht angreifbar werden.

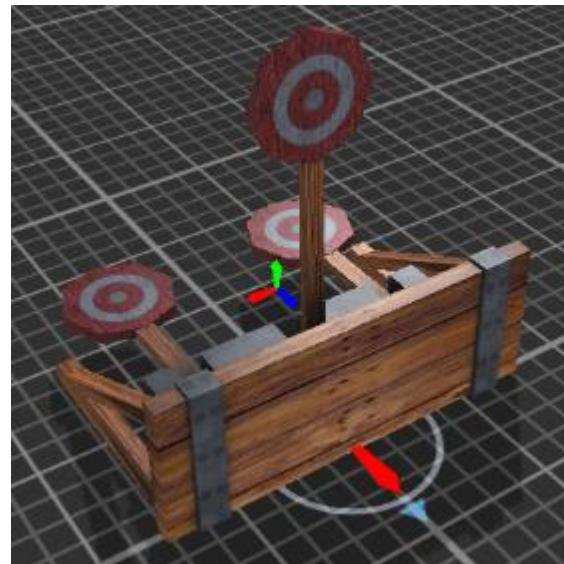


Abbildung 55: Shooting Gallery, Zielscheibe

Die Animationen sind im FBX³⁰-Model gespeichert und werden mittels einem AnimationController ausgelöst. Zu zufälligen Zeitpunkten wird eine der Show-States aktiviert um die Animation abzuspielen. Trifft während einer gewissen Zeitspanne kein Schuss die Zielscheibe, aktiviert sich der Hide-State und geht anschliessend zurück in den Idle-State. Im Gegenzug, trifft der Spieler auf die Zielscheibe, wird die Animation beim Hit-State abgespielt und es werden Punkte gutgeschrieben. Zudem wird als Audiofeedback ein entsprechender Sound gehört. Anschliessend fängt die Sequenz von vorne an.

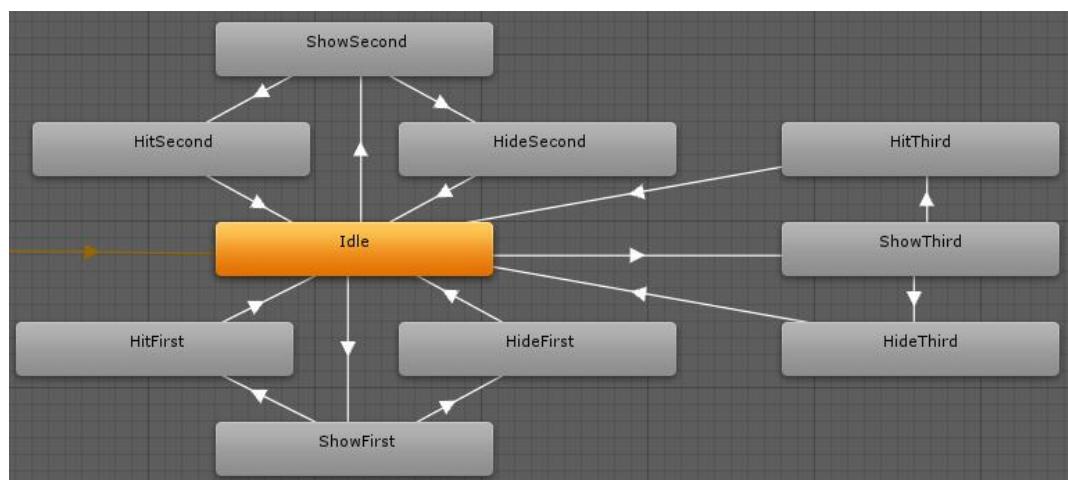


Abbildung 56: Shooting Gallery, Zielscheibe Animation Controller

³⁰ FBX – <https://en.wikipedia.org/wiki/FBX>

- **Ente**

Das zweite abzuschiessende Objekt ist eine Gummiente, die sich auf einer festgelegten Bahn mit zufälliger Geschwindigkeit nach vorne und hinten bewegt.



Abbildung 57: Shooting Gallery, Ente

Am Anfang und Ende der Bahn befinden sich jeweils Trigger, die ausgelöst werden, sobald sich das Mesh der Ente damit überschneidet. In diesem Moment ändert sich die Bewegungsrichtung der Ente und eine neue Geschwindigkeit wird zufällig zwischen einem definierten Bereich gewählt.

Bei einem Treffer wird der Winkel zwischen der aktuellen Kamera und der Ente berechnet und entsprechend ein Impuls auf die Ente angewandt, damit sie in die korrekte Richtung davonfliegt. Sobald die Ente keinen Kontakt mit der Oberfläche mehr hat auf der sie sich bewegt, wird ein Event losgeschickt um nach einer gewissen Zeitspanne die Position und Rotation wiederherzustellen und Punkte gutzuschreiben. Bei einem Treffen ist zudem ein Audiofeedback (Quaken) zu hören.

Weil das gefundene Model der Ente für die Anwendungszwecke viel zu detailliert war, musste aus Performancegründen mit Blender³¹ eine Reduktion der Faces erfolgen. Von ehemals 7160 sind noch 1288 Faces übrig geblieben. Dank der Textur und der Distanz, die zwischen dem Spieler und der Ente liegt, fällt der Unterschied nicht auf.

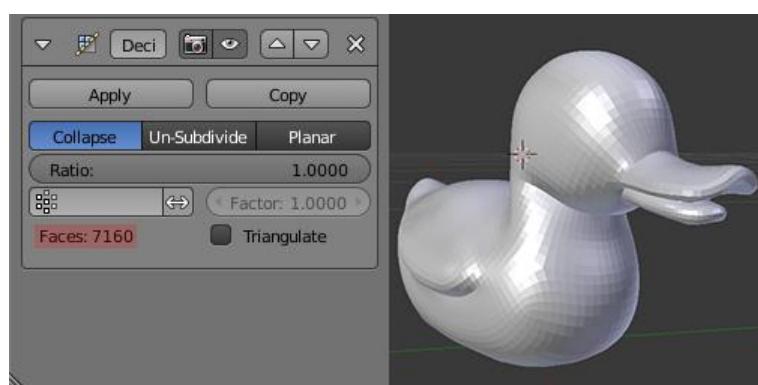


Abbildung 58: Shooting Gallery, Ente mit vielen Details

³¹ Blender – <https://www.blender.org/>

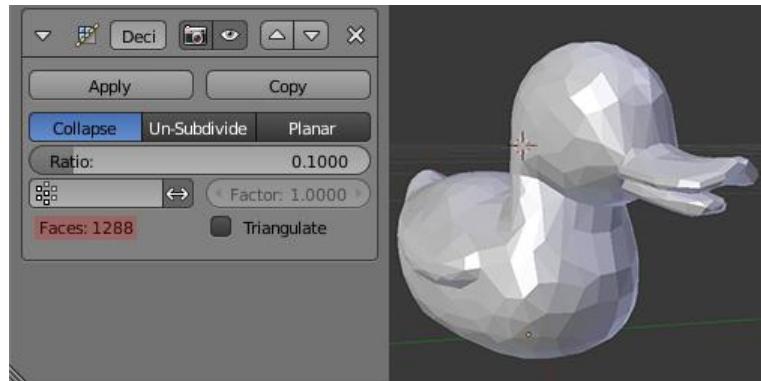


Abbildung 59: Shooting Gallery, Ente mit reduzierten Details

- **Punktesystem**

Nebst der bereits verstrichenen Zeit ist in einer Ecke als UI-Element die erspielte Punktzahl sichtbar. Die beiden abzuschiessenden Ziele (Ente und Zielscheibe) stellen öffentliche, statische Delegates (Events) zur Verfügung, mittels denen andere Objekte Methoden registrieren können. Wird eine beliebige Ente abgeschossen, wird die Methode aufgerufen. Die Klasse, welche für die Punkteberechnung und -darstellung zuständig ist, registriert entsprechend eine Methode, welche als Parameter die erspielten Punkte erhält. Somit kann dort sauber Buch geführt werden über den aktuellen Punktestand und ihn als GUI-Element anzeigen.

```

Duck.cs
    public delegate void DelegateHit(int points);
    public static event DelegateHit OnHit;

    public void Hit()
    {
        if (!alreadyHit)
        {
            alreadyHit = true;
            OnHit(POINTS);
        }
    }

UIPoints.cs
    Duck.OnHit += OnHit;

    void OnHit(int points)
    {
        _points += points;
        _text.text = "Punktzahl: " + _points.ToString();
    }

```

Quellcode 11: Shooting Gallery, Ente und Punkte

5.2 Model-Viewer

Da der CAVE nicht nur zum Spielen gedacht ist, zeigt diese zweite Demo noch ein weiteres Anwendungsgebiet. Es handelt sich um mehrere Operationssäle, welche begangen werden können.

Die gesamte Umgebung ist statisch, einige Personen haben jedoch vordefinierte Animationen.

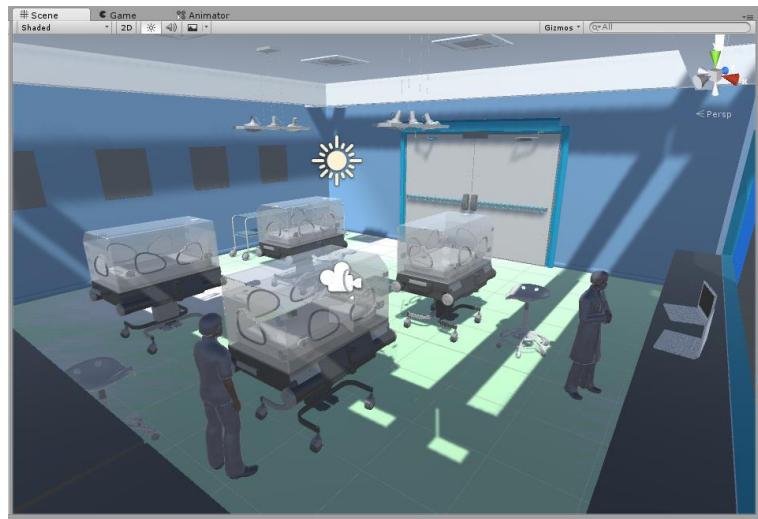


Abbildung 60: Model-Viewer OP Raum 1

- **Modelle**

Es werden Modelle von Dosch³² verwendet, welche bereits im Besitz der BFH sind.

Es folgt eine Auflistung der verfügbaren Personen und Räume. Auf die Detailliste aller Equipments wird hier verzichtet.

Person	Infos
Surgeon	Person a1
SurgeonAssistant	Person a2
TOA	Person a3
TOA	Person a4
OPattendant	Person a5
Anesthetist	Person a6

Tabelle 10: Personen Modelviewer

³² Dosch – www.doschdesign.com

Raum	Infos
MedicalRoom01	CT Raum mit Kontrollraum
MedicalRoom02	Zahnarzt
MedicalRoom03	Zahnarzt schlicht
MedicalRoom04	Bettenzimmer
MedicalRoom05	Babystation
MedicalRoom06	Zahnarzt mit Vorzimmer
MedicalRoom07	Patientendoppelzimmer
MedicalRoom08	Quadratisches Zahnarztzimmer
MedicalRoom09	Psychiater
MedicalRoom10	Fitnessstudio

Tabelle 11: Räume Modelviewer

- **Szenen**

Für jeden verfügbaren Raum ist eine vordefinierte Szene erstellt, in welcher mit dem Gamepad navigiert werden kann.

- **Hauptmenu**

Die Szenen können über ein statisches Menu geladen werden.

5.3 App Drittpartei

Dieser Anwendungsfall soll aufzeigen, dass nicht nur konkret auf das umgesetzte Unity Plugin hin erarbeitete Applikationen problemlos verwendet werden können. Ausgewählt wurde das mit Unity 5 ausgelieferte Beispielprojekt namens „Standard Assets Example Project“.

Die Integrierung des Plugins verlief einwandfrei und das Spiel konnte erfolgreich in 3D im CAVE gespielt werden. Alle Elemente, zum Beispiel die Anzeige der UI Elemente auf einem spezifischen Screen, verursachen keinerlei Schwierigkeiten. Der nachfolgende Screenshot zeigt die Aufteilung der Viewports nach dem Aktivieren des Plugins.

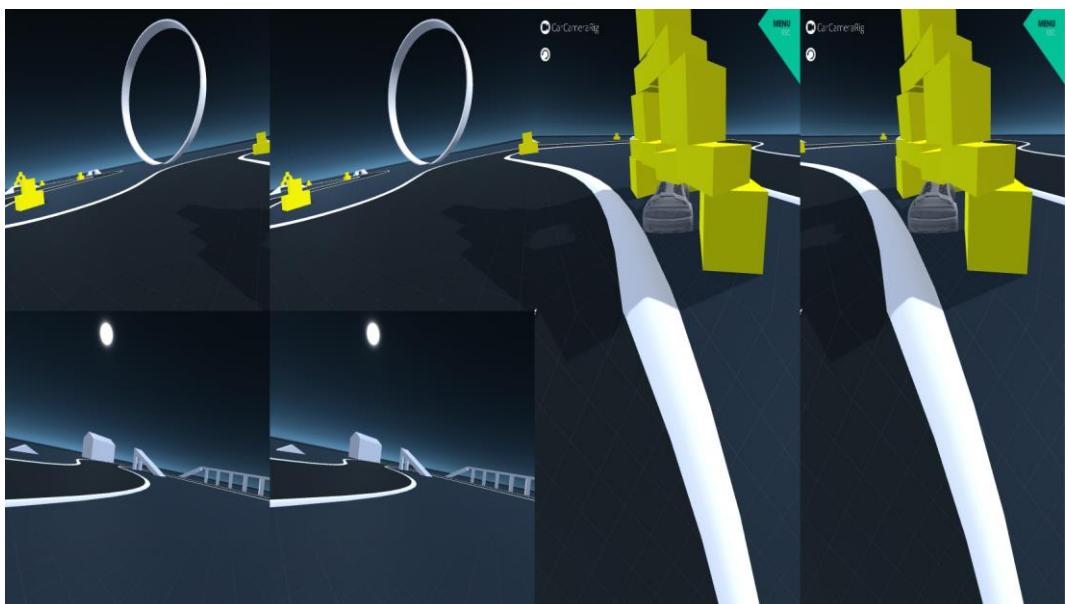


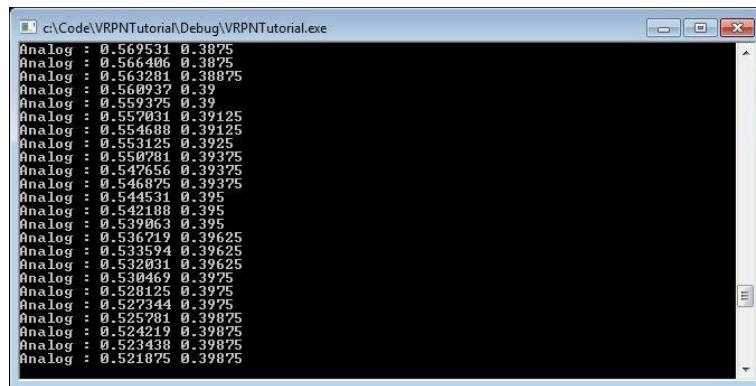
Abbildung 61: Standard Assets Example Project mit Unity Plugin

6 Testing

Dieses Kapitel beinhaltet die Tests der Prototypen, welche für verschiedene Module erstellt wurden, das Testen des Plugins mit der eigenen Testszene, sowie Performancetests der gesamten Anwendung.

6.1 PPT-Studio und VRPN

Die Funktionalität des internen VRPN Servers des PPT Studios kann mit einem einfachen Konsolenclient überprüft werden, welcher die Trackerdaten ausgibt. Das Programm «vrpn_print_devices.exe» von vrpn.org liefert diese Informationen.



```
c:\Code\VRPNTutorial\Debug\VRPNTutorial.exe
Analog : 0.569531 0.3875
Analog : 0.566406 0.3875
Analog : 0.563281 0.38875
Analog : 0.560937 0.39
Analog : 0.559325 0.39
Analog : 0.557933 0.39125
Analog : 0.554681 0.39125
Analog : 0.553125 0.3925
Analog : 0.550781 0.39375
Analog : 0.547656 0.39375
Analog : 0.546875 0.39375
Analog : 0.544531 0.395
Analog : 0.542188 0.395
Analog : 0.539863 0.395
Analog : 0.536719 0.39625
Analog : 0.533594 0.39625
Analog : 0.532031 0.39625
Analog : 0.530461 0.3975
Analog : 0.528125 0.3975
Analog : 0.527344 0.3975
Analog : 0.526701 0.3975
Analog : 0.524419 0.3975
Analog : 0.523438 0.3975
Analog : 0.521875 0.39875
```

Abbildung 62: VRPN Debug Ausgabe – <http://www.cs.unc.edu/Research/vrpn/index.html>

6.2 Wrapping VRPN in C# und Unity

Die Funktionalität der kompilierten DLL des VRPN Projekts sowie die .net Wrapperklasse wurde Anhand eines kleinen Unity Testprojektes überprüft. Dies verwendete die erhaltenen Trackerdaten direkt als Positions- und Rotationsangaben der Kamera. Weiter ist eine Plane im Raum sichtbar, welche die Ausrichtung im Unity bekanntgibt, damit dies dann mit der realen Trackingposition überprüft werden kann.

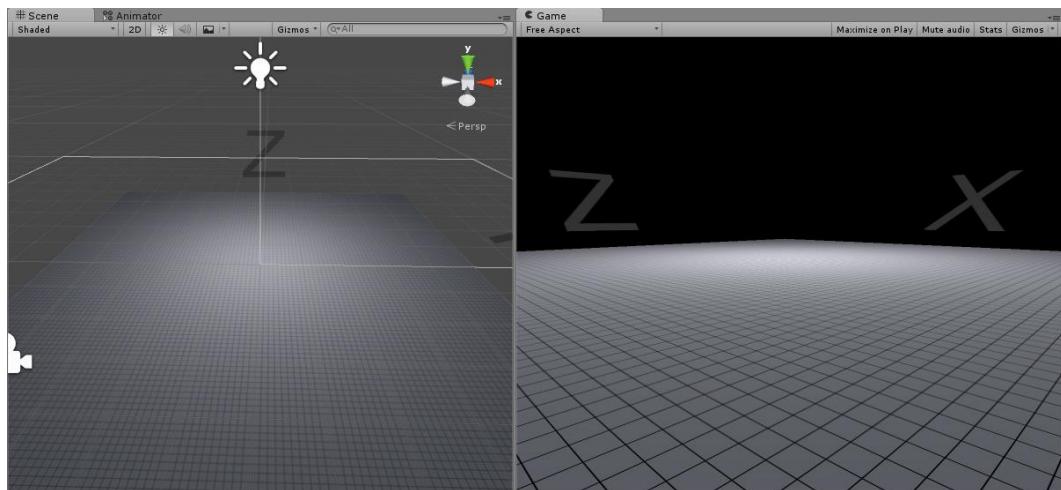


Abbildung 63: VRPN Prototyp

```

public class Tracker : MonoBehaviour
{
    public string host = "localhost";
    public string obj;
    public int channel = 0;

    public bool trackPosition = true;
    public bool trackRotation = true;

    // Update is called once per frame
    void Update()
    {
        if (trackPosition)
        {
            transform.position = VRPN.vrpnTrackerPos(obj + "@" + host, channel);
        }
        if (trackRotation)
        {
            transform.rotation = VRPN.vrpnTrackerQuat(obj + "@" + host, channel);
        }
    }
}

```

Quellcode 12: VRPN Prototyp

6.3 Unity Plugin Projekt als Debugger

Das Unity Plugin kommt mit einer Testszene, mit welcher folgende Elemente getestet werden können:

- Stereoskopie Einstellungen der Kameras
- Sekundäre Kameras
- GUI Elemente
- Mousecursor
- Verhalten der Frustumtransformation bei Bewegungen im CAVE.

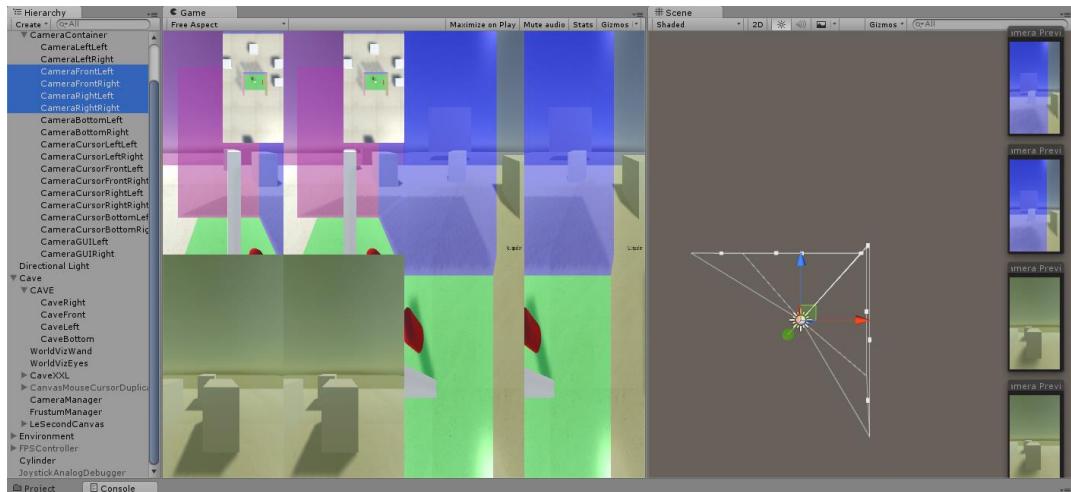


Abbildung 64: CAVE Testszene

Auf der obigen Abbildung sind die Frustums von vier Kameras sichtbar und ob die Aufspannung der Projektion korrekt ist. Eine sekundäre Kamera ist auf die linke Projektionsseite gemappt, so dass auch dieser Output direkt im Plugin überprüft werden kann. Zusätzlich können die Frustums noch über Gizmos³³ von Unity angezeigt werden.

Die Simulationen der Bewegung übernimmt ein Debugscript, welches bei Bedarf auf den Komponenten aktiviert werden kann. So führt die Komponente eine Bewegung im Raum durch zusammen mit einer Rotation.

```
void Update()
{
    TimeCounter += Time.deltaTime * speed;
    float x = Mathf.Abs(Mathf.Cos(TimeCounter) * width);
    float y = Mathf.Abs(Mathf.Sin(TimeCounter) * height);
    float z = 0.1f;

    transform.position = new Vector3(x, y, z);
    transform.rotation = new Quaternion(x, y, z, 1f);
}
```

Quellcode 13: Debugmover

6.4 Performance Tests Unity Server

Die Grafik Performance des Unity Servers ist von zentraler Bedeutung. Die Konfiguration des Unity Servers ist folgendermassen:

Auflösung: 5120 x 2048
Qualität Unity: Fantastic (maximum)

Um eine aussagekräftige Analyse zu machen, wurden die Testapplikationen zweimal ausgeführt und analysiert. Im ersten Durchlauf ohne das Unity Plugin, in einem zweiten Durchlauf mit aktiviertem Unity Plugin. So ist die Einbusse der Performance klar ersichtlich.

- **Shooting Gallery**

Diese eigens entwickelte Applikation wurde als erstes getestet.

FPS ohne Plugin: 218
FPS mit Plugin: 171

Verlust in %: 21.6 %

³³ Unity Gizmo – <http://docs.unity3d.com/ScriptReference/Gizmos.html>



Abbildung 65: Shootinggallery Demospiel ohne Plugin

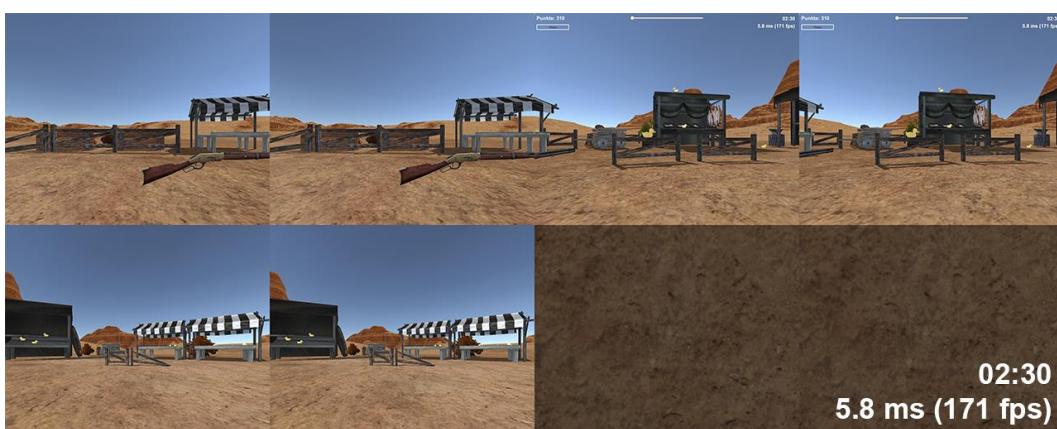


Abbildung 66: Shootinggallery Demospiel mit Plugin

- **Standard Assets Example Project**

Die von Unity gelieferte Applikation wurde ebenfalls getestet.

FPS ohne Plugin: 127

FPS mit Plugin: 92

Verlust in %: 27.7 %

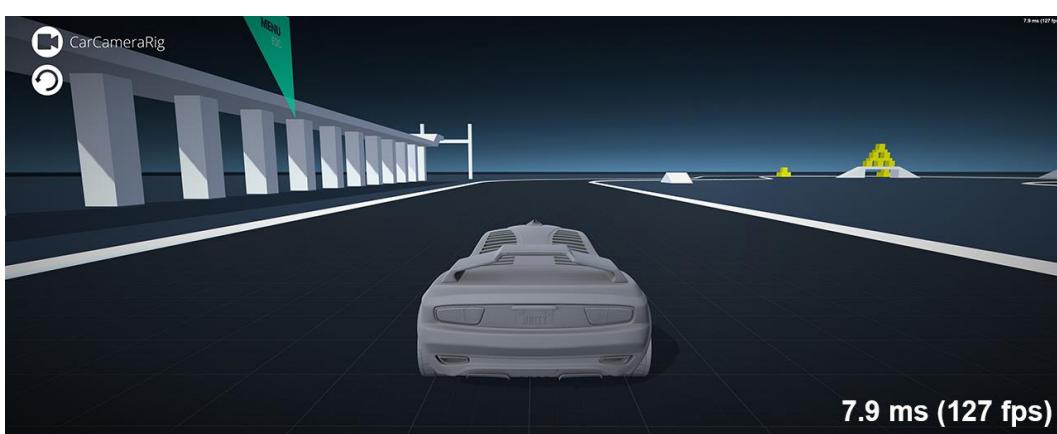


Abbildung 67: Unity Example Project ohne Plugin

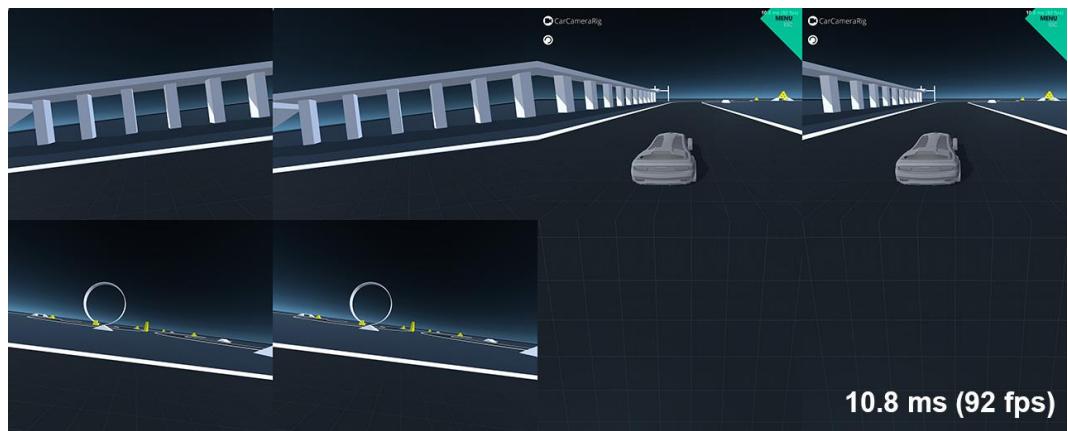


Abbildung 68: Unity Example Project mit Plugin

- **Bowling**

Das Bowling Spiel wurde im Rahmen des Moduls BTI7534p Advanced Game Development für den CAVE umgesetzt.

FPS ohne Plugin: 92

FPS mit Plugin: 55

Verlust in %: 40.2 %



Abbildung 69: Bowlingdemo AdvGameDev ohne Plugin

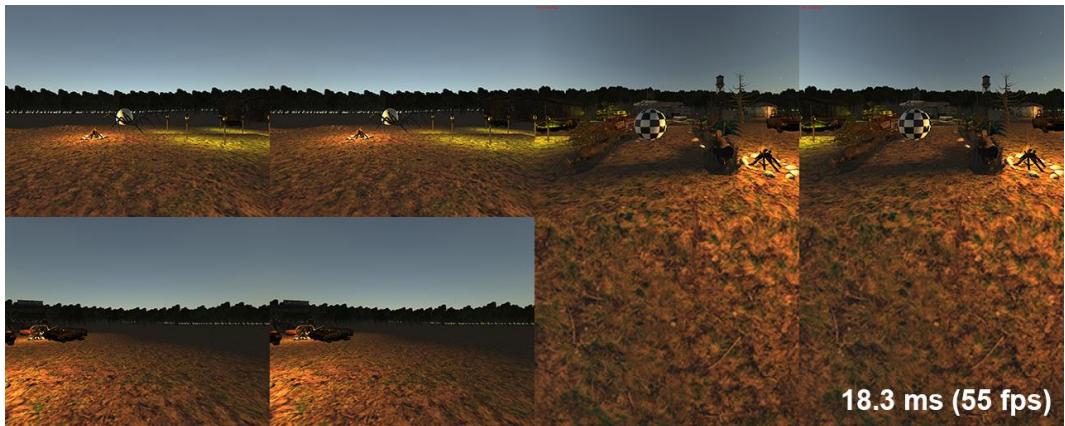


Abbildung 70: Bowlingdemo AdvGameDev mit Plugin

Fazit

In Anbetracht dessen, dass sich der Viewport vervielfacht, ist eine Einbusse von durchschnittlich 29% tragbar. Statt einem Viewport müssen nun sieben weitere, plus eventuell zusätzliche GUI-Viewports berechnet werden. Je nach Applikationen ist der Verlust auch schwindend gering. Die starke Hardware im BFH CAVE kann diesen Verlust gut ausgleichen und die Grafikeinstellungen waren bei sämtlichen Tests auf das Maximum gestellt. Werden diese auf ein Mittelmaß runtergeschraubt, vergrössert sich die Framerate wiederum um ein x-faches. Keine Applikation wurde durch das Plugin unspielbar.

7 Projektmanagement

7.1 Aufgabenstellung

Die Aufgabenstellung lautet «Neben der bestehenden CAVE Cluster-Rendering Lösung soll ein Unity 3D Render-Server in Betrieb genommen werden, um der zunehmenden Bedeutung von Unity im CPVRLab und im Unterricht Rechnung zu tragen. Es ist dabei eine Lösung zu entwickeln, welche eine einfache Integration von neuen oder bestehenden Unity Applikationen in das Multi-Screen Rendering Setup des CAVEs ermöglicht»

Folgende Anforderungen sollen dabei umgesetzt werden:

- Entwicklung eines geeigneten Setups oder APIs für ein einfaches Unity Multi-Screen Rendering
- Integration des WorldWiz Tracking Systems für User Head Tracking und Kamerasteuerung
- Implementation einer Demo-Applikation welche die Features des Frameworks demonstriert
- Erstellen eines Entwickler-Handbuchs mit detaillierte Anleitung und Tutorial

7.2 Zieldefinitionen

Durch die Ist-Soll Analyse werden die Ziele definiert.

Beschrieb	Ist	Soll
Stereoskopisches Rendering von Unity im CAVE	Der neue Unityserver ist noch unkonfiguriert. Keine Mosaiceinstellungen vorhanden und kein Plugin vorhanden.	Mosaic soll so konfiguriert werden, dass das Unity Plugin verwendet werden kann.
Verwenden des WorldViz Trackings	Die Hardware (Infrarotkameras sowie Wand und Eyes) sind vorhanden, der Trackingserver ist neu und unkonfiguriert.	Die vorhandene Hardware soll mit dem neuen Trackingserver verwendet und die Daten müssen über VPRN vom Unity Plugin empfangen werden können.
Erstellen von Demo- Applikationen	Keine Unity Demos vorhanden für den CAVE.	Demo-Applikationen programmieren für den CAVE.
Erstellen eines Handbuchs mit Tutorial	Kein Unity Handbuch für den CAVE (zusammen mit dem Plugin) vorhanden.	Handbuch, Tutorial und weitere Hilfe sollen angeboten werden.

Tabelle 12: Ist-Soll Analyse

Somit ergeben sich folgende primären Ziele, welche **im Rahmen der Bachelorthesis erreicht wurden**:

- Entwicklung eines Unity Plugins, welches das Multi-Screen Stereo Rendering, das Tracking, sowie die Berechnung der Frustums übernimmt.
- Abdeckung Spezialfälle wie sekundäre Kameras, Darstellung GUI, benutzerdefinierter Cursor usw.
- Verwenden der VR-Inputdevices (PPT Wand / Eyes) mittels einem neuen Interface im Unity Plugin.
- Demoapplikation (ShootingGallery und ModelViewer) programmiert und im CAVE vorhanden.
- Handbuch mit einer Schritt-für-Schritt Anleitung zur Verwendung des Unity Plugins erstellt.

7.2.1 Funktionale Anforderungen

Durch die Zieldefinition ergeben sich folgende funktionale Anforderungen, welche erreicht wurden:

	Prio.	Keyfeatures	Kritische Punkte	Ziel erreicht
Adaption Unity Anwendung für den CAVE	1	<ul style="list-style-type: none"> - Automatische Erstellung der zusätzlichen Unitykameras - Rotation der Unitykameras - Justierung des Frustums und FoV 	<ul style="list-style-type: none"> - Optimale Frustums, welches die genaue Position im CAVE als Unitykamera übernehmen. 	100% erreicht
VRPN Unterstützung	1	<ul style="list-style-type: none"> - Implementierung des Protokolls für VR-Eingabegeräte 	<ul style="list-style-type: none"> - VRPN Server von WorldViz ist eine Blackbox (Gerätenamen) 	100% erreicht
Kompatibilität	2	<ul style="list-style-type: none"> - Aktuelle Unityversion und zukünftige sollen supportet werden 	<ul style="list-style-type: none"> - Abhängig von Unity 	100% erreicht (Version 5.2.3)
Tracking	1	<ul style="list-style-type: none"> - Erkennung der Positionen und Rotationen von Eyes und Wand im CAVE 	<ul style="list-style-type: none"> - Abhängig von den vorhandenen Kameras, sowie deren Bildqualität 	100% erreicht
Einstellungsmöglichkeiten	2	<ul style="list-style-type: none"> - Alle Freiheitsgrade müssen pro Anwendung einstellbar oder direkt sperrbar sein 		100% erreicht
Setup, Dokumentation	2	<ul style="list-style-type: none"> - Plug & Play für Unityanwendungen 	<ul style="list-style-type: none"> - Spezielle Anwendungen können eventuell nicht durchgängig unterstützt werden 	100% erreicht
Demo-Applikationen	3	<ul style="list-style-type: none"> - Siehe Tabelle 2 (Anhang Pflichtenheft) 		100% erreicht
Unity Output Warping	4	<ul style="list-style-type: none"> - Das Ausgabebild soll auf die Eigenheiten des CAVEs angepasst werden können 	<ul style="list-style-type: none"> - Optionales Feature 	Abklärungen getroffen und Prototypen getestet

Tabelle 13: Funktionale Anforderungen

7.3 Projektorganisation

Auf eine stark strukturierte Projektorganisation wird bewusst verzichtet. Die Teammitglieder sind gleichberechtigt. Es kann vorkommen, dass verschiedene Teilprojekte und Verantwortungsbereiche den Teammitgliedern zugewiesen werden. Dies bedeutet aber nicht die alleinige Durchführung dieser Tasks.

7.3.1 Projektteam

Daniel Inversini daniel.inversini@students.bfh.ch
Julien Villiger julien.villiger@students.bfh.ch

7.3.2 Betreuer

Prof. Urs Künzler urs.kuenzler@bfh.ch

7.3.3 Experte

Dr. Harald Studer harald.studer@iss-aq.ch

7.4 Projektplanung, Meilensteine

Der Gantt³⁴-Projektplan ist unter <https://pm.ti.bfh.ch/projects/unity-cave-thesis/issues/gantt> erreichbar.

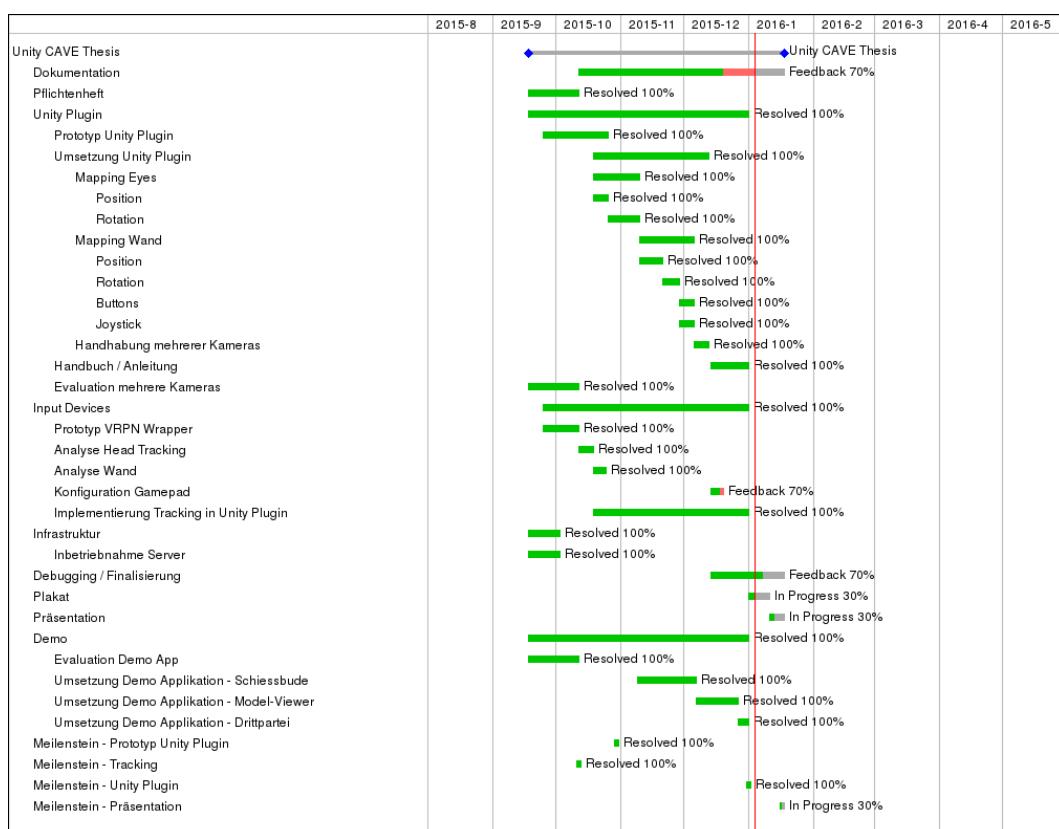


Abbildung 71: Gantt Plan

³⁴ Gantt – <http://de.wikipedia.org/wiki/Gantt-Diagramm>

Folgende Meilensteine wurden in der Voranalyse und im Pflichtenheft bereits festgelegt:

Bezeichnung	Fälligkeit
Prototyp Unity Plugin	29.10.2015
Tracking	29.10.2015
Unity Plugin / Handbuch / Dokumentation	31.12.2015
Präsentation	17.01.2015

Tabelle 14: Meilensteine

- **Prototyp Unity Plugin**

Eine erste Implementierung des Unity Plugins mit grundlegender Funktionalität wurde umgesetzt.

- **Tracking**

Die Analysephase des VRPN Protokolls wurde abgeschlossen, damit die Integration des WorldViz Tracking Systems in das Unity Plugin erfolgen kann. Zusätzlich wird die Umsetzung einer Demo-Applikation gestartet.

- **Unity Plugin / Handbuch / Dokumentation**

Das Unity Plugin wurde fertiggestellt und getestet. Kleinere Anpassungen und die Finalisierung erfolgen noch. Zusätzlich wurden ein Handbuch und eine Anleitung erstellt, um die Verwendung des Plugins zu vereinfachen. Die Dokumentation ist an dieser Stelle ebenfalls möglichst weit fertiggestellt.

- **Präsentation**

Sämtliche Dokumente und Arbeiten sind abgeschlossen und eine Präsentation wurde erstellt.

7.5 Qualitätssicherung

Als Qualitätssicherung soll regelmässig der aktuelle Ist-Zustand des Projektes mit dem Projektplan verglichen werden. Weiter finden alle zwei Wochen Projektmeetings mit dem Betreuer statt, bei denen Fortschritte und Probleme diskutiert werden können.

7.6 Risikoanalyse

Sollten erkennbare Risiken auftreten, wird der Auftraggeber informiert und es werden vom Projektteam erarbeitete Lösungsansätze vorgeschlagen.

A: Auswirkungen W: Wahrscheinlichkeit

Risiko	Beschreibung	Massnahmen	A	W
Neue Infrastruktur nicht lauffähig	Der Unity Render Server sowie der Trackingserver für WorldViz wurden erneuert, jedoch noch nicht konfiguriert und getestet.	Intensives Behandeln der WorldViz Trackingsoftware zu Beginn der Bachelorarbeit um Probleme frühzeitig zu erkennen.	Gross	Mittel
Fehlerhafte Hardware im Bereich Trackingsystem	Defekte oder störende Kameras / Devices können die Trackingpositionen beeinflussen oder verunmöglichen.	Neukalibrierung der Kameras und das Erkennen von defekten Geräten, eventuelles Austauschen der Komponenten.	Gross	Mittel
Softwareprobleme	Die vorgesehenen Softwaren (Mosaic) könnten u.U. Probleme bereiten oder nicht alle gewünschten Features unterstützen.	Frühes Testen mit Prototypen, Spezialfälle eruieren, Alternativen suchen.	Mittel	Mittel
Performance	Die Leistung der Hardware ist nicht ausreichend oder das Plugin ist zu ressourcenintensiv.	Regelmäßige Tests und Optimierungen des Plugins mittels Profiler.	Mittel	Klein
VRPN	Die Daten können in Unity nicht ausgelesen oder interpretiert werden. Die Netzwerkkapazität ist nicht ausreichend.	In einer frühen Phase das Auslesen und Interpretieren sicherstellen, ev. auf externe Libraries zugreifen oder Beispiele zu Hilfe nehmen. Frequenz der ausgelieferten Daten reduzieren.	Mittel	Mittel

Tabelle 15: Risikoanalyse

8 Fazit

Hervorzuheben ist die dankbare Unity Plattform. Durch unser Vorwissen, dem Besuch der Module „BTI7527a Game Development“ und „BTI7534p Advanced Game Development“ sowie guten Kenntnissen in C# konnte rasch und ohne grosse Einarbeitung entwickelt werden. Viele komplexe Berechnungen, beispielsweise die Ermittlung des Schnittpunktes auf einer Ebene, sind bereits in Unity integriert und können dank einem umfassenden API ausgelagert werden und erfordern keine eigene, mathematische Implementierung.

Leider sorgte das Trackingsystem von WorldViz regelmässig für unangenehme Überraschungen. Die Bestimmung der Position und Rotation ist häufig unzuverlässig und ein Device war eine Zeit lang sogar defekt und wurde dank der guten Zusammenarbeit mit dem CPVR-Lab rasch repariert. Trotzdem beeinträchtigt die Ungenauigkeit dieses Systems die verknüpften Applikationen.

Die vorgängig detaillierte Planung konnte uns vor Engpässen bewahren und dank dem bereits zu Beginn grossen Einsatz eine Zuspitzung der Arbeitsleistung am Ende des Semesters verhindern. Der grosse Vorteil einer Partnerarbeit ist die stetige Kommunikation. Dies fördert die Kreativität bei der Lösungserarbeitung eines Problems und verhindert den Gang in eine gedankliche Sackgasse.

Sehr Motivierend ist die Tatsache, dass wir grosses Potenzial in der zukünftigen Verwendung des Plugins sehen und den CAVE der BFH beleben wird. Die beiden Unity Module sorgen in Zukunft für eine grosse Bandbreite an geeigneten Unity Applikationen, die im CAVE Verwendung finden könnten.

Ein grosser Lerneffekt wurde in der Verwendung von Unity erzielt. Durch die intensive Nutzung vieler Komponenten verfestigte sich der Umgang mit der Spielentwicklungsplattform erneut. Weiter war die Anwendung des Trackingsystems, also die Interaktion mit Soft- und Hardware, eine gute Erfahrung und zeigte dessen Schwierigkeiten auf, die es zu meistern galt.

Basierend auf dem umgesetzten Plugin sind weitere Arbeiten denkbar. Dazu gehört die Umsetzung des Warpings, also eine Behebung der Bildverzerrung durch die Projektoren. Möglich ist auch eine Erweiterung der unterstützten Input-Devices, beispielsweise des bereits vorhandenen Trackingballs. Damit das Plugin auch weiterhin problemfrei funktioniert, müssen möglicherweise bei neuen Unity-Versionen Adaptionen gemacht werden.

Im Allgemeinen wurde das Produkt zielstrebig umgesetzt und bietet einen Mehrwert für den CAVE der BFH.

9 Abbildungsverzeichnis

Abbildung 1: Vergleich Varianten Projekt 2	5
Abbildung 2: Prototyp 1 Projekt 2	7
Abbildung 3: Prototyp 2 Projekt 2	7
Abbildung 4: CAVE BFH	8
Abbildung 5: Infrastruktur CAVE	9
Abbildung 6: PPT Eyes, Quelle: www.worldviz.com	10
Abbildung 7: PPT Wand, Quelle: www.worldviz.com	11
Abbildung 8: Gamepad, Quelle: www.androidrundown.com	11
Abbildung 9: Übersicht der Komponenten	14
Abbildung 10: Paketdiagramm simplifiziert	15
Abbildung 11: Sequenzdiagramm	17
Abbildung 12: Einstellungen Augendistanz	18
Abbildung 13: Custom Cursor	19
Abbildung 14: Minimap	20
Abbildung 15: Sekundäre Kameras Einstellungen	20
Abbildung 16: Mosaic Setting Unity Server	22
Abbildung 17: Frustum, Quelle: www.stackoverflow.com	23
Abbildung 18: Unity Plugin Frustum 1	24
Abbildung 19: Unity Plugin Frustum 2	24
Abbildung 20: Vergleich FoV 120 & 60 Grad	25
Abbildung 21: Unity Plugin Frustum GGM	25
Abbildung 22: Frustum Mode	26
Abbildung 23: Frustum CAVE XXL FoV Anpassung	27
Abbildung 24: Unity Plugin, virtueller CAVE	28
Abbildung 25: Virtueller Wand und Eyes	29
Abbildung 26: Unity Plugin, Schnittpunkt Wand / CAVE	30
Abbildung 27: Zuordnung Schnittpunkt CAVE und Unity Server	31
Abbildung 28: Wand Button Zuordnung	31
Abbildung 29: Inputmanager Unity	34
Abbildung 30: Beispielverbindung VRPN, Quelle WorldViz Dokumentation	35
Abbildung 31: VRPN Wrapper C#	35
Abbildung 32: 1€ Smoothing Vergleich	37
Abbildung 33: Unity Plugin, Settings Wand	38
Abbildung 34: Unity Plugin, Settings Eyes	39
Abbildung 35: Unity Plugin, Settings Sekundäre Kameras	40
Abbildung 36: Unity Plugin, Settings Cave	40
Abbildung 37: Unity Plugin, Settings System	41
Abbildung 38: Unity Plugin, Klassendiagramm	42

Abbildung 39: Unity Plugin, Export Package	49
Abbildung 40: Unity Plugin, Export Package Selection	49
Abbildung 41: Kopieren des Unity Plugins	50
Abbildung 42: Unity Plugin, Import Package	50
Abbildung 43: Hinzufügen Cave Prefab	51
Abbildung 44: Player Settings, Api Compatibility Level	52
Abbildung 45: Platform Settings	53
Abbildung 46: Zuweisung Prefabs	53
Abbildung 47: Export 64bit	54
Abbildung 48: Performance analysieren mit Unity Profiler	55
Abbildung 49: Initialposition Warpingbeispiel	57
Abbildung 50: Verkleinerung Perspektive Warping	57
Abbildung 51: Shooting Gallery Ingame	59
Abbildung 52: Desert Model, Quelle: Unity Asset Store	60
Abbildung 53: Shooting Gallery, Rotation des Gewehrs	61
Abbildung 54: Shooting Gallery, Rauch	61
Abbildung 55: Shooting Gallery, Zielscheibe	62
Abbildung 56: Shooting Gallery, Zielscheibe Animation Controller	62
Abbildung 57: Shooting Gallery, Ente	63
Abbildung 58: Shooting Gallery, Ente mit vielen Details	63
Abbildung 59: Shooting Gallery, Ente mit reduzierten Details	64
Abbildung 60: Model-Viewer OP Raum 1	65
Abbildung 61: Standard Assets Example Project mit Unity Plugin	66
Abbildung 62: VRPN Debug Ausgabe – http://www.cs.unc.edu/Research/vrpn/index.html	67
Abbildung 63: VRPN Prototyp	67
Abbildung 64: CAVE Testszene	68
Abbildung 65: Shootinggallery Demospiel ohne Plugin	70
Abbildung 66: Shootinggallery Demospiel mit Plugin	70
Abbildung 67: Unity Example Project ohne Plugin	70
Abbildung 68: Unity Example Project mit Plugin	71
Abbildung 69: Bowlingdemo AdvGameDev ohne Plugin	71
Abbildung 70: Bowlingdemo AdvGameDev mit Plugin	72
Abbildung 71: Gantt Plan	75

10 Tabellenverzeichnis

Tabelle 1: Mosaic Setting schematisch	21
Tabelle 2: CaveMain Methoden	43
Tabelle 3: CaveMain Enums	43
Tabelle 4: Eyes Methoden	44
Tabelle 5: Wand Methoden	45
Tabelle 6: CameraManager Methoden	46
Tabelle 7: CameraManager Enums	46
Tabelle 8: CameraContainer Methoden	46
Tabelle 9: FrustumManager Methoden	47
Tabelle 10: Personen Modelviewer	65
Tabelle 11: Räume Modelviewer	66
Tabelle 12: Ist–Soll Analyse	73
Tabelle 13: Funktionale Anforderungen	74
Tabelle 14: Meilensteine	76
Tabelle 15: Risikoanalyse	77

11 Quellcodeverzeichnis

Quellcode 1: Kamerainstanziierung	19
Quellcode 2: Zugriff über das API	29
Quellcode 3: Raycast	30
Quellcode 4: Unity Tastaturabfrage	32
Quellcode 5: Joystick Handling	33
Quellcode 6: VRPN–Positionshandling	36
Quellcode 7: Einstellungsänderung über API	41
Quellcode 8: CameraManager DataStruct	45
Quellcode 9: API	48
Quellcode 10: Zugriff über das API	48
Quellcode 11: Shooting Gallery, Ente und Punkte	64
Quellcode 12: VRPN Prototyp	68
Quellcode 13: Debugmover	69

12 Glossar

dreidimensionale Körper zu modellieren, sie zu texturieren, zu animieren und zu rendern.		
C		
CameraContainer	18, 41, 43, 46, 53	
Das plugineigene Konstrukt um die Kameras zu verwenden.		
I		
1€ Filter	36	
Smoothingfilter um die Positionen von VRPN zu glätten.		
A		
AnimationController	62	
Ist eine Zustandsmaschine welche die zu spielende Animation auf einem Objekt bestimmt sowie Übergängen zwischen Animationen ermöglicht.		
API	21, 29, 32, 33, 41, 43, 44, 47, 48, 52	
Eine Programmierschnittstelle, genauer Schnittstelle zur Anwendungsprogrammierung, häufig nur kurz API, ist ein Programmteil, der von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird.		
Asset Store	48, 59, 60	
Der Unity Asset Store ist eine Sammlung von frei verfügbaren wie auch kommerziellen Assets (Plugins) für Unity. Es können Scripts, Models, Texturen, Sound, etc. sein. Alles was Unity unterstützt.		
B		
BFH	1, 2, 8, 28, 65	
Berner Fachhochschule BFH Die Institution des Studiums.		
Blender	63	
Blender ist eine freie, mit der GPL lizenzierte 3D-Grafiksoftware. Sie enthält Funktionen, um		
C		
CAVE	1, 2, 5, 8, 9, 12, 13, 16, 19, 23, 24, 25, 26, 27, 28, 30, 31, 38, 40, 43, 45, 47, 56, 57, 65, 66, 68, 71, 72, 73	
Bezeichnet den Raum, der verwendet wird um die dreidimensionale Illusionswelt zu erschaffen.		
Chromium	5	
Chromium Rendering System		
Bezeichnet ein System, welches fähig ist, interaktives Rendering auf mehreren Clusters durchzuführen.		
Cluster	73	
Ein Rechnerverbund oder Computercluster, meist einfach Cluster genannt, bezeichnet eine Anzahl von vernetzten Computern.		
CMake	35	
CMake (cross-platform make) ist ein plattformunabhängiges Programmierwerkzeug für die Entwicklung und Erstellung von Software.		
Coroutinen	32	
Coroutinen sind eine Verallgemeinerung des Konzepts einer Prozedur oder Funktion. Der prinzipielle Unterschied zwischen Coroutinen und Prozeduren ist, dass Coroutinen ihren Ablauf unterbrechen und später wieder aufnehmen können, wobei sie ihren Status beibehalten.		

Cursor	18, 19, 31, 39, 44, 45, 46, 61	FixedUpdate	56
Der Mauspfeil unter Windows.		Unityfunktion, welche ein Update zu bestimmten Zeitpunkten durchführt und nicht jedes Frame.	
	D		
Delegates	32, 45, 64	FoV	<i>Siehe Field of View</i>
Objektorientierte Variante von Methodenzeiger unter C#.		Frustum	23, 24, 25, 26, 27, 43, 47
DLL	35, 67	Frustum	Frustum ist in der Computergrafik die mathematische Abbildung eines 3D-Universums auf den Bildschirm.
Auch oft als dynamic linked Library verwendet, enthält Programmcode, Daten und Ressourcen für eine Anwendung (EXE).			
	E		G
Equalizer	5	Git	35
Equalizergraphics Parallel Rendering Middleware-Software, um OpenGL basierende Software auf mehrere Nodes, Graphikkarten und Prozessoren zu verteilen.		Git	Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien.
Eyes	10, 13, 16, 28, 29, 33, 36, 39, 40, 43, 44, 46, 47, 56, 60, 73	Gizmo	69
WorldViz Eyes		Visuelles Hilfsmittel von Unity, um in der Szenenview Objekte anzuzeigen.	
Brillengestütztes Trackinggerät, welches mit Infrarot LEDs ausgerüstet ist.		GPU	6
	F		
FBX	62	GPU	Grafikprozessor, ist ein auf die Berechnung von Grafiken spezialisierter und optimierter Prozessor für Computer, Spielkonsolen und Smartphones.
Dateiformat für 3D Modelle von Autodesk.		GUI	2, 18, 19, 26, 31, 40, 46, 64, 68
Field of View	24	GUI	Grafische Benutzeroberfläche GUI bezeichnet eine Form von Benutzerschnittstelle eines Computers. Sie hat die Aufgabe, Anwendungssoftware auf einem Rechner
Sichtfeld des Blickes, wie viele Grad sichtbar sind.			

mittels grafischer Symbole, Steuerelemente oder auch Widgets genannt, bedienbar zu machen.

Gyrometer 10, 33

Ein Gyrometer registriert beziehungsweise misst Drehbewegungen.

H

Hauptkamera 18, 19, 28, 40, 41, 43, 46, 55

Das Objekt in Unity, welche den standartmässigen Videooutput rendert.

I

immersive *Siehe Immersion*

InputSimulator 32

C# Interface zum Simulieren von Tastatur- und Mauseingaben über den Win32 SendInput Befehl.

K

Keycode 32

Definition der Tastaturtaste, Moustaste, und weiteren Eingabegeräten (Mausrad etc).

L

Lowpass 36

Filter, welcher die Signalanteile mit Frequenzen unterhalb ihrer Grenzfrequenz annähernd ungeschwächt passieren lässt, Anteile mit höheren Frequenzen dagegen dämpft.

M

Mesh 63

Untereinander mit Kanten verbundene Punkte bilden in der Computergrafik ein Mesh, Polygonnetz, das die Gestalt eines Polyeders definiert.

MiddleVR 5, 77

MiddleVR for Unity

Kommerzielle Applikation, welche die Erstellung von VR Anwendungen unterstützt. Verschiedene Soft- und Hardware werden unterstützt und verbindet sie untereinander,

Minimap 19, 20

Stellt einen Überblick über die Spielwelt dar. Abstrahiert und simplifiziert die Details, sodass nur wichtige Akteure (Spieler, Quests, Gelände) angezeigt werden.

Mosaic 6, 11, 21, 22, 73, 77

Die NVIDIA Mosaic Mehrbildschirm Technologie dient zur einfachen Skalierung jeder Anwendung auf mehrere Bildschirme, und das ohne Softwareanpassungen oder Leistungseinbussen.

Motion-Tracking 34

Das Verfolgen von Körperbewegungen des Benutzers.

N

Nvidia 6, 11, 21

Einer der weltweit grössten Hersteller von Chipsätzen und Graphikprozessoren.

P

Polfilter 2, 8, 10

Ein Polarisationsfilter (kurz auch Polfilter) ist ein Polarisator für Licht, der auf Dichroismus beruht, also komplementär polarisiertes Licht absorbiert, statt es wie polarisierende Strahlteiler zu reflektieren. Dadurch eignet es sich, Lichtstrahlen, die in der „falschen“ Ebene schwingen, zu unterdrücken.

PPT Studio 2013 9, 33

Software von WorldViz welche die Installation im CAVE der BFH softwareseitig unterstützt (bietet Kalibrierungseinstellungen, VRPN Output und vieles mehr an).

Prefab 28, 33, 38, 40, 51, 53

Unity Hilfsmittel, um Spielobjekte zu Instanzieren.

R

Raycast 26, 30, 45, 56, 61

Das Verwenden von Rays (Strahlen) in der Computergraphik um Bilder zu Rendern oder 3D Objekte in der Tiefe zu erkennen.

Render 18, 73, 77

Das Berechnen der Computergraphik. Meist über die Graphikkarte.

S

Smoothing 36, 37, 39, 40, 44
Siehe 1€ Filter

Stereoskopie 2, 18, 68

Die Stereoskopie ist die Wiedergabe von Bildern mit einem räumlichen Eindruck von Tiefe, der physikalisch nicht vorhanden ist.

Stereoskopische *Siehe Stereoskopie*
U

Unity 1, 2, 5, 6, 11, 13, 16, 19, 20, 21, 22, 24, 25, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 46, 47, 48, 49, 50, 52, 53, 56, 57, 59, 60, 66, 67, 68, 69, 70, 71, 73, 76, 77

Unity ist eine Laufzeit- und Entwicklungsumgebung für Spiele und Simulationen

V

Video Matrix Switch 12

Hardwareswitch, um verschiedene Video/Bild Ein- und Ausgänge zu koppeln.

Viewfrustum *Siehe Frustum*

Viewport 19, 55

Je nach Anwendungsgebiet ein Ausschnitt eines Bildes, eines Videos, einer virtuellen oder realen Welt bezeichnet oder der für die Darstellung zur Verfügung stehende Bereich.

VR Namespace 20

Unity Namespace für VR Geräte wie die Oculus Rift.

VRPN 9, 29, 32, 33, 34, 35, 36, 39, 40, 43, 44, 45, 47, 67, 68, 76

Virtual Reality Peripheral Network
Stellt verschiedene Tools (Server, Client, Klassen und Protokolle) zur Verfügung, um ein transparentes Interface zwischen einer Applikation und physikalischen Geräten zu bieten.

W		
Wand	10, 11, 13, 16, 19, 23, 26, 28, 29, 30, 31, 32, 33, 36, 38, 39, 44, 45, 47, 56, 59, 60, 73, 74	Windows 21, 39
Einhändiger Controller, welcher über Infrarot LEDs verfügt, sowie mehrere Buttons und einen analogen Joystick.		Microsoft Windows Betriebssystem von Microsoft, aktuell in der Version 10.
Warping	57	WorldViz 9, 10, 33, 34, 73, 76, 77
		Kommerzielle Firma welche komplette Systeme mit Infrarotkameras und Software anbietet.
		Technologie, um ein Bild zu verzerrn.

13 Literaturverzeichnis

- chiragraman, G. (01. 01 2016). *github.com/chiragraman*. Von github.com/chiragraman: <https://github.com/chiragraman/Unity3DProjectionMapping> abgerufen
- Géry Casiez, N. R. (5. 5 2012). <http://cristal.univ-lille.fr/>. Von <http://cristal.univ-lille.fr/>: <http://cristal.univ-lille.fr/~casiez/publications/CHI2012-casiez.pdf> abgerufen
- Kooima, R. (01. 01 2008). *Generalized Perspective Projection*. Louisiana: Louisiana State University, Computer Science and Engineering.
- Nvidia. (01. 01 2015). www.nvidia.de. Von www.nvidia.de: <http://www.nvidia.de/object/nvidia-mosaic-technology-de.html> abgerufen
- Unity. (1. 3 2015). *Unity 3D*. Von <http://docs.unity3d.com/Manual/VROverview.html> abgerufen
- Warping, W. I. (01. 01 2016). *Wikipedia*. Von Wikipedia: https://en.wikipedia.org/wiki/Image_warping abgerufen

14 Anhang

- Benutzerhandbuch
- Benutzerhandbuch WorldViz
- Selbstständigkeitserklärung
- Pflichtenheft
- Plakat
- Bucheintrag
- Aufgabenstellung



Unity 3D Server for CAVE Rendering

Benutzerhandbuch

Julien Villiger
Daniel Inversini
22.12.2015

1 Systeme starten

Um Unity im CAVE in Betrieb zu nehmen, gilt es folgende Systeme zu starten.

1. CS0 starten.



2. Trackingserver starten.



3. Unity Server starten.



4. Korrektes Mapping am Video-Matrix-Switch vornehmen.

Der Unity Server liefert Bilddaten an die Ports 21 – 28 und die Beamer im CAVE sind von der linken Leinwand ausgehend an den Ports 1 – 8 angeschlossen. Daraus ergibt sich folgende Zuweisung:

Zuordnung Video-Matrix-Switch								
Output	21	22	23	24	25	26	27	28
Input	1	2	3	4	5	6	7	8



5. Alle 8 Beamer starten.

2 Import des Unity Plugins

Nun muss das Plugin in die eigene Unity-Applikation integriert werden.

1. Auf den Unity Server zugreifen (per Remotedesktop oder direkt den freigegebenen Ordner öffnen).

Server

Name: RSO
IP: 192.168.0.109

Login

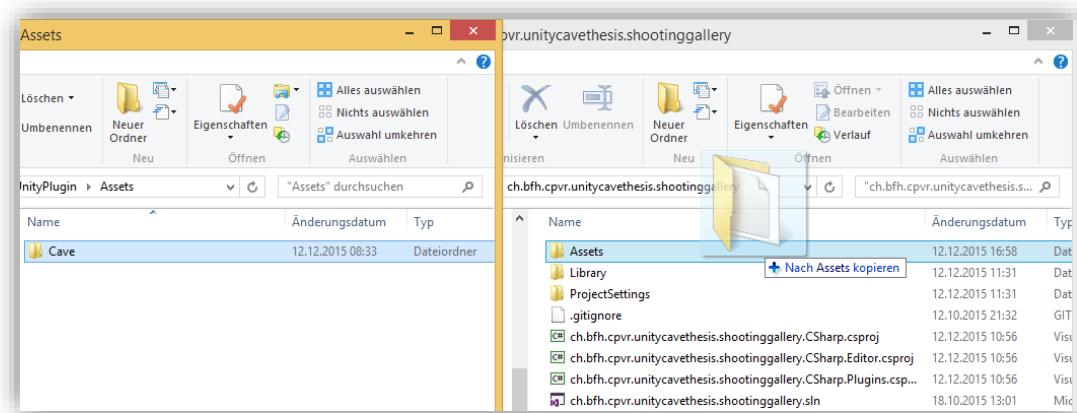
Name: cpvr
Passwort: cpvr

Ordner

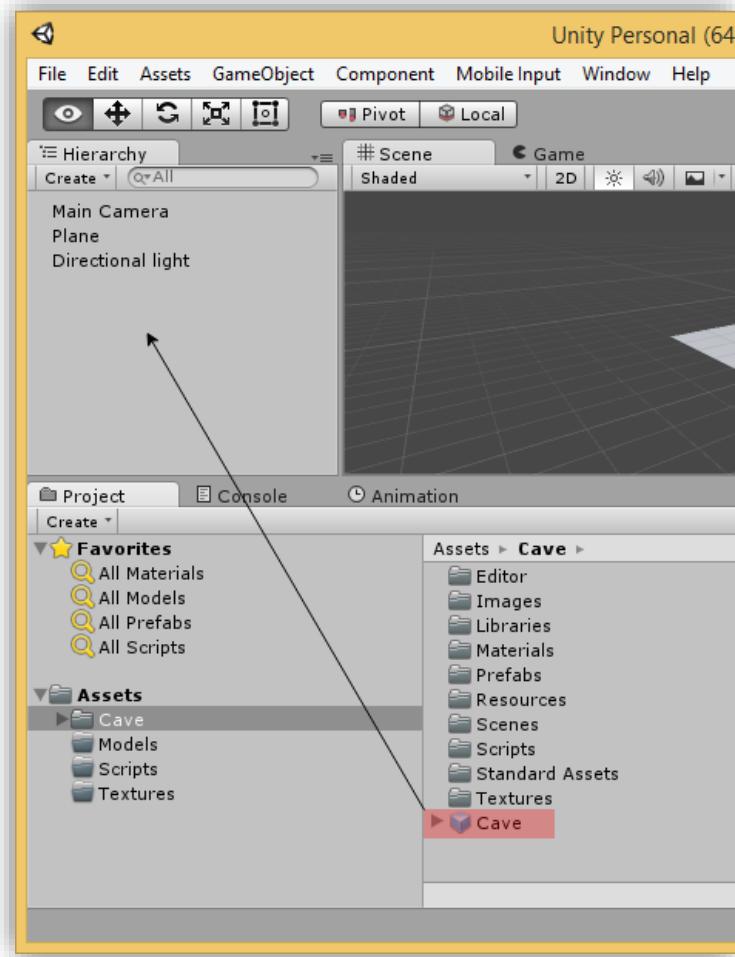
D:\UnityPlugin\Assets

Vorsicht: Der angeschlossene Computer muss sich im CAVE-Netzwerk befinden, d.h. Netzwerkkabel anschliessen.

2. Ordner „Cave“ kopieren und in den eigenen Assets-Ordner einfügen.

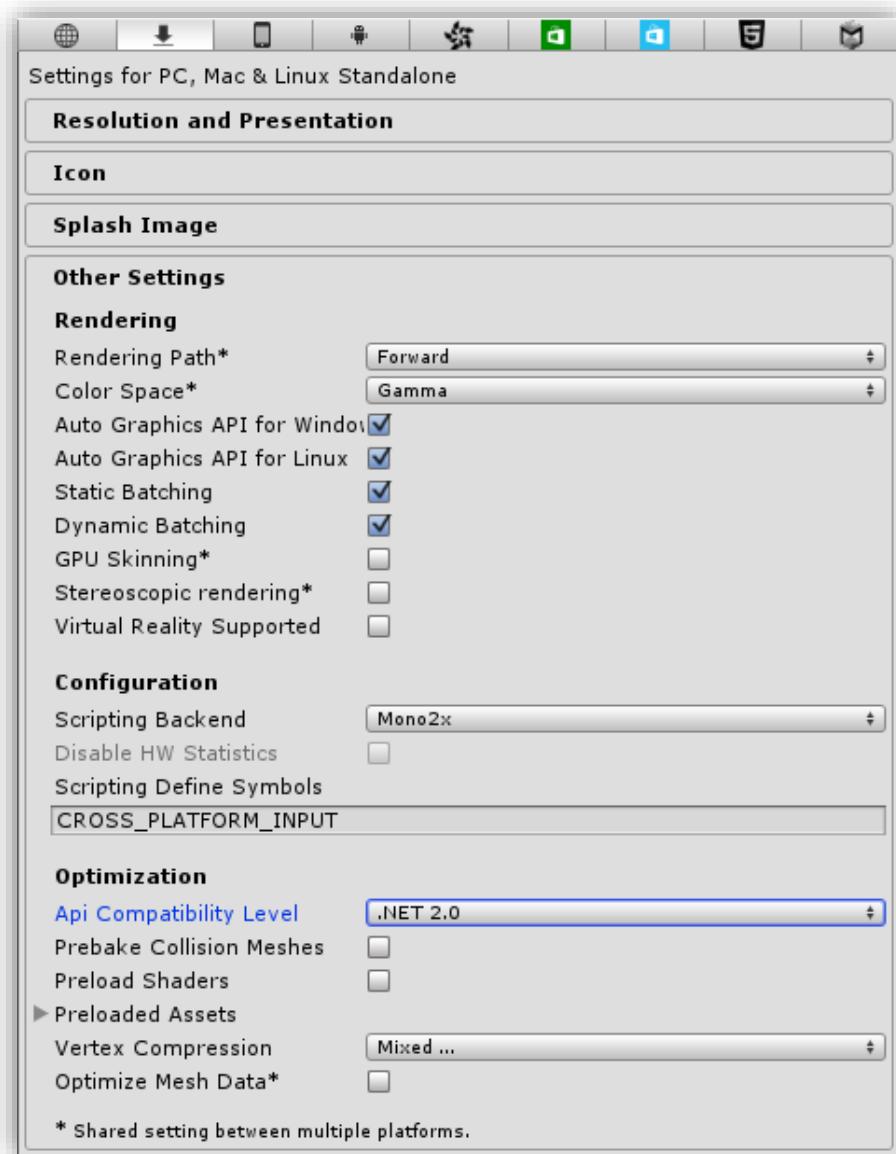


3. Das Cave-Prefab ins Root der Hierarchie kopieren.



4. API Kompatibilitätslevel auf .NET 2.0 setzen.

Dazu bei den Player Settings unter Optimization das „Api Compatibility Level“ auf „.NET 2.0“ (nicht Subset) setzen.



5. Sicherstellen, dass die zusätzlichen DLLs für die Zielplattform exportiert werden. Dazu im Projekt zu den DLLs navigieren (Cave -> Libraries) und bei allen DLLs die entsprechenden Häkchen setzen.



6. Die gewünschten Einstellungen des Plugins können nun im Inspector vorgenommen werden. Für weitere Informationen bezüglich den Einstellungsmöglichkeiten bitte die Dokumentation konsultieren.
7. Falls gewünscht, kann mittels API auf Komponenten des Plugins zugegriffen werden. Für weitere Informationen bitte die Dokumentation konsultieren.

8. Applikation auf den Unity Server exportieren.

Die Unity Applikation muss als Standalone für Windows exportiert werden. Dazu gibt es auf dem Unity Server einen entsprechenden Ordner namens „Export“:

Server

Name: RSO
IP: 192.168.0.109

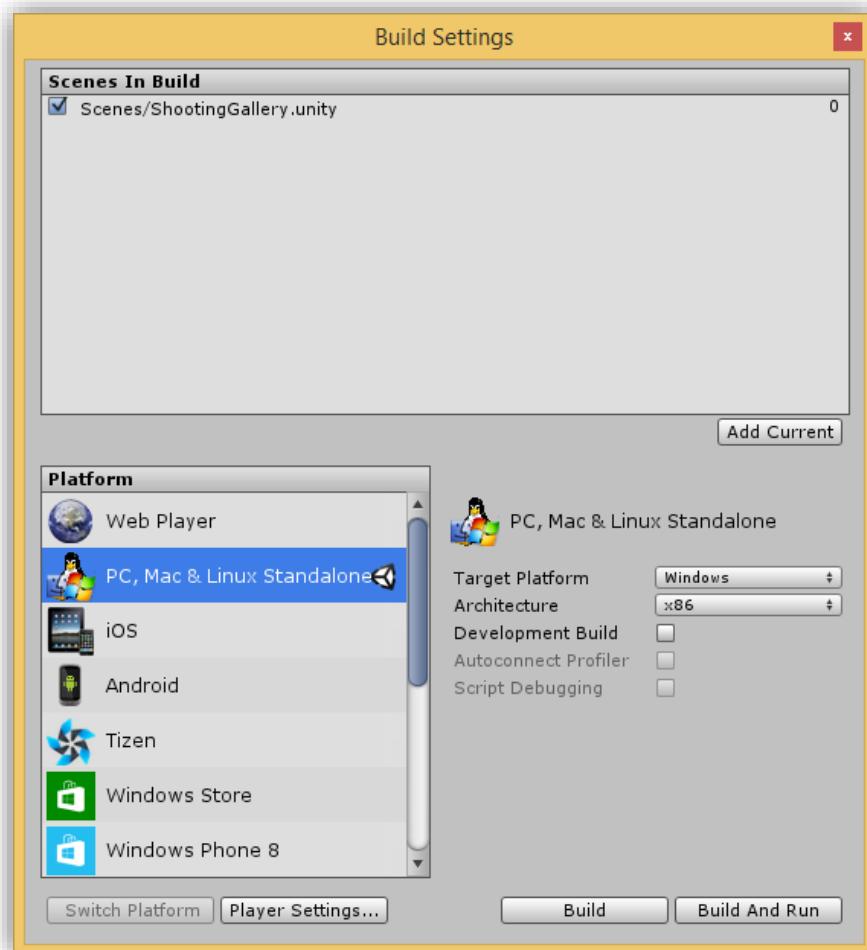
Login

Name: cpvr
Passwort: cpvr

Ordner

D:\Export

Vorsicht: Der angeschlossene Computer muss sich im CAVE-Netzwerk befinden, d.h. Netzwerkkabel anschliessen.



3 Einrichten Tracking

Um das Tracking nutzen zu können, auf dem Tracking Server (TS1) die benötigten Applikationen starten.

1. Mittels Switch-Konsole auf den Tracking Server (TS1) verbinden.

Server

Name: TS1
IP: 192.168.0.201

Login

Name: cpvr
Passwort: cpvr

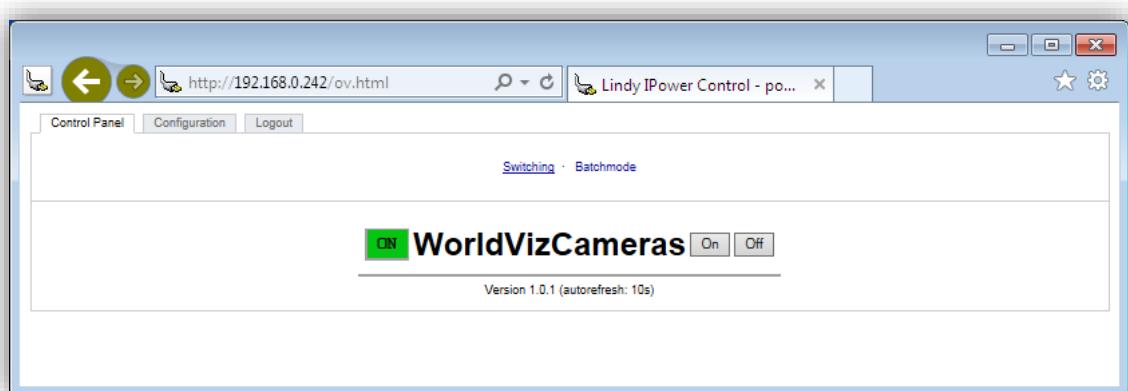
2. Server für die Kameras starten.



3. Stromversorgung der Kameras einschalten.



Anschliessend den On-Schalter betätigen. Es sollte ein leichtes Summen der Kamera-Lüfter hörbar werden.



4. Nun warten, bis die IPs an die Kameras verteilt wurden. Nach kurzer Zeit sollte sich ungefähr dieses Bild zeigen:

```
01_Server - Cameras
s Client 00:30:53:0e:20:8b <PPTH-0e208b> allotted 192.168.99.105 for 360000 second
s Client 00:30:53:0d:ef:48 <PPTH-0def48> offered 192.168.99.101
Client 00:30:53:0d:ef:48 <PPTH-0def48> allotted 192.168.99.101 for 360000 second
s Client 00:30:53:0e:20:88 <PPTH-0e2088> offered 192.168.99.104
Client 00:30:53:0e:20:8c <PPTH-0e208c> offered 192.168.99.107
Client 00:30:53:0e:20:88 <PPTH-0e2088> allotted 192.168.99.104 for 360000 second
s Client 00:30:53:0e:20:8c <PPTH-0e208c> allotted 192.168.99.107 for 360000 second
s Client 00:30:53:0d:e9:ad <PPTH-0de9ad> offered 192.168.99.102
Client 00:30:53:0d:e9:ad <PPTH-0de9ad> allotted 192.168.99.102 for 360000 second
s Client 00:30:53:0d:ef:4f <PPTH-0def4f> offered 192.168.99.108
Client 00:30:53:0f:43:16 <PPTH-0f4316> offered 192.168.99.100
Client 00:30:53:0e:20:89 <PPTH-0e2089> offered 192.168.99.106
Client 00:30:53:0d:ef:4f <PPTH-0def4f> allotted 192.168.99.108 for 360000 second
s Client 00:30:53:0f:43:16 <PPTH-0f4316> allotted 192.168.99.100 for 360000 second
s Client 00:30:53:0e:20:89 <PPTH-0e2089> allotted 192.168.99.106 for 360000 second
s
```

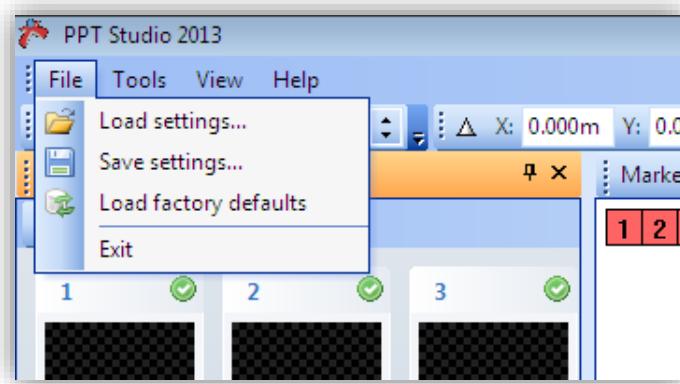
5. PPT Studio 2013 starten.



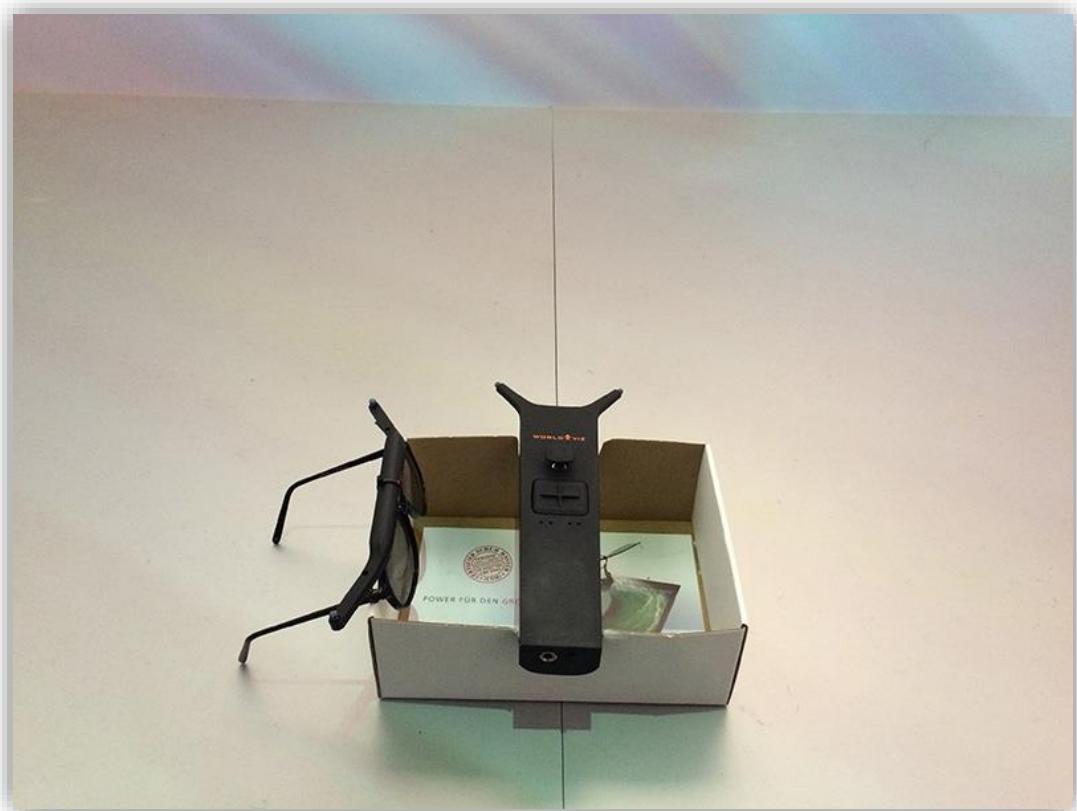
6. Konfiguration mit dem Prefix „PPT“ laden.

Ordner

D:\PPTStudioConfigs



7. Nun erfolgt das Einschalten der Tracking-Devices. Bitte die Eyes und den Wand in den CAVE legen (in die dafür vorgesehene Box) und zwingend die untenstehende Reihenfolge einhalten.



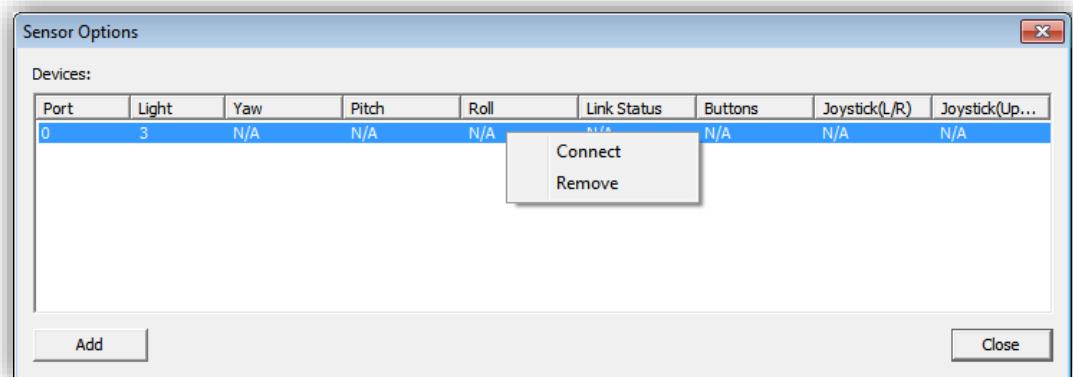
- a. Wand einschalten. Dazu den Schalter unten am Gerät nach unten betätigen, so dass nur der linke Marker aufleuchtet.



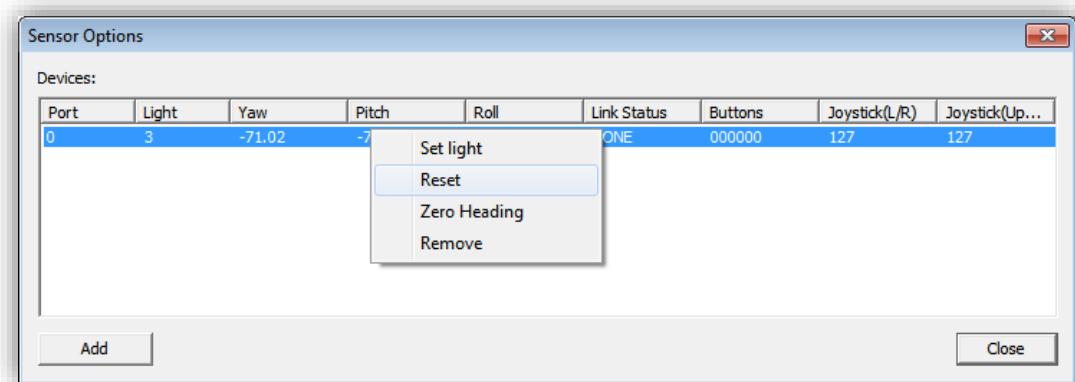
b. Eyes einschalten.



c. Den Wand im PPT Studio 2013 verbinden. Dazu oben links bei „Post-Process“ auf „PPT Wand“ klicken und beim erschienen Popup mit Rechtsklick die Verbindung aufbauen.



- d. Wand kalibrieren. Sicherstellen, dass der Wand in der dafür vorgesehenen Box liegt und mit Rechtsklick die Kalibrierung vornehmen. Dazu auf "Reset" klicken und das Popup bestätigen.



8. Nun sind die Devices und das Trackingsystem einsatzbereit. Für weitere und detailliertere Informationen zu WorldViz bitte das entsprechende Handbuch konsultieren.

4 Applikation starten

Im letzten Schritt muss die Applikation bloss noch gestartet werden. Die an den Unity Server angeschlossene Tastatur und Maus nehmen, zu der exportieren Applikation navigieren und die EXE-Datei ausführen.

Viel Spass!

F
H

Benutzerhandbuch WorldViz & CAVE

Konfiguration

Daniel Inversini
Julien Villiger

V1.00, 06.11.2015

Inhalt

Konfiguration	1
1 Einleitung	3
1.1 Kurzfassung	3
2 Systemvoraussetzungen	4
3 Benutzung	4
3.1 Starten der Kameras	4
3.2 Starten des WorldViz Studios	5
3.2.1 Verwenden des WorldViz über eine gespeicherte Konfiguration	6
3.2.2 Verwenden des WorldViz mit einer Neukonfiguration	6
3.3 Sequenzielles Einschalten der Komponenten	8
4 FAQ	8
5 Weitere Hinweise	9
5.1 Reseten des Wand	9
5.2 Rekalibrierung des CAVE	9
6 Anhang WorldViz Hilfe	9
6.1 Configuring WorldViz PPT system for wand use	10
6.1.1 Enabling the wand using a single light with Marker ID	10
6.1.2 Reset/Calibrate wand's virtual North	12
6.2 PPT Eyes Setup	13
6.2.1 Starting the Plugin	13
6.2.2 Configuring PPT Eyes	14
6.3 PPT Wand with PPT Eyes	16
6.3.1 Configure PPT Eyes	16
6.4 Calibrating	19
6.4.1 Standard calibration	19
6.4.2 Chained calibration	20
6.4.3 Clearing calibrations	21
7 Abbildungsverzeichnis	21
8 Versionskontrolle	22

1 Einleitung

Dieses Dokument dient dazu, die Software „PPT Studio 2013“ zu konfigurieren. Dazu gehört insbesondere:

- Das Booten der Komponenten
- Die Überprüfung der Kameras
- Die Überprüfung und Konfiguration der Plugins
- Das sequenzielle Einschalten der Geräte und deren Verwendung

sowie erweiterte Themen, namentlich:

- Das Kalibrieren des CAVEs
- Das Resetten des Wand-Devices

Weitere Schritte sollten Mithilfe der WorldViz Hilfe durchgeführt werden, dies ist nicht Teil dieser Dokumentation.

1.1 Kurzfassung

1. Starten des PPT Servers
2. Starten des DHCPs für die Kameras
3. Starten der Kameras
4. PPT Studio öffnen
5. Aktuelle Config laden
6. Wand mit einem LED starten
7. Wand verbinden über das Post Process Plugin
8. Wand resetten in der Resetbox am Boden des CAVEs
9. Eyes verbinden

2 Systemvoraussetzungen

3 Benutzung

3.1 Starten der Kameras

Verwenden Sie folgende Verknüpfungen:



Abbildung 1: Verknüpfungen Desktop

- 01_Server – Cameras
Startet einen DHCP-Server welcher IP's für die Kameras verteilt. Dies ist dann Sichtbar in der Ausgabe des Servers (Abbildung 3).
- 02_Power – Cameras
Öffnet einen Hyperlink zu einer gemanagten Steckerleiste. Dort können die Kameras auf „on“ geschalten werden.

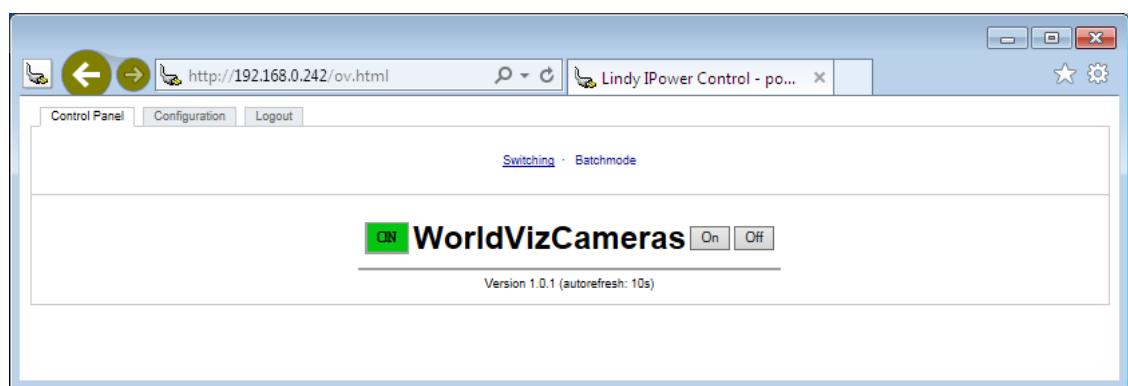


Abbildung 2: Ein-, Ausschalten der Kameras

Der DHCP-Server für die Infrarotkameras hat eine Kommandozeilenausgabe, welche bei erfolgreichem Starten wie folgt aussieht:

```
DA_01_Server - Cameras
s
Client 00:30:53:0e:20:8b (PPTH-0e208b) allotted 192.168.99.105 for 360000 second
s
Client 00:30:53:0d:ef:48 (PPTH-0def48) offered 192.168.99.101
Client 00:30:53:0d:ef:48 (PPTH-0def48) allotted 192.168.99.101 for 360000 second
s
Client 00:30:53:0e:20:88 (PPTH-0e2088) offered 192.168.99.104
Client 00:30:53:0e:20:8c (PPTH-0e208c) offered 192.168.99.107
Client 00:30:53:0e:20:88 (PPTH-0e2088) allotted 192.168.99.104 for 360000 second
s
Client 00:30:53:0e:20:8c (PPTH-0e208c) allotted 192.168.99.107 for 360000 second
s
Client 00:30:53:0d:e9:ad (PPTH-0de9ad) offered 192.168.99.102
Client 00:30:53:0d:e9:ad (PPTH-0de9ad) allotted 192.168.99.102 for 360000 second
s
Client 00:30:53:0d:ef:4f (PPTH-0def4f) offered 192.168.99.108
Client 00:30:53:0f:43:16 (PPTH-0f4316) offered 192.168.99.100
Client 00:30:53:0e:20:89 (PPTH-0e2089) offered 192.168.99.106
Client 00:30:53:0d:ef:4f (PPTH-0def4f) allotted 192.168.99.108 for 360000 second
s
Client 00:30:53:0f:43:16 (PPTH-0f4316) allotted 192.168.99.100 for 360000 second
s
Client 00:30:53:0e:20:89 (PPTH-0e2089) allotted 192.168.99.106 for 360000 second
s
```

Abbildung 3: Ausgabe DHPC-Server Kameras

3.2 Starten des WorldViz Studios

Verwenden Sie folgende Verknüpfungen:



Abbildung 4: Verknüpfungen WorldViz

Falls eine Warnung des Windows UAC (User Account Control) kommt, bitte bestätigen Sie mit OK.

3.2.1 Verwenden des WorldViz über eine gespeicherte Konfiguration

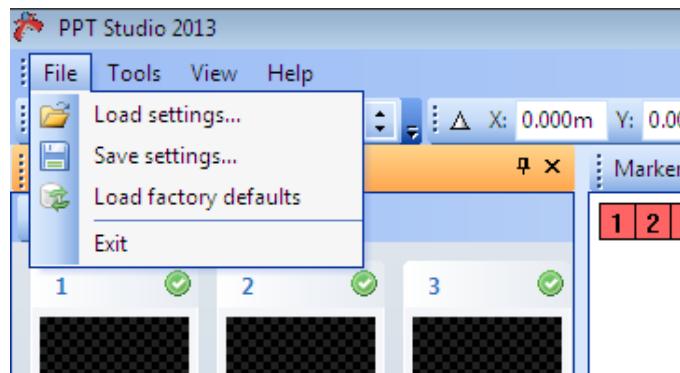


Abbildung 5: Konfiguration laden

Wählen sie im Dialogfenster dann die gewünschte Konfiguration. Damit können Sie bereits das WorldViz Studio verwenden.

Wir empfehlen jeweils die neuste stable-Version der vorhandenen Konfigurationen zu verwenden. Diese sind Identifizierbar mit Prefix «PPT».

3.2.2 Verwenden des WorldViz mit einer Neukonfiguration

Achtung, folgender Abschnitt empfiehlt sich nur für erfahrene Benutzer.

Falls Sie das WorldViz Studio neu konfigurieren möchten, benötigen Sie folgende Plugins:

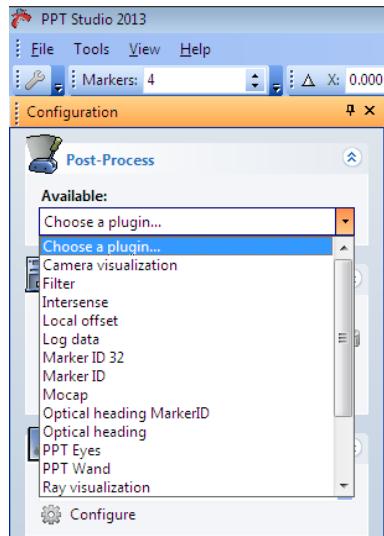


Abbildung 6: WorldViz Post-Process Plugins

- Marker ID
- PPT Wand
- PPT Eyes

Sowie für den Output:

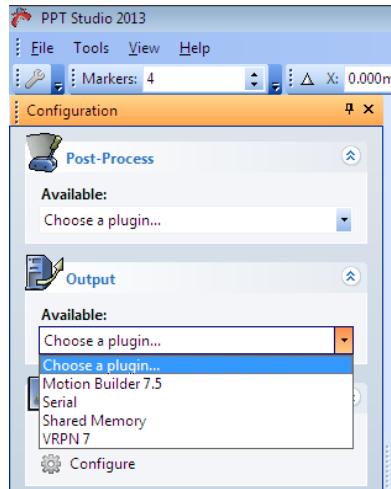


Abbildung 7: WorldViz Output-Plugins

- VRPN 7

Führen Sie die Konfiguration der Plugins anhand der WorldViz Hilfe durch. (Siehe Anhang WorldViz Hilfe).

3.3 Sequenzielles Einschalten der Komponenten

Komponenten in dieser Reihenfolge einschalten:

1. PPT Wand mit einem Marker einschalten (Schalter nach unten)
2. PPT Wand verbinden

Verbinden Sie diesen über das Popupmenu auf dem PPT Wand Plugin.

3. PPT Eyes einschalten
4. PPT Wand kalibrieren

Verwenden Sie dazu die Kalibrierungsbox und platzieren Sie diese in der Mitte des CAVEs am Boden, Ausrichtung nach vorne.

4 FAQ

Q: Wie muss ich mich auf dem TS1 einloggen?

A: Wenden Sie sich an einen Verantwortlichen des CPVR-Labs.

Q: Die Kameras verbinden sich nicht mit dem DHCP-Server

A: Führen Sie ein „ipconfig -flushdns“ durch oder rufen Sie www.google.com auf.

Q: Der Wand funktioniert nicht korrekt.

A: Vergewissern Sie sich, dass Sie den Wand auf 1 LED eingestellt haben.



Abbildung 8: PPT Wand 1, 2 LED's

Q: Die Eyes funktionieren nicht :

A: Vergewissern Sie sich, dass diese eingeschalten sind.



Abbildung 9: PPT Eyes

5 Weitere Hinweise

5.1 Reseten des Wand

Öffnen Sie das Wand-Plugin im PPT-Studio und reseten Sie den Wand über das Kontextmenü:

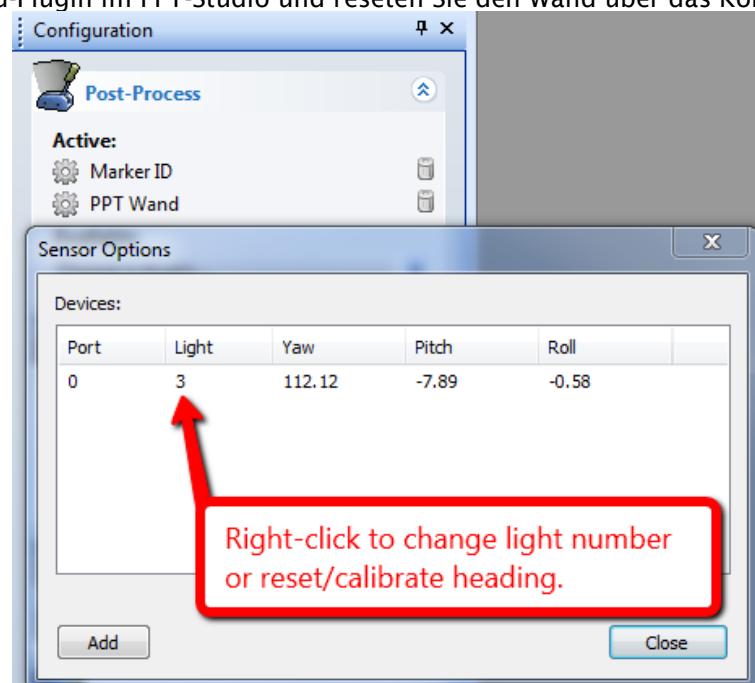


Abbildung 10: WorldViz Reset Wand

5.2 Rekalibrierung des CAVE

Sehen Sie dazu Kapitel 6.4.

6 Anhang WorldViz Hilfe

Folgende Anhänge sind Copyright bei Worldviz. Sie wurden aus der Hilfedatei extrahiert um eine möglichst vollständige Dokumentation für den CAVE der BFH bereitzustellen.

6.1 Configuring WorldViz PPT system for wand use

6.1.1 Enabling the wand using a single light with Marker ID

1. Turn the wand on (1 light mode is switch down position) before starting PPT Studio (because PPT Studio automatically attempts to connect to the wand if it was last used with a wand connected)
2. Place the wand on a stable, non-metallic surface with joystick pointing up
3. In the Configuration pane, add “Marker ID” under Post-Process options if it’s not already added using the dropdown menu. This plug-in must be topmost in your list of plug-ins (drag to reorder if necessary). If you have a factory configured wand, its Marker ID is 3 for single light use (right hand).
4. Click on Post- Process / Marker ID and uncheck “Automatically search inventory” if it is currently selected. Instead, put a checkmark for ID 3 under the list of physical IDs and do not change the virtual ID value. NOTE: If you’re also tracking other markers in your scene, ie. PPT Eyes, you must now use the Marker ID plug-in to establish the number and ID of the other markers. See your PPT manual for details about Marker ID. Hit “OK” when you complete the selections.

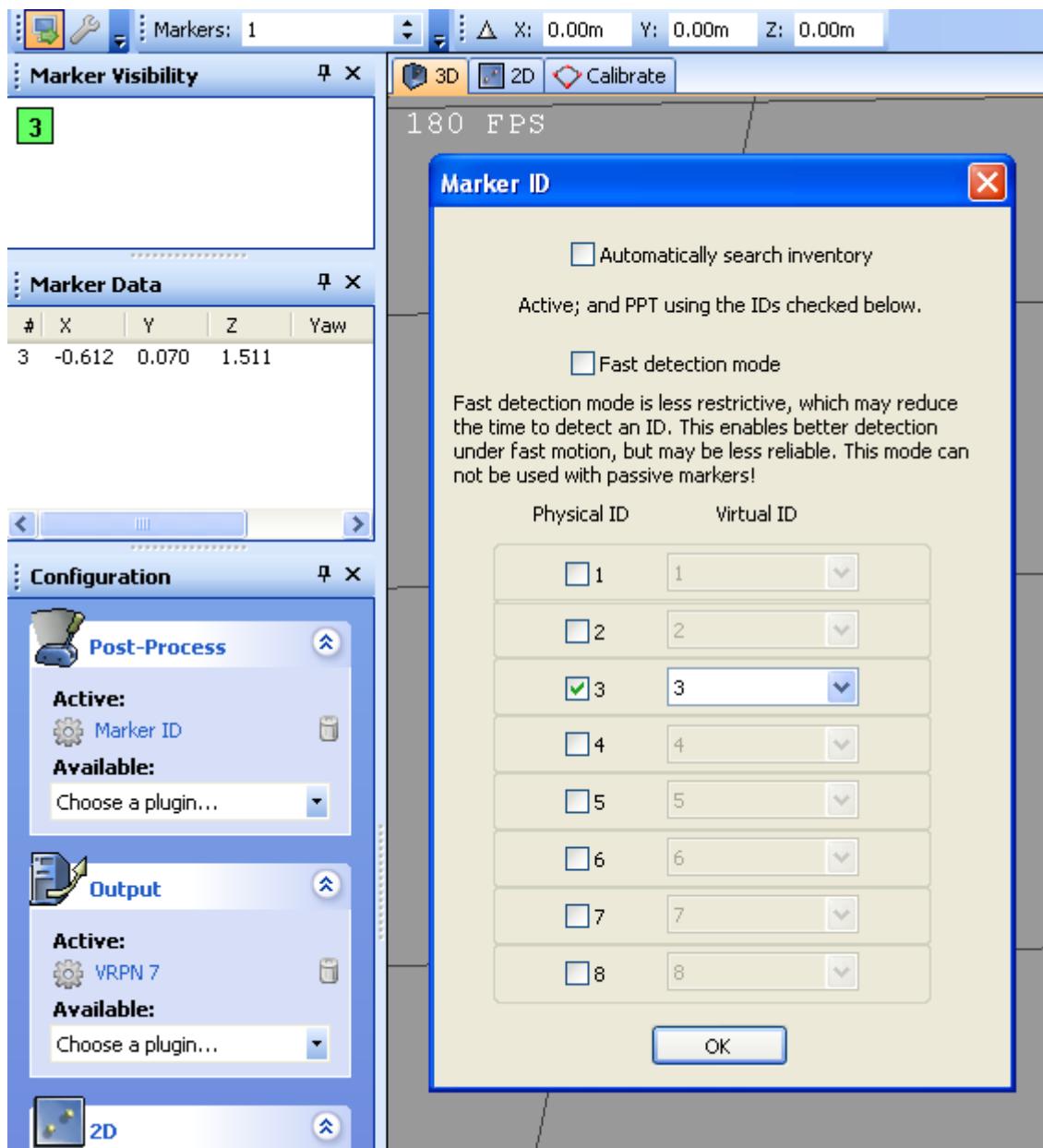


Abbildung 11: WorldViz: Wand 1

5. In the Configuration pane, add “PPT Wand” under Post-Process options if it’s not already added using the dropdown menu. It might take few seconds to load. This plug-in must be beneath Marker ID in your list of plug-ins.

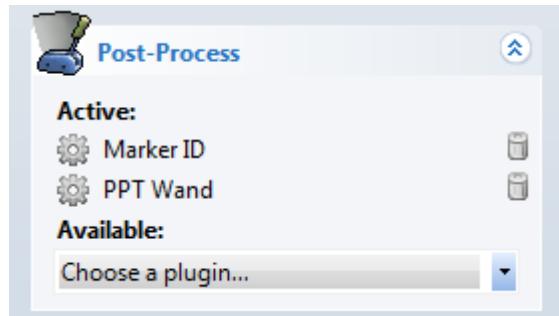


Abbildung 12: WorldViz Wand 2

6. Click on Post-Process / PPT Wand, and add an orientation sensor if none is presently added. To add a sensor, press the "Add" button and enter 0 (zero) for port number and hit "Add"; this will auto-detect the wand.
7. Configure the light number (in Post-Process / PPT Wand) by right-clicking the row showing the wand data. Under normal circumstances, this will be 3 (the marker ID of your wand). Hit "OK".
8. The current screen should be similar to the one shown below and finally hit "Close" to complete the PPT Wand configuration.

NOTE: For magnetic compensation (2 light) mode of the PPT Wand, please refer to section "[Compensation for magnetic distortion](#)" for further details.

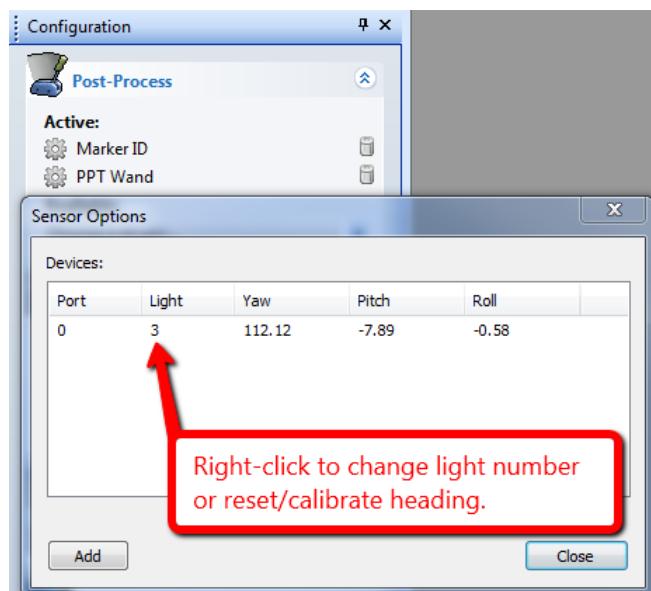


Abbildung 13: WorldViz Wand 3

6.1.2 Reset/Calibrate wand's virtual North

1. Since the wand uses a magnetic sensor in the standard 1 light mode, you need to reset the straight ahead or North direction.
2. Click on Post-Process / PPT Wand
3. Right-click on the row showing the Wand data and select "Reset"
4. Follow the Reset Wizard to completely calibrate your wand (Normally, you want to have your wand's LED end pointing to the Z direction or to the screen)
5. Hit "Close." Your calibration is now complete and is stored; the next time you run PPT you do not need to reset the wand

6.2 PPT Eyes Setup

The PPT Eyes is part of the Worldviz PPT product family and integrates into any PPT system. This page describes the setup of PPT Eyes.

6.2.1 Starting the Plugin

1. Start PPT Studio.
2. Setup PPT and calibrate the tracking system according to the instructions given in the PPT Studio Help.
3. Set the number of tracked markers to at least 2 and turn on the PPT Eyes device. You should now see two tracked lights in the 3D view. You will need to increase this value to be larger than 2 if you are tracking other objects in your environment.

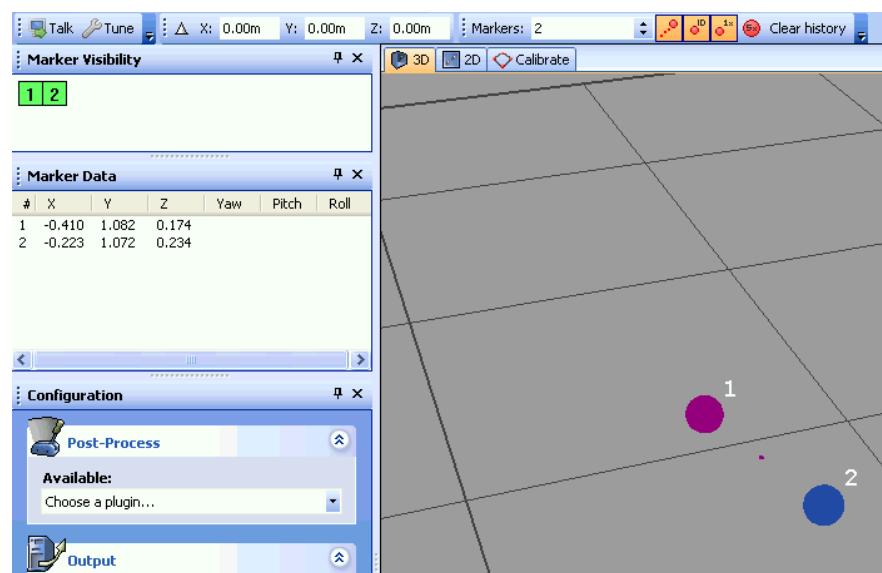


Abbildung 14: WorldViz Eyes 1

- Next, enable the PPT Eyes plugin by choosing it from the Post-Process list. PPT Eyes will now automatically detect the two tracked lights and merge them into a single traced light with orientation data. Check the Marker Visibility panel that this is so. The 3D view should also show a single tracked light with an axis to indicate orientation data.

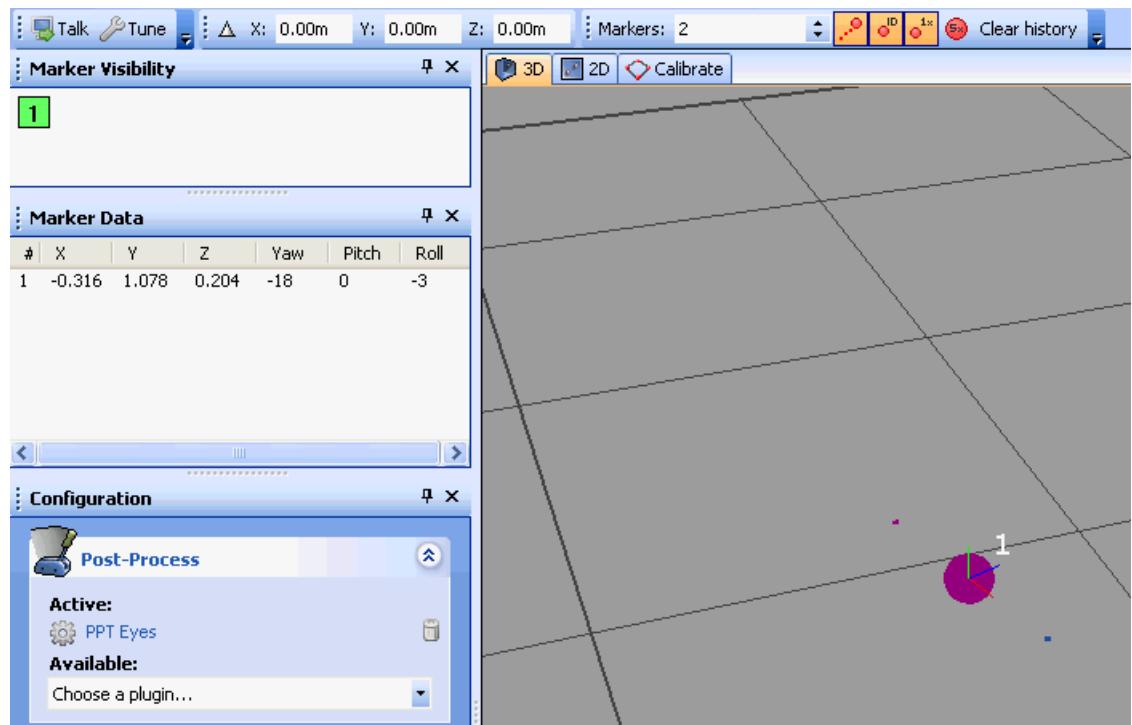


Abbildung 15: WorldViz Eyes 2

- PPT Eyes will flip the orientation by 180 degrees if the plugin determines that the user is facing away from the display. Because of this you may see an error warning in the Messages panel. This is normal operation. If you find that the orientation is incorrect, face towards PPT north and the orientation will correct itself.



Abbildung 16: WorldViz Eyes 3

6.2.2 Configuring PPT Eyes

To configure PPT Eyes, click on the PPT Eyes plugin in the active list. This will bring up the Settings dialog.

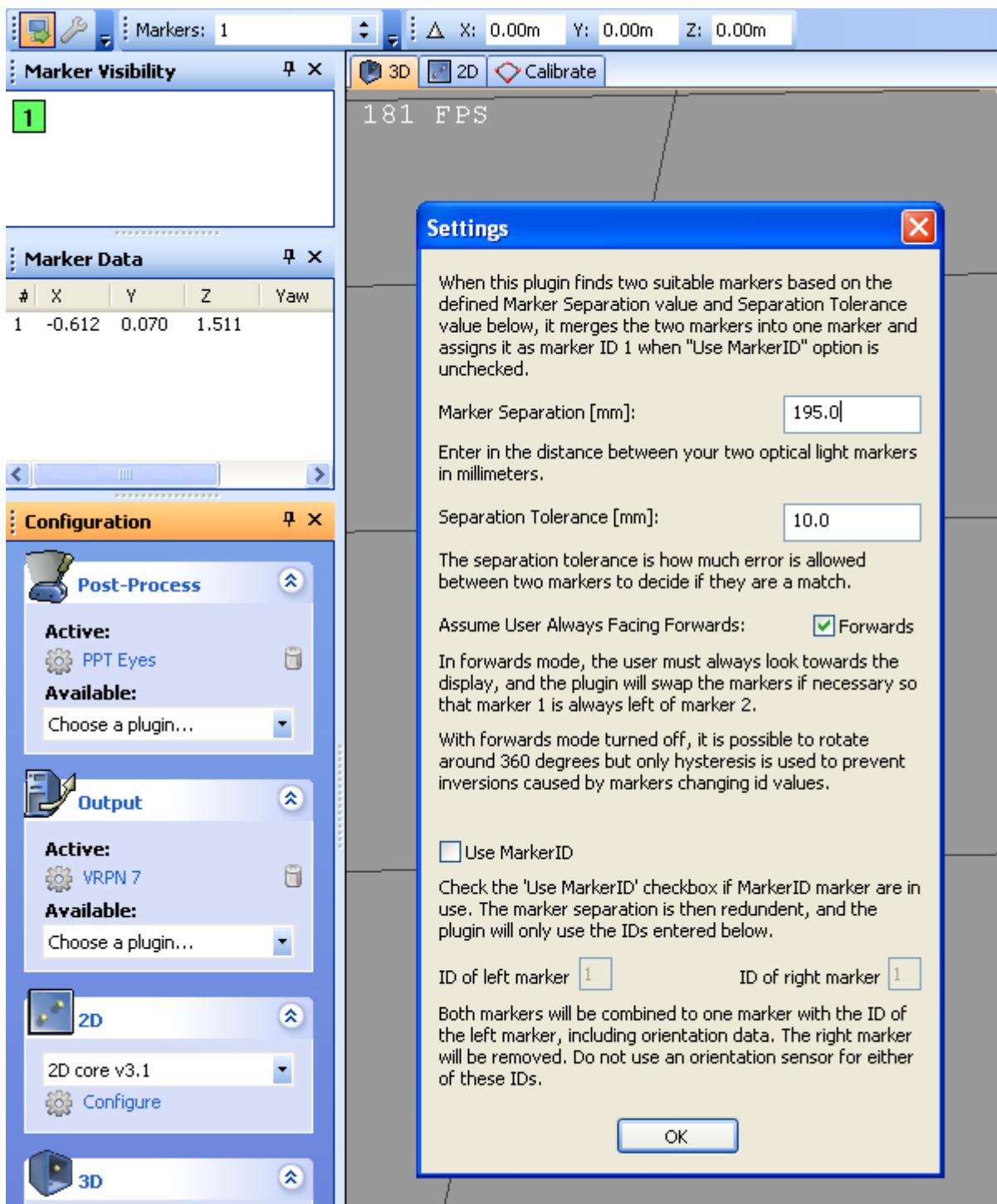


Abbildung 17: WorldViz Eyes 4

Marker Separation: This is the distance between the two LED lights on the PPT Eyes device. By default this should be 195 mm. *Do not change this unless instructed to by Worldviz support, or if you have built your own custom LED device.*

Separation Tolerance: This is the maximum absolute difference between the observed distance of the two LED lights on the PPT Eyes device and the Marker Separation value can be. PPT Eyes will fail to detect the device if this tolerance is exceeded.

Assume User Always Facing Forwards: By default this mode should be turned off. The standard operating mode allows the user to rotate around and attempts to guess which direction PPT north is located. If you enable always facing forwards mode, it will be impossible to look more than 90 degrees from PPT north, and no previous information will be used to auto-detect the direction.

Use MarkerID: If you are using MarkerID-based markers, then you can skip the need to detect markers based on separation, and can use the IDs directly. This mode is only useful if you have built your own custom LED device, and should not be used with the standard PPT Eyes device.

6.3 PPT Wand with PPT Eyes

PPT Eyes is a device that provides position and orientation tracking of a user's head and is typically mounted onto a pair of 3D glasses for viewing a 3D projection screen. PPT Eyes is designed to work in conjunction with PPT Wand, and together they provide a rich head and hand interactive solution for CAVE and powerwall environments.

Using PPT Eyes in conjunction with PPT Wand is as simple as combining the configuration for the two stand-alone devices. Below the configuration technique for PPT Eyes is provided. If the Marker ID and PPT Wand plug-ins have not been set up, it is recommended to follow section 2.2 "Configuring WorldViz PPT system for wand use" at this stage. While PPT Eyes do not need to be configured first, its plug-in should always be moved to below the "Marker ID" and above the "PPT Wand" in the Post-Process stack (drag to reorder).

6.3.1 Configure PPT Eyes

1. Turn on PPT Eyes (slide the micro-switch on the back to the top)
2. Place PPT Eyes in the tracking field where both LED markers can be seen.
3. Correctly set PPT Studio number of markers (the Eyes count as 2 additional markers so adjust accordingly)
4. With standard PPT Eyes and Wand configuration, you would have 3 markers in total. Remember to select additional 2 marker IDs in the Marker ID plug-in. Normally, we uncheck the "automatically search inventory" and manually select physical ID 1, 2, and 3 for the PPT eyes and wand. Add "Marker ID" under Post-Process options if it's not already added. NOTE: It is not necessary to add Marker ID plug-in if you use PPT Eyes alone.
5. In the Configuration pane, add "PPT Eyes" under Post-Process options if it's not already added using the dropdown menu.

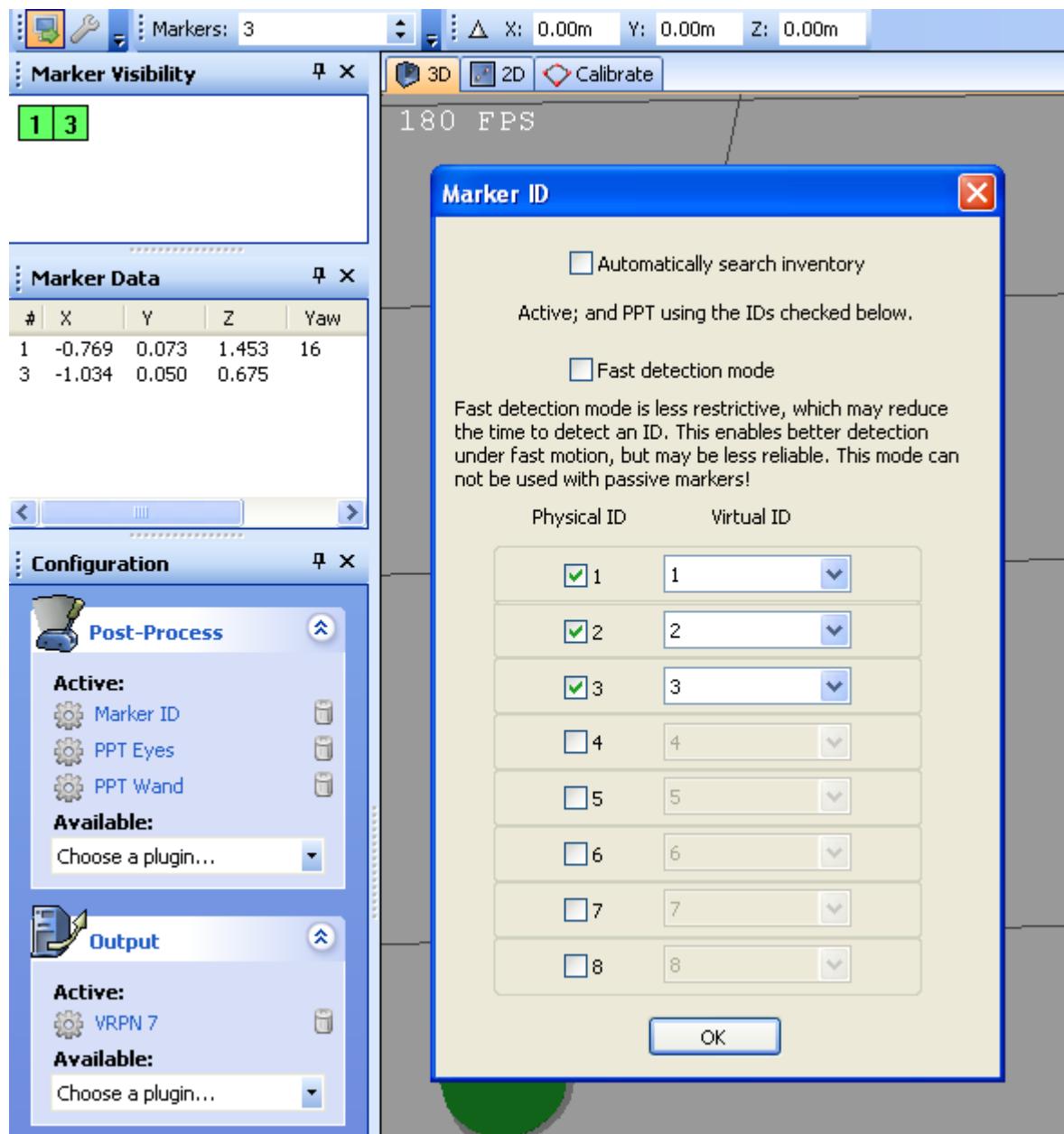


Abbildung 18: WorldViz Wand & Eyes 1

- Drag to re-order PPT Eyes so that it is below "Mark ID" and above "PPT Wand" in the Post-Process.



Abbildung 19: WorldViz Wand & Eyes 2

7. You should not need to configure the PPT Eyes plug-in as its default settings are correct for nearly all uses. The default value is shown below.
8. You should now see orientation data shown for marker ID # 1. This is the ID data computed from the PPT Eyes' two LED markers.

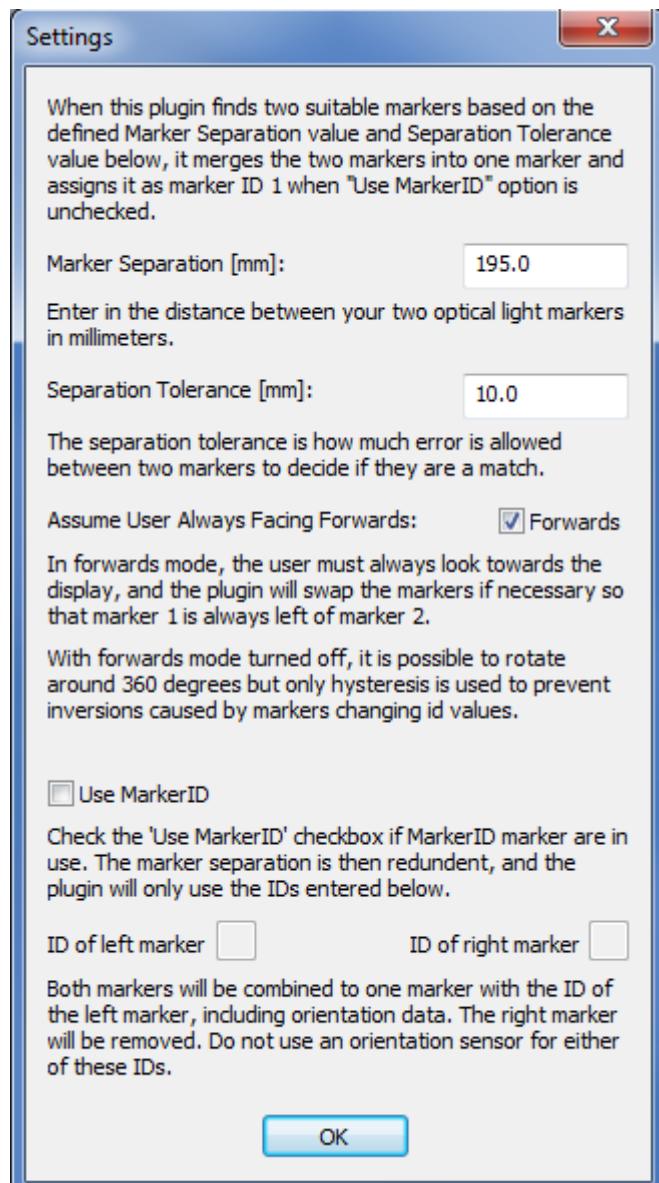


Abbildung 20: WorldViz Wand & Eyes 3

6.4 Calibrating

NOTE for PPT-X users:

Before running the calibration wizard, you need to be absolutely certain that your cameras are connected in the correct order. To verify this, wave your hand or a marker in front of each camera and double-check that each camera's physically labeled number matches its number in the PPT software interface. A mismatch cannot be detected by the software and calibration will be possible but the calibration quality will be significantly deteriorated. PPT-E and PPT-H systems automatically identify themselves and ensure correct connections.



Abbildung 21: WorldViz Kalibrierungsrig

6.4.1 Standard calibration

In this method, all cameras are calibrated simultaneously.

1. Before continuing, ensure that you have configured your PPT cameras to optimal sensitivity and thresholds, and that your workspace lighting is ready for data collection (e.g., outside windows blocked, warm lights off).
2. Turn off all of your PPT markers.
3. Turn on the PPT calibration rig and place it in the center of your workspace. Orient the calibration rig so that the +X and +Z axis markers are aligned in the directions that you desire for PPT's coordinate system. PPT north is defined as the direction of the +Z axis.
4. Try to keep the rig as close to the center of each camera's field-of-view as possible. Avoid placing the calibration rig markers at extreme edges of a camera's view. Use the Cameras panel to view all of the cameras simultaneously.
5. Click the Calibrate tab in the main viewport. This will launch the calibration wizard.

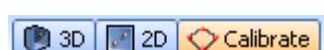


Abbildung 22: WorldViz Kalibrierungstab PPT Studio

- Find the Calibration Rig Size options and select the proper calibration rig size. The size is defined by the distance between the adjacent IR LEDs. In general, a system usually comes with a calibration rig which has the default size (57cm).

Calibration Rig Size: Default (57cm) Small (26cm) Large (90cm) Other

Abbildung 23: WorldViz PPT Studio Kalibrierungseinstellungen

- For a standard calibration, all cameras should be reset to Uncalibrated (as indicated by the red icon next to each camera). If this is not the case, click the Reset button before proceeding.



Abbildung 24: WorldViz PPT Studio Kalibrierungseinstellungen Camera 1

- Click the Calibrate button at the top of the window. Each of the camera's four indicator lights will turn green for each flash of the PPT calibration rig. If any camera fails to light up all green, then there was a problem with that camera seeing all four markers of the PPT calibration rig. Use the Cameras panel to re-examine that camera.
- If all cameras calibrated successfully, you'll receive a quality score. Good scores are in the range of 95-100; scores greater than 90 are still acceptable.



Abbildung 25: WorldViz PPT Studio Kalibrierungseinstellungen Camera 2

6.4.2 Chained calibration

In this method, cameras are calibrated in stages. Use this method for a physical workspace where not all cameras can see the calibration rig simultaneously (e.g., the room is L-shaped).

- Follow steps 1 - 4 above.
- If you're starting a new chained calibration, click the Reset button to clear all previous data.
- Click the Calibrate button at the top of the window. Each of the camera's four indicator lights will turn green for each flash of the PPT calibration rig.
- Cameras that are fully calibrated are now indicated with the green checkmark icon. Cameras that show a yellow icon indicate that the camera saw all four makers on the calibration rig but it cannot yet chain due to lack of data from a neighboring (connecting) camera. Cameras that show a red icon indicate cameras that saw less

than four markers on the calibration rig.

5. Move the calibration rig enough so that some or all those cameras indicated as red can now fully see the calibration rig.
6. Once all cameras are calibrated, you'll receive a quality score. Good scores are in the range of 95-100; scores greater than 90 are still acceptable.

NOTE about chaining

Use as few calibration snapshots as possible, typically this is done by "sweeping" the calibration rig from one end of the space to the other. If you suspect a camera calibrated but may contain poor measurements (e.g., the rig was at the extreme edge of the camera's view), you can easily right click on the camera to mark it as uncalibrated, and then redo it.

6.4.3 Clearing calibrations

From either the Cameras panel or the 2D view, you can right-click a camera and select Clear calibration to void a particular camera's calibration, forcing the PPT system to re-calibrate it during the next calibration. You can also clear all or some cameras directly in the Calibration guide.

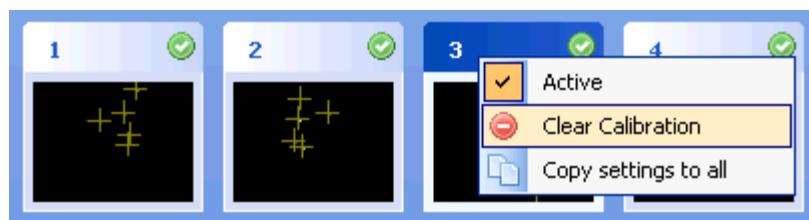


Abbildung 26: WorldViz PPT Studio Kalibrierung löschen

7 Abbildungsverzeichnis

Abbildung 1: Verknüpfungen Desktop	4
Abbildung 2: Ein-, Ausschalten der Kameras	4
Abbildung 3: Ausgabe DHPC-Server Kameras	5
Abbildung 4: Verknüpfungen WorldViz	5
Abbildung 5: Konfiguration laden	6
Abbildung 6: WorldViz Post-Process Plugins	6
Abbildung 7: WorldViz Output-Plugins	7
Abbildung 8: PPT Wand 1, 2 LED's	8
Abbildung 9: PPT Eyes	9
Abbildung 10: WorldViz Reset Wand	9
Abbildung 11: WorldViz: Wand 1	11
Abbildung 12: WorldViz Wand 2	12
Abbildung 13: WorldViz Wand 3	12

Abbildung 14: WorldViz Eyes 1	13
Abbildung 15: WorldViz Eyes 2	14
Abbildung 16: WorldViz Eyes 3	14
Abbildung 17: WorldViz Eyes 4	15
Abbildung 18: WorldViz Wand & Eyes 1	17
Abbildung 19: WorldViz Wand & Eyes 2	17
Abbildung 20: WorldViz Wand & Eyes 3	18
Abbildung 21: WorldViz Kalibrierungsrig	19
Abbildung 22: WorldViz Kalibrierungstab PPT Studio	19
Abbildung 23: WorldViz PPT Studio Kalibrierungseinstellungen	20
Abbildung 24: WorldViz PPT Studio Kalibrierungseinstellungen Camera 1	20
Abbildung 25: WorldViz PPT Studio Kalibrierungseinstellungen Camera 2	20
Abbildung 26: WorldViz PPT Studio Kalibrierung löschen	21

8 Versionskontrolle

Version	Datum	Beschreibung	Autor
1.0	06.11.2015	Dokument erstellt	Daniel Inversini



Erklärung der Diplandinnen und Diplanden

Déclaration des diplômant-e-s

Selbständige Arbeit / Travail autonome

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.

Name/Nom, Vorname/Prénom

Villiger Julien

Datum/Date

15.01.2016

Unterschrift/Signature

J. Villiger

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
Ce formulaire doit être joint au rapport de la thèse de bachelor.



Erklärung der Diplandinnen und Diplanden

Déclaration des diplômant-e-s

Selbständige Arbeit / Travail autonome

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.

Name/Nom, Vorname/Prénom

Ineschi Daniel

Datum/Date

15.01.2016

Unterschrift/Signature

Daniel

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
Ce formulaire doit être joint au rapport de la thèse de bachelor.



Unity 3D Server for CAVE Rendering

Pflichtenheft

Julien Villiger, Daniel Inversini
V2.0, 18.10.2015

Inhaltsverzeichnis

1	Allgemeines	3
1.1	Zweck dieses Dokumentes	3
1.2	Lesekreis	3
1.3	Ausgangslage	3
1.4	Unity	4
1.5	Nvidia Mosaic	4
1.6	Infrastruktur	4
1.6.1	Inputs und Tracking	5
1.6.2	Rendering	6
1.6.3	Darstellung im CAVE	7
2	Umfang und Ziele der Bachelor Thesis	8
2.1	Abgrenzungen	8
2.1.1	Technische Abgrenzungen	8
2.1.2	Weitere Abgrenzungen	8
2.2	Voraussetzungen und Ressourcen	8
3	Funktionale Anforderungen	9
3.1	Adaption Unity Anwendung für den CAVE	9
3.2	VRPN Unterstützung	9
3.3	Kompatibilität	9
3.4	WorldViz Tracking	10
3.4.1	Head Tracking	10
3.4.2	Wand Tracking	11
3.5	Einstellungsmöglichkeiten für 6DoF	11
3.6	Setup	11
3.7	Demoapplikation	12
3.8	Features	13
3.9	Zusätzliche Anforderungen	13
3.9.1	Unity-Output Warping	13
4	Nicht funktionale Anforderungen	14
4.1	Wiederverwendbarkeit	14
4.2	Ergonomie	14
4.3	Skalierbarkeit sekundäre Kamera	14
5	Testing	15
5.1	System Tests	15
5.2	Usability Tests	15
5.3	Szenarien	15
6	Administratives	15
6.1	Projektorganisation	15
6.1.1	Projektteam	15
6.1.2	Betreuer	15
6.1.3	Experte	15
6.2	Projektplan	16
6.3	Projektsitzungen	16
6.4	Meilensteine	16
6.4.1	Prototyp Unity Plugin	17
6.4.2	Tracking	17
6.4.3	Unity Plugin / Handbuch / Dokumentation	17
6.4.4	Präsentation	17
7	Versionskontrolle	18
8	Abbildungs- und Tabellenverzeichnis	19
9	Glossar	19

1 Allgemeines

1.1 Zweck dieses Dokumentes

Mit diesem Pflichtenheft wird der Rahmen, die Vorgehensweise und die Ziele der Bachelorthesis dokumentiert.

1.2 Lesekreis

Der Inhalt dieses Dokumentes richtet sich in erster Linie an die Betreuer und Experten dieser Arbeit, Prof. Urs Künzler und Herrn Harald Studer, die BFH-TI Abteilung CPVR¹ und an die Studenten, welche diese Projektarbeit durchführen.

1.3 Ausgangslage

Das cpvrLab besitzt eine CAVE² (Cave Automatic Virtual Environment) Installation mit dem virtuelle 3D-Welten in Echtgrösse über drei Projektionswände und eine Bodenprojektion erzeugt werden können. Alle Projektionsflächen werden dabei mit zwei Projektoren stereoskopisch projiziert, sodass eine nahezu perfekte Raumwahrnehmung entsteht. Die Entwicklung von virtuellen 3D-Welten mit Basis-APIs wie OpenGL³ oder OpenSceneGraph⁴ ist nach wie vor eine zeitraubende und aufwendige Arbeit und jedes Mal eine Einzelentwicklung. Es liegt deshalb nahe, eine High-Level Game Engine einzusetzen, mit der die Entwicklungszyklen vereinfacht und verkürzt werden können. Unity hat sich in den letzten Jahren in diesem Bereich durchgesetzt und ermöglicht es, gratis Spiele zu entwickeln.

Im Rahmen des vorgängigen Projekts, die Projekt 2 Arbeit, wurden verschiedene Methoden geprüft wie eine Integration von Unity in den CAVE erfolgen kann.

In einem ersten Schritt erfolgte eine Prüfung der bereits verwendeten Frameworks (Equalizer⁵, Chromium⁶) und der Software MiddleVR, die den Einsatz von Unity in einem CAVE deutlich vereinfachen sollte. Die dadurch entstehende Abhängigkeit der Software, die Anforderungen an die Unity⁷-Applikationen (etliche Adaptionen am Code waren jeweils notwendig) und die nicht gebrauchte Fülle an Features waren der Grund, wieso eine eigene Umsetzung eines Unity-Plugins erfolgen sollte.

Bezüglich der Infrastruktur wurden folgende Methoden analysiert und bewertet:

1. Mehrere Hosts / Mehrere GPUs / Mehrere Unity Instanzen
2. Ein Host / Mehrere GPUs / Mehrere Unity Instanzen
3. Ein Host / Mehrere GPUs / Eine Unity Instanz
4. Ein Host / Mehrere GPUs / Eine Unity Instanz mit Mosaic⁸ Treiber

Basierend auf Prototypen der verschiedenen Systeme und theoretischer Abklärungen konnte sich Variante 4 (Ein Host / Mehrere GPUs / Eine Unity Instanz mit Mosaic Treiber) klar hervorheben. Die dadurch erreichte hohe Flexibilität, die obsolete Synchronisierung und die gute Performance durch Verteilung der Last auf verschiedene GPUs waren ausschlaggebend.

¹ CVPR - <https://www.cpvrlab.ti.bfh.ch/>

² CAVE - https://de.wikipedia.org/wiki/Cave_Automatic_Virtual_Environment

³ OpenGL - <https://www.opengl.org/>

⁴ OpenSceneGraph - <http://www.openscenegraph.org/>

⁵ Equalizer - <http://www.equalizergraphics.com/>

⁶ Chromium - <http://chromium.sourceforge.net/doc/index.html>

⁷ Unity - <https://unity3d.com/>

⁸ Nvidia Mosaic - <http://www.nvidia.com/object/nvidia-mosaic-technology.html>

1.4 Unity

Die Entwicklungsumgebung erlaubt die Entwicklung von Computerspielen und anderer interaktiver 3D und 2D-Grafik-Anwendungen. Die Umgebung läuft auf den Betriebssystemen Windows und OS X. Zielplattformen sind neben PCs auch Spielkonsolen, mobile Geräte und Webbrowser.

(Quelle: [https://de.wikipedia.org/wiki/Unity_\(Spiel-Engine\)](https://de.wikipedia.org/wiki/Unity_(Spiel-Engine)))



Abbildung 1: Unity Editor, Engine

1.5 Nvidia Mosaic

Mosaic ist eine Technologie von Nvidia, um mehrere Graphikkarten über den Treiber zu verlinken und auf einem Desktop darzustellen. Windows wird ein einzelner Output vorgegaukelt, auch wenn physisch mehrere GPUs mit multiplen Ausgängen angeschlossen sind.

„Die NVIDIA® Mosaic™ Mehrbildschirm-Technologie dient zur einfachen Skalierung jeder Anwendung auf mehrere Bildschirme, und das ohne Softwareanpassungen oder Leistungseinbußen. Durch die Mosaic Technologie werden Mehrbildschirm-Konfigurationen vom Betriebssystem als einzelner Bildschirm wahrgenommen.“

(Quelle: <http://www.nvidia.de/object/nvidia-mosaic-technology-de.html>)

1.6 Infrastruktur

Nach Abschluss der Projekt 2 Arbeit wurden basierend auf den gewonnenen Erkenntnissen und den gestellten Anforderungen die Infrastruktur überarbeitet und erweitert. Konkret beinhaltet das folgende Punkte:

- Ersetzen des Servers für das WorldViz Trackingsystem
- Beschaffung und Inbetriebnahme des Unity Servers
- Beschaffung eines Gamepads

Neu gestaltet sich die Infrastruktur wie folgt:

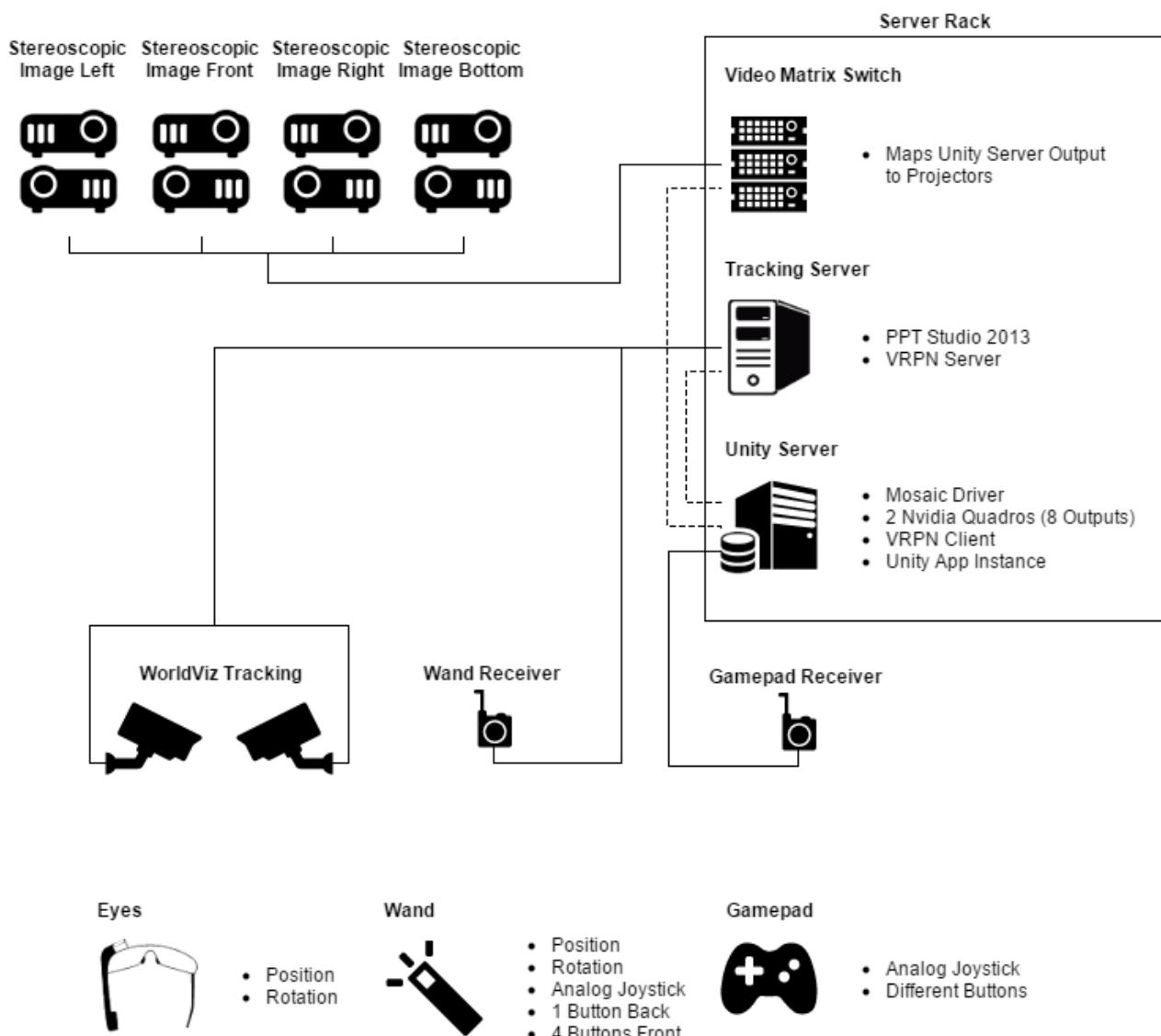


Abbildung 2: Infrastruktur CAVE

1.6.1 Inputs und Tracking

Die beiden Tracking Devices (Eyes und Wand⁹) werden durch 10 Infrarotkameras (WorldViz¹⁰ Tracking) geortet. Die Signale werden vom Tracking Server interpretiert und eine genaue Position und Rotation der beiden unabhängig voneinander bewegten Devices werden bestimmt. Auf diesem Server läuft die von WorldViz herausgegebene Software „PPT Studio 2013“, welche Einstellungen des Trackingsystems (Kalibrieren der Kameras, Überprüfen der Signale, Definition der Devices usw.) erlaubt. Zusätzlich wird, direkt im PPT Studio 2013 integriert, ein VRPN¹¹ Server betrieben, welcher die berechneten Werte der Tracking Devices über einen Socket ins Netz liefert.

Der Wand hat zusätzliche Inputs wie ein analoger Joystick, ein Button auf der Rückseite und 4 Buttons auf der Vorderseite. Diese Inputs werden per Funk an eine Station, den Wand Receiver, geschickt und an den Trackingserver weitergeleitet.

⁹ Eyes und Wand - <http://www.worldviz.com/products/ppt/wand-and-eyes>

¹⁰ WorldViz - <http://www.worldviz.com/>

¹¹ VRPN - <https://en.wikipedia.org/wiki/VRPN>



Abbildung 3: WorldViz Wand

Das Gamepad wird nicht durch das Tracking System geortet, sondern dient als Ersatz für standardmä-
sige Inputs wie Maus und Tastatur. Das Signal wird direkt an den Unity Server übermittelt.

1.6.2 Rendering

Auf dem Unity Server läuft die Unity Applikation und die über das VPRN Protokoll gelieferten Informati-
onen bezüglich der Tracking Devices werden, mit Hilfe des entwickelten Unity Plugins, interpretiert.
Dank des Mosaic Treibers von Nvidia wird das Output-Signal, welches von der Unity Applikation geren-
dert wird, auf die 8 vorhandenen Grafikkarten Outputs verteilt und an den Video Matrix Switch überlie-
fert.

1.6.3 Darstellung im CAVE

Als letzten Schritt empfangen die Projektoren das Signal über den konfigurierten Matrix Switch und stellen eine stereoskopische Ansicht sicher, indem jeweils zwei Projektoren leicht versetzte Bildinformationen erhalten und zusammen eine Leinwand beleuchten. Mittels Polfilter an den Projektoren und einer Polfilterbrille nimmt der Benutzer ein 3D-Bild wahr.

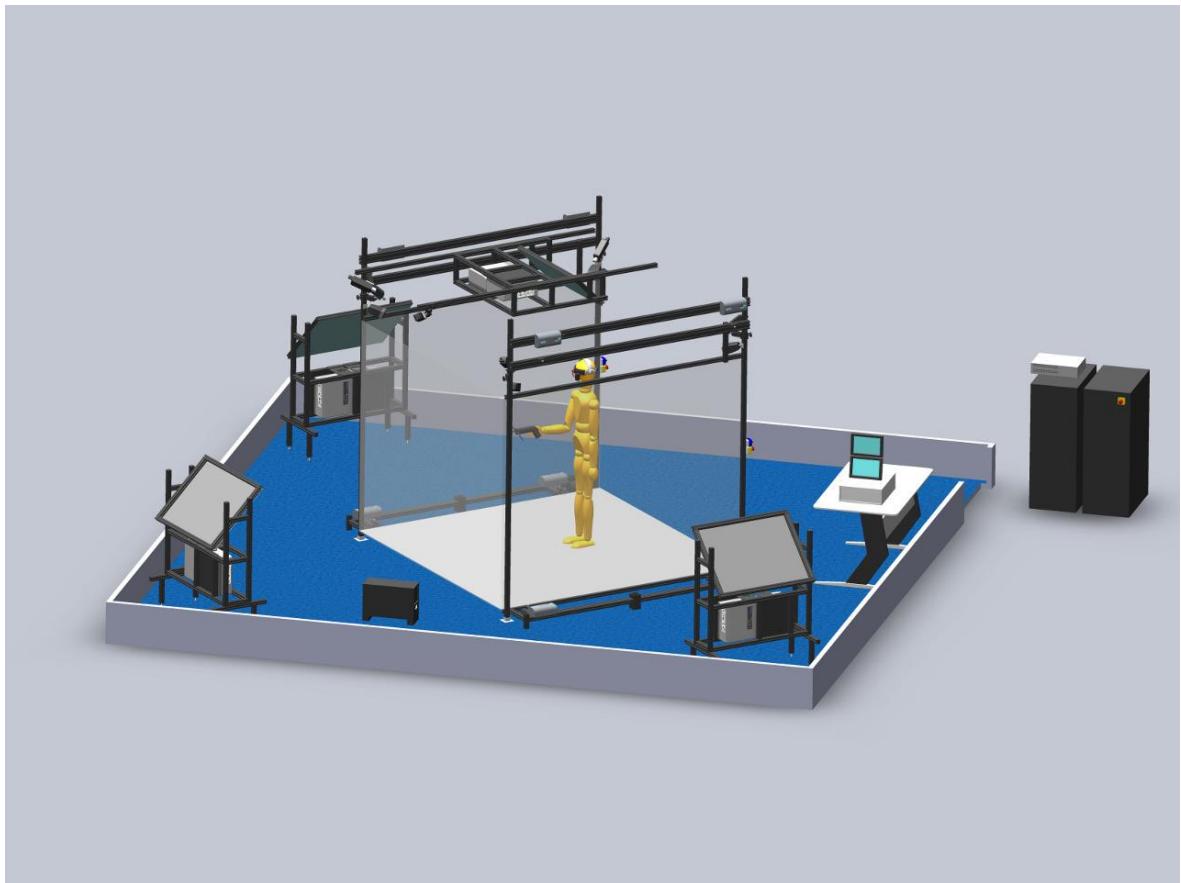


Abbildung 4: CAVE BFH

2 Umfang und Ziele der Bachelor Thesis

Folgende vier Schwerpunkte bestimmen den Umfang der gesamten Thesis:

1. Unabhängiges Unity Plugin, welches die Konfiguration für den CAVE ermöglicht. Hier muss besonders auf die Konfiguration des Nvidia-Mosaic Rücksicht genommen werden.
2. Einbindung der bereits installierten Motion-Tracking Lösung von WorldViz ins Unity (über das oben genannte VRPN-Plugin).
3. Implementierung einer/mehreren Unity-Demoapplikationen, welche die Fähigkeiten dieser Installation demonstriert.
4. Erstellen eines Handbuchs sowie ein Tutorial für kommende Unity Anwendungen (der Aufwand für die Portierung in den CAVE sollte extrem einfach und minimal gehalten werden).

Die Zieldefinition ergibt sich direkt aus diesen Anforderungen:

Es muss möglich sein, mit geringem Aufwand eine gängige Unity-Applikation in kurzer Zeit im CAVE der BFH zu verwenden.

Die neue und bereits vorhandene Infrastruktur soll benutzerfreundlich und ansprechend in das Projekt eingebunden werden.

2.1 Abgrenzungen

2.1.1 Technische Abgrenzungen

Um eine möglichst saubere, wartbare und moderne Applikation anbieten zu können, wird nach Möglichkeit nur auf Unity, respektive C# gesetzt. Low-Level Implementationen in C, C++ (auch FreeGlut) werden keine vorgenommen.

2.1.2 Weitere Abgrenzungen

Als Hardwarekonfiguration des Unityrechners wird die Empfehlung aus dem Projekt 2 umgesetzt / übernommen. Die WorldViz Installation findet, ausgenommen der neuen Servern, wie anhin Verwendung. Es sind höchstens Kalibrierungsarbeiten oder Softwarekonfigurationen vorgesehen.

2.2 Voraussetzungen und Ressourcen

Die Voraussetzungen wurden durch das Projekt 2 abgeklärt und bei Bedarf beschafft und installiert. Desweitern muss der Zutritt zu den Räumlichkeiten der BFH, wo sich die Installation des CAVEs befindet, sichergestellt werden.

3 Funktionale Anforderungen

Folgende Tabelle dient als Übersicht, die einzelnen Punkte sind in den Unterkapiteln beschrieben:

	Priorisierung	Keyfeatures	Kritische Punkte
Adaption Unity Anwendung für den CAVE	1	<ul style="list-style-type: none"> - Automatische Erstellung der zusätzlichen Unitykameras - Rotierung der Unitykameras - Justierung des Frustums und FOV 	<ul style="list-style-type: none"> - Optimale Frustums, welches die genaue Position im CAVE als Unitykamera übernehmen.
VRPN Unterstützung	1	<ul style="list-style-type: none"> - Implementierung des Protokolls für VR-Eingabegeräte 	<ul style="list-style-type: none"> - VRPN Server von WorldViz ist eine Blackbox (Gerätenamen)
Kompatibilität	2	<ul style="list-style-type: none"> - Aktuelle Unityversion und zukünftige sollen supportet werden 	<ul style="list-style-type: none"> - Abhängig von Unity
Tracking	1	<ul style="list-style-type: none"> - Erkennung der Positionen und Rotationen von Kopf und WAND im CAVE 	<ul style="list-style-type: none"> - Abhängig von den vorhandenen Kamerassen sowie deren Bildqualität
Einstellungs-möglichkeiten	2	<ul style="list-style-type: none"> - Alle Freiheitsgrade müssen pro Anwendung einstellbar oder direkt sperrbar sein 	
Setup, Dokumentation	2	<ul style="list-style-type: none"> - Plug & Play für Unityanwendungen 	<ul style="list-style-type: none"> - Spezielle Anwendungen können eventuell nicht durchgängig unterstützt werden
Demoapplikationen	3	<ul style="list-style-type: none"> - Siehe Tabelle 2 	
Unity Output Warping	4	<ul style="list-style-type: none"> - Das Ausgabebild soll auf die Eigenheiten des CAVEs angepasst werden können 	<ul style="list-style-type: none"> - Optionales Feature

Tabelle 1: Übersicht Funktionale Anforderungen

3.1 Adaption Unity Anwendung für den CAVE

Beliebige Spiele, Simulationen oder sonstige Anwendungen die mit Unity umgesetzt wurden, sollen so manipuliert werden, dass auf sämtlichen Leinwänden des CAVEs eine stereoskopische Projektion dargestellt wird.

3.2 VRPN Unterstützung

VR-Geräte sollen über das VRPN Protokoll angesprochen werden können und ihre Inputs über das Plugin ins Unity übertragen werden können (WorldViz Wand und Eyes).

3.3 Kompatibilität

Sämtliche, quelloffene Unity Anwendungen ab Version 5.0 sollten mit dem umgesetzten System kompatibel sein. Der Export der Unity Anwendung muss für das spätere Einpflegen in den CAVE für Windows Desktop erfolgen.

Die Schnittstelle zum umgesetzten System kann mit verschiedenen Methoden erfolgen.

- **Dynamic Link Library¹² (.dll)**
Unabhängig von Managed und Native Plugins, können sämtliche Funktionen über eine kompilierte .dll erfolgen, die ins Projekt integriert werden muss. Diese Methode hätte den Vorteil, dass der Code nicht eingesehen und modifiziert werden kann. Code Completion wird dank der .dll gewährleistet.
- **Asset Store (Packages)**
Um das Integrieren einer Library zu vereinfachen, kann im Asset Store ein Package angeboten werden, welches direkt an den vorgesehenen Ort kopiert und mit dem Projekt verknüpft wird. Das Package kann unter anderem eine .dll oder offener Code beinhalten.
- **Source Code API¹³**
Die Schnittstelle kann über offenen Source Code erfolgen. Die entsprechenden Klassen werden ins Unity integriert und können bei Bedarf adaptiert werden. Maximale Flexibilität wird gewährleistet.

Die optimal passende Variante wird durch Prototypen evaluiert. Diese werden bewertet nach folgenden Kriterien:

- **Konfigurierbarkeit**
Können die nötigen Parameter (CAVE Abmessungen, 6DoF¹⁴-Einstellungen, und weiteres) gepflegt werden.
- **Plattformunabhängigkeit**
Werden alle gewünschten Betriebssysteme und Architekturen unterstützt.
- **Skalierbarkeit**
Können die Features (Tracking, 3D-Stereoskopie) getrennt verwaltet werden.

3.4 WorldViz Tracking

3.4.1 Head Tracking

Die Position und Rotation der Hauptkamera ist durch die Unity Anwendung gegeben und kann durch unterschiedliche Inputs (z.B. Maus, Tastatur, Gamepad) erfolgen.

Zusätzlich zu der von der Anwendung definierten Kamerabewegung erfolgt eine leichte Verschiebung und Drehung der Kamera durch das Infrarot Tracking. Diese Translation und Rotation ist aber nur eine minime Veränderung des Kopfes in Relation zur Änderung, die ohnehin von der Applikation gegeben ist.

Entscheidend ist, dass die zwei verschiedenen Inputs, Unity und WorldViz, klar getrennt werden. Sonst kann u.U. eine Situation entstehen, welche nicht definiert ist (Kombinationen der jeweiligen Positionen und Rotationen). Ansonsten wäre eine generische Lösung, die möglichst alle Applikationen abdeckt, unrealistisch.

Die Immersion wird durch diese Methode deutlich gesteigert, weil sich die Oberkörper-, bzw. Kopfbewegung in der virtuellen Welt genau gleich wie in der realen Welt verhält.

Folgende Inputs des Head Tracking Devices werden vom Unity Plugin interpretiert:

- **Position**
Verschiebt die Kameras um den Betrag, welcher von einem initialen Punkt aus erfolgte. Rein mit dem Head Tracking ist es nicht möglich, die Steuerung des Spiels (welche die Hauptkamera beeinflussen würde) zu übernehmen. Nur dieses Offset wird interpretiert.

¹² DLL - https://en.wikipedia.org/wiki/Dynamic-link_library

¹³ API - https://en.wikipedia.org/wiki/Application_programming_interface

¹⁴ 6DoF - https://en.wikipedia.org/wiki/Six_degrees_of_freedom

- **Rotation**

Je nach Ausrichtung des Kopfes, bzw. des Head Trackings, dreht sich auch die Kamera im Spiel.

3.4.2 Wand Tracking

Ein weiteres Input Device ist der Wand von WorldViz. Das Tracking dieses Gerätes bewirkt die Steuerung der Applikation, die anstelle einer Tastatur, der Maus oder des Gamepads erfolgen kann.

Folgende Inputs des Wand Tracking Devices werden vom Unity Plugin interpretiert und auf die Applicationslogik so weit wie möglich angewandt:

- **Rotation**

Die Rotation des Wands wird als Mausbewegung interpretiert.

- **Joystick**

Diese Eingabe simuliert das Drücken der Pfeiltasten (und gleichzeitig, wie in vielen Spielen üblich, W, A, S, D)

- **Buttons**

Die verschiedenen Buttons des Wands sind auf die meistüblichen Tastatureingaben abgebildet, die in einem Spiel benutzt werden. (Linke Maustaste, rechte Maustaste, Leertaste, Control, usw.) Die Konfiguration des Unity Plugins lässt aber eine neue Zuordnung zu, um der aktuellen Applikation zu entsprechen.

- **Position**

Da die WorldViz Installation über das VRPN Protokoll die Positionen und Rotationen aller erkannten und verfolgbaren Objekte übermittelt, kann alternativ statt des Head Trackings auch die Position des Wands verwendet werden.

3.5 Einstellungsmöglichkeiten für 6Dof

Je nach Anwendung muss die Sensibilität der Achsen angepasst werden können, damit die physischen Bewegungen im CAVE stärker oder schwächer interpretiert werden können. Zusätzlich können selektiv Achsen deaktiviert werden.

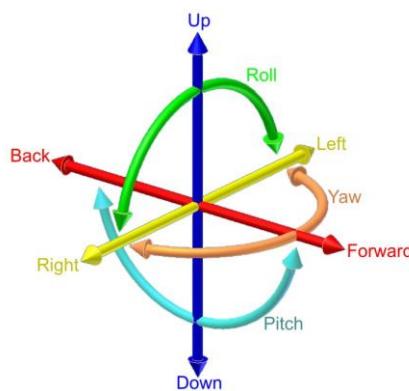


Abbildung 5: 6Dof

3.6 Setup

Damit Anwender, die nicht in das Projekt involviert waren, das Unity Plugin problemlos, schnell und einfach mit ihrer eigenen Unity Anwendung benutzen können, wird eine Schritt für Schritt Anleitung und ein Setup erstellt.

3.7 Demoapplikation

Um sämtliche umgesetzten Features und die Usability praktisch veranschaulichen zu können, werden Demoapplikationen erstellt, bzw. verwendet.

- **Schiessbude (Priorität 1)**

Diese eigens erstellte Applikation bietet optimale Voraussetzungen, um sämtliche Features des Unity Plugins veranschaulichen zu können.

Das Setting dieses Demospieles ist eine Schiessbude, wie sie auf einem Jahrmarkt anzutreffen ist. Die Galerien mit den abzuschiessenden Zielen verteilen sich jedoch rund um den Spieler. Mit Hilfe des Head Trackings kann sich der Spieler in der gesamten Szenerie umschauen, Bewegungen ausführen und die Objekte aus verschiedenen Perspektiven betrachten. Das Wand-Device steuert das Gewehr, um die Zielobjekte anzuvisieren und abzuschiessen. Die Buttons des Wands werden gebraucht um das Gewehr abzufeuern.

Spiellogik

Das Spiel startet unmittelbar auf dem Schiessgelände, welches im Wildwest-Setting spielt. Kreisförmig um den Spieler sind verschiedene Marktstände angeordnet, auf denen die abzuschiessenden Objekte erscheinen. Wird ein Zielobjekt mit dem Gewehr getroffen, werden dem Spieler Punkte gutgeschrieben, dessen Summe als GUI-Element in einer Ecke des Frontscreens sichtbar ist. Ein weiteres GUI-Element ist die Zeit, die seit dem Start des Spiels verstrichen ist.

Zielobjekte

Abzuschiessende Ziele erscheinen an fixierten Positionen, zu einem zufälligen Zeitpunkt und bleiben wiederum eine zufällige Zeitspanne sichtbar. Während dieser Zeit kann mit dem Gewehr das Objekt abgeschossen und Punkte gesammelt werden. Die Meshes der Ziele sind Gummienten und Zielscheiben.

WorldViz Wand

Der Wand positioniert das virtuelle Gewehr. Ein Fadenkreuz wird entsprechend der Position des Wands angezeigt und dient als Hilfe fürs Abschiessen der Objekte. Das Gewehr richtig sich möglichst genau gleich aus wie der Wand in der realen Welt. Der Button auf der Rückseite feuert das Gewehr ab. Der Joystick hat in dieser Applikation keine Funktion.

WorldViz Eyes

Damit sich der Spieler in der virtuellen Welt bewegen kann, wird die Position der Eyes aufgezeichnet und in einem definierten Radius ins Spiel übertragen. Die Rotation ist die Ausrichtung des Spielers, also in welche Richtung das Gewehr grundsätzlich zielt. Zu dieser Rotation kommt dann noch die Rotation, welche vom Wand definiert wird, hinzu.

Ziel

Das Spiel hat kein fixes Ziel, es geht primär darum, möglichst viele Punkte in kurzer Zeit zu sammeln.

- **Model-Viewer (Priorität 2)**

Es wird ein Operationssaal dargestellt, in dem sich der Benutzer im CAVE wie ein Arzt bewegen kann. Als Alternative zum Operationssaal könnten auch Gebäude, Städte und sonstige statische Modelle Verwendung finden. Mit Hilfe des Joysticks auf dem Wand-Device kann sich der Benutzer frei im virtuellen Raum bewegen. Leichte Bewegungen sind mittels Eyes-Device ebenfalls möglich. Spezielles Setting hier ist, dass bspw. ein Körperquerschnitt, MRI- oder Röntgenbild als sekundäre Kamera angezeigt wird, welche über das Plugin konfigurierbar ist. (Fix an einer Seite des CAVEs, fix positioniert in einer Ecke des Front-Screens oder sogar verschiebbar (durch die Wand)).

- **Demoapp Drittpartei (Priorität 3)**

Eine nicht spezifisch auf das Unity Plugin massgeschneiderte Applikation wird ausgewählt, um die Wiederverwendbarkeit und Kompatibilität des erstellten Unity Plugins zu demonstrieren. Möglicherweise können nicht alle Features, die das Plugin bieten würde, vom Spiel interpretiert werden, weil dies von der Spiellogik her nicht möglich ist.

Diese Applikation muss noch evaluiert werden, naheliegend ist das automatisch ausgelieferte Autorennspiel von Unity. Alternativ finden Demo-Apps aus dem Appstore Verwendung.

3.8 Features

Alle diese Applikationen stellen verschiedene Anforderungen an das Unity Plugin. Die Apps wurden so gewählt, dass eine möglichst grosse Abdeckung der Features erfolgt. Abhängig vom Projektfortschritt werden möglicherweise nicht alle Demoapps umgesetzt.

	Schiessbude (Priorität 1)	Demoapp Drittpartei (Priorität 2)	Statische Welt (Priorität 3)
Headtracking	X	X	X
Wand	X	?	X
Gamepad	X	?	
Vordefinierte Kamerasteuerung	X	X	
Vordefinierte Spiellogik	X	X	
Sekundäre Kameras		?	X

Tabelle 2: Features Unitygames

X: Vorhanden

? : Abhängig der Unity Applikation

3.9 Zusätzliche Anforderungen

Aufgrund erkannter Hardwareschwächen im CAVE (unterschiedliche Schärfen der Beamer, Beweglichkeit der Beamerlaufhängungen und Wärmeausdehnung bei Betrieb) wird folgendes als optionale Anforderung definiert:

3.9.1 Unity-Output Warping

Es soll eine Machbarkeitsstudie durchgeführt werden, ob der Output der Unity-Rendering Pipeline noch mit einem frei justierbaren Filter erweitert werden kann, der Unebenheiten ausgleicht.

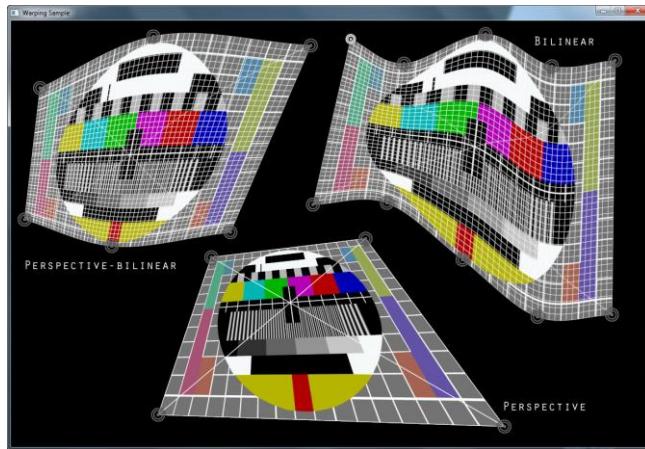


Abbildung 6: Image Wrapping (Quelle: github.com)

4 Nicht funktionale Anforderungen

4.1 Wiederverwendbarkeit

Damit zukünftige Entwickler effizient eigene Anwendungen in den CAVE einpflegen können, wird viel Wert auf die Wiederverwendbarkeit gelegt.

Anwender aus verschiedenen Bereichen wie Architektur, Autoindustrie, Game Development usw. können ihre Simulationen in den CAVE einpflegen und ausführen.

4.2 Ergonomie

Im Rahmen eines kleinen Tutorials wird Schritt für Schritt erklärt, wie die eigene Unity Anwendung für den CAVE aufbereitet werden kann.

4.3 Skalierbarkeit sekundäre Kamera

Ein Spiel kann eine sekundäre Kamera verwenden um beispielsweise eine Minimap anzuzeigen. Die Projektion dieser Minimap sollte nicht 1:1 von einem 1-Bildschirm-Setting (fix oben rechts) übernommen werden sondern flexibel sein. Sei dies fix gepinnt an den Center Screen oder dass eine ganzer Screen für die Minimap übernommen wird.

Bei sehr spezifischen Kameraeinstellungen können Randcases auftreten, welche nicht adaptiert werden können.



Abbildung 7: Minimap Game

5 Testing

5.1 System Tests

Während der Prototypingphase werden laufend Tests auf unabhängigen Rechnern sowie im CAVE durchgeführt um sicherzustellen, dass während der Entwicklung mögliche Probleme sofort erkannt werden und Massnahmen ergriffen werden können.

5.2 Usability Tests

Abhängig vom Fortschritt der Prototypen werden Tests mit potenziellen Anwendern durchgeführt um die Usability der Lösung abschätzen und optimieren zu können. Sowohl die Inbetriebnahme des CAVEs wie auch die Adaption der eigenen Unity Anwendungen werden berücksichtigt.

5.3 Szenarien

Als Testszenarien kommen sämtliche umgesetzten Demoapps zum Einsatz. Somit kann der komplexe Sachverhalt weitgehend geprüft werden. Diese Szenarien erscheinen glaubwürdig, sind komplex und können leicht überprüft werden.

6 Administratives

6.1 Projektorganisation

Auf eine stark strukturierte Projektorganisation wird bewusst verzichtet. Die Teammitglieder sind gleichberechtigt. Es kann vorkommen, dass verschiedene Teilprojekte und Verantwortungsbereiche den Teammitgliedern zugewiesen werden. Dies bedeutet aber nicht die alleinige Durchführung dieser Tasks.

6.1.1 Projektteam

Daniel Inversini daniel.inversini@students.bfh.ch
Julien Villiger julien.villiger@students.bfh.ch

6.1.2 Betreuer

Prof. Urs Künzler urs.kuenzler@bfh.ch

6.1.3 Experte

Dr. Harald Studer harald.studer@iss-ag.ch

6.2 Projektplan

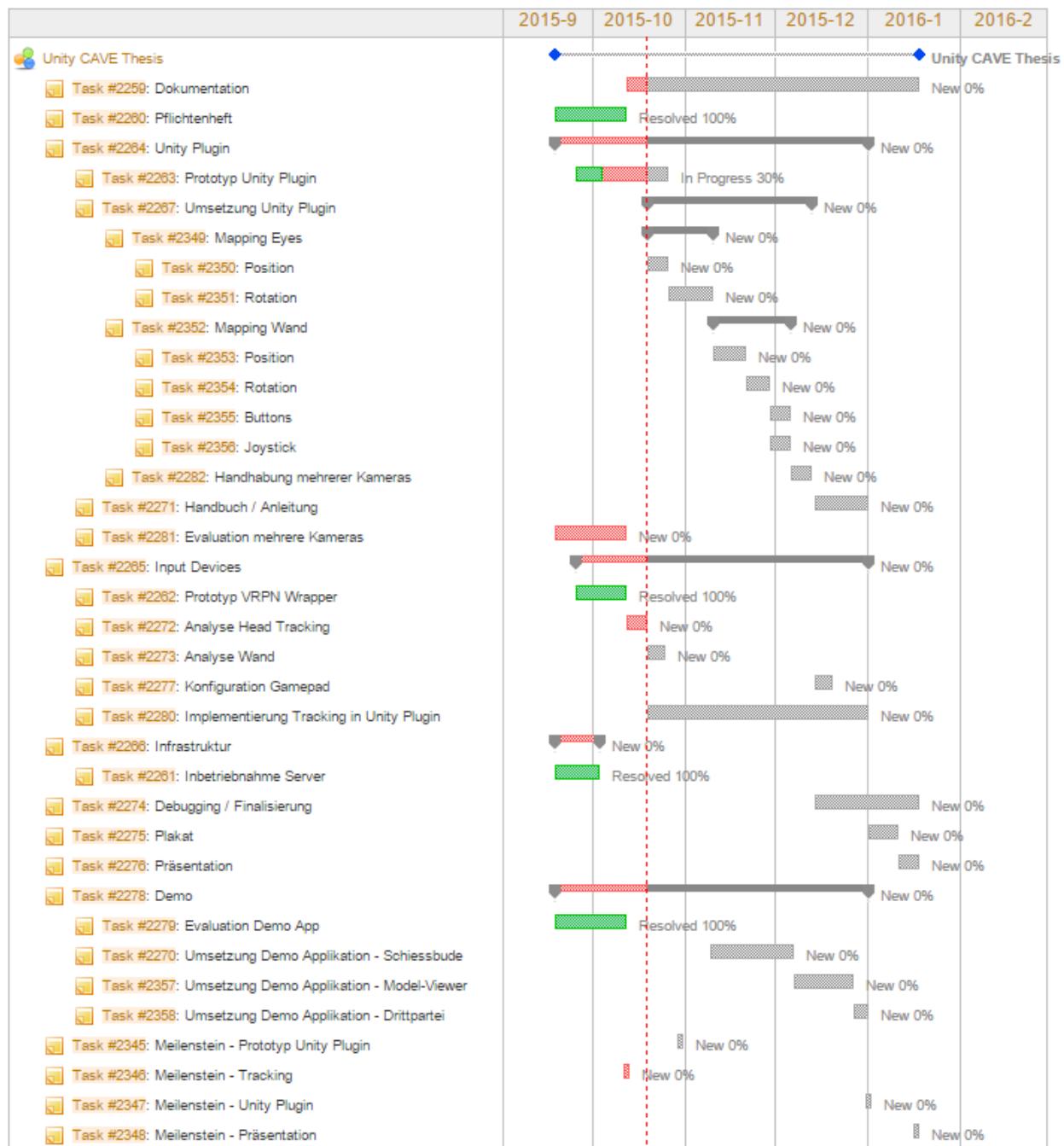


Abbildung 8: Projektplan

<https://pm.ti.bfh.ch/projects/unity-cave-thesis/issues/gantt>

6.3 Projektsitzungen

Rund alle zwei Wochen wird ein Projektmeeting des Teams mit Betreuer durchgeführt. Startend ab dem 16. September 2015.

6.4 Meilensteine

Folgende Tasks aus dem Projekt wurden als Meilensteine definiert:

6.4.1 Prototyp Unity Plugin

Stichtag 29.10.2015

Eine erste Implementierung des Unity Plugins mit grundlegender Funktionalität wurde umgesetzt.

6.4.2 Tracking

Stichtag 29.10.2015

Die Analysephase des VRPN Protokolls wurde abgeschlossen, damit die Integration des WorldViz Tracking Systems in das Unity Plugin erfolgen kann. Zusätzlich wird die Umsetzung einer Demo-Applikation gestartet.

6.4.3 Unity Plugin / Handbuch / Dokumentation

Stichtag 31.12.2015

Das Unity Plugin wurde fertiggestellt und getestet. Kleinere Anpassungen und die Finalisierung erfolgen noch. Zusätzlich wurde ein Handbuch / Anleitung erstellt, um die Verwendung des Plugins zu vereinfachen. Die Dokumentation ist an dieser Stelle ebenfalls möglichst weit fertiggestellt.

6.4.4 Präsentation

Stichtag 17.01.2016

Sämtliche Dokumente / Arbeiten sind abgeschlossen und eine Präsentation wurde erstellt.

7 Versionskontrolle

Version	Datum	Beschreibung	Autor(en)
0.1	18.09.2015	Dokument erstellt / Struktur definiert	Daniel Inversini Julien Villiger
0.2	20.09.2015	Funktionale Anforderungen	Julien Villiger
0.3	01.10.2015	Infrastruktur / Ausgangslage / allgemeines Updates	Julien Villiger
0.4	05.10.2015	Generelles Update, Review	Daniel Inversini
1.0	06.10.2015	Generelles Update	Julien Villiger Daniel Inversini
2.0	18.10.2015	Überarbeitung gemäss 14.10.2015	Feedback Daniel Inversini Julien Villiger

8 Abbildungs- und Tabellenverzeichnis

Abbildung 1: Unity Editor, Engine	4
Abbildung 2: Infrastruktur CAVE.....	5
Abbildung 3: WorldViz Wand.....	6
Abbildung 4: CAVE BFH	7
Abbildung 5: 6Dof.....	11
Abbildung 6: Image Wrapping (Quelle: github.com).....	14
Abbildung 6: Minimap Game.....	14
Abbildung 7: Projektplan	16

Tabelle 1: Übersicht Funktionale Anforderungen	9
Tabelle 1: Features Unitygames	13

9 Glossar

6	Bezeichnet den Raum, der verwendet wird um die dreidimensionale Illusionswelt zu erschaffen.
6DoF _____	2, 9, 11
Six degrees of freedom	
Beschreibt die Bewegung eines Körpers im Raum mit Freiheitsgrad 6. Drei unabhängige Richtungen (Translation) und drei unabhängige Achsen (Rotation).	
A	
Asset Store _____	9
Unity Asset Store	
Der Unity Asset Store ist eine Sammlung von frei verfügbaren wie auch kommerziellen Assets (Plugins) für Unity. Es können Scripts, Models, Texturen, Sound, etc sein. Alles was Unity unterstützt.	
B	
BFH _____	3, 7, 8, 18
Berner Fachhochschule BFH	
Die Institution des Studiums.	
C	
CAVE_____	1, 2, 3, 5, 7, 8, 9, 11, 13, 14, 18
Cave automatic virtual environment	
D	
Dynamic Link Library _____	9
Auch oft als DLL verwendet, enthält Programmcode, Daten und Ressourcen für eine Anwendung (EXE).	
E	
Equalizer_____	3
Equalizergraphics Parallel Rendering	
Middleware-Software, um OpenGL basierende Software auf mehrere Nodes, Graphikkarten und Prozessoren zu verteilen.	

Eyes _____ 5, 9

WorldViz Eyes
Brillengestütztes Trackinggerät, welches mit Infrarot LEDs ausgerüstet ist.

F

Frameworks _____ 3

Rahmenstruktur, welches dem Programmierer wiederverwendbare Strukturen zur Verfügung stellt.

G

Game Engine _____ 3

Spielengine

Stellt ein spezielles Framework für Computerspiele zur Verfügung. Unterstützt meistens mehrere Plattformen, und verschiedene Funktionalitäten von Spielen (Physik, Graphik, Sound, Netzwerk, KI, etc).

GPU _____ 3

graphical processing unit

Prozessor, der auf die Berechnung von Graphiken optimiert ist.

M

Matrix Switch _____ 6, 7

DVI Matrix Switch

Stellt ein Routing zur Verfügung von mehreren DVI Eingängen zu mehreren Displays.

MiddleVR _____ 3

middleVR for Unity

Kommerzielle Applikation, welche die Erstellung von VR Anwendungen unterstützt. Verschiedene Soft- und Hardware werden unterstützt und verbindet sie untereinander.

Minimap _____ 14, 19

Miniatur-Map

Stellt einen Überblick über die Spielwelt dar. Abstrahiert und simplifiziert die Details, sodass nur wichtige Akteure (Spieler, Quests, Gelände) dargezeigt werden.

Mosaic _____ 3

Nvidia Mosaic

Die NVIDIA Mosaic Mehrbildschirm Technologie dient zur einfachen Skalierung jeder Anwendung auf mehrere Bildschirme, und das ohne Softwareanpassungen oder Leistungseinbußen.

N

Nvidia _____ 4

Nvidia Corporation

Einer der weltweit grössten Hersteller von Chipsätzen und Graphikprozessoren.

O

OpenGL _____ 3

Open Graphics Library

Plattform- und Programmiersprachen unabhängige Programmierschnittstelle von 2D und 3D Computeranwendungen.

OpenSceneGraph _____ 3

Open Source Toolkit für Graphikanwendungen.

OS X _____ 4

Mac OS X

Kommerzielles Betriebssystem von Apple Computers Inc.

P

PPT Studio 2013 _____ 5

Software von WorldViz welche die Installation im CAVE der BFH softwareseitig unterstützt (bietet Kalibrierungseinstellungen, VRPN Output und vieles mehr an).

U

Unity _____ 1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 13, 14, 16, 19

Unity Game Engine

Unity ist eine Laufzeit- und Entwicklungsumgebung für Spiele und Simulationen.

V

VRPN _____ 2, 5, 8, 9, 10, 16

Virtual Reality Peripheral Network

Stellt verschiedene Tools (Server, Client, Klassen und Protokolle) zur Verfügung, um ein transparentes Interface zwischen einer Applikation und physikalischen Geräten zu bieten.

W

Wand _____ 2, 5, 6, 9, 10, 11, 13, 19

Einhändiger Controller, welcher über Infrarot LEDs verfügt, sowie mehrere Buttons und einen analogen Joystick.

Windows _____ 4

Microsoft Windows

Betriebssystem von Microsoft, aktuell in der Version 10.

WorldViz _____ 2, 4, 5, 6, 8, 9, 10, 19

Kommerzielle Firma welche komplette Systeme mit Infrarotkameras und Software anbietet.

Unity 3D Server for CAVE Rendering

Fachgebiet: Computer Perception and Virtual Reality

Betreuer: Prof. Urs Künzler

Experte: Dr. Harald Studer (Optimo Medical AG)

Neben der bestehenden CAVE Cluster-Rendering Lösung soll ein Unity 3D Render-Server in Betrieb genommen werden, um der zunehmenden Bedeutung von Unity Rechnung zu tragen. Existierende oder neue Applikationen sollen möglichst einfach in das Multi-Screen Rendering Setup des CAVEs integriert werden können, damit der Benutzer die Anwendung in 3D erleben kann. Das Plugin stellt das Rendern auf die Seitenwände des CAVEs sicher und macht Gebrauch vom vorhandenen Trackingsystem.

Ausgangslage

CAVE-Systeme (Cave Automatic Virtual Environment) werden für verschiedene Bereiche eingesetzt: CAD, Simulationen, medizinische Forschung, Unterhaltung, Psychologie und weitere Fachgebiete. Der CAVE der BFH war bisher eine Multi-Cluster Architektur, welche mittels OpenGL-Framework Befehle über das Netzwerk auf die verschiedenen Cluster verteilt. Durch die Unterstützung von Unity soll der ganze Workflow simplifiziert und optimiert werden.

Umsetzung

Die Hauptbestandteile der Umsetzung sind die virtuelle Abbildung der Komponenten in der Anwendung sowie die Erstellung aller benötigten Kameras für die Seitenwände des CAVEs. Mit Hilfe von Unity wird die Weiterverarbeitung und Interpretation vereinfacht und ist somit Basis für sämtliche Manipulationen der Applikation. Wie auf der Abbildung sichtbar ist, wurde der CAVE massstabsgetreu in die virtuelle Welt übernommen. So mit werden reale Gegebenheiten des Benutzers im CAVE übernommen und ermöglichen die Verwendung in Unity. So kann beispielsweise eine Kopfbewegung eine Änderung der Ansicht im Spiel bewirken.

Ergebnis

Das Plugin bietet ein konfigurierbares Interface, welches per Drag & Drop in Unity verwendet werden

kann. Neben Primäraufgaben, wie dem Verteilen des Renderings auf die verschiedenen Seitenwände und dem Verwenden des Infrarottrackings, sind noch viele zusätzliche Funktionen verfügbar. Sekundäre Kameras und UI-Elemente können frei auf den Seitenwänden des CAVEs platziert werden und der Wand von World-Viz übermittelt seine Eingaben als simulierte Tastatur- und Mauseingaben. Zudem lassen sich die Trackingdaten über VRPN filtern oder ganz deaktivieren. Im Hintergrund übernimmt das Plugin weitere Aufgaben. Dazu gehören die Bereitstellung der Stereoskopie für eine vollständige Immersion und 3D-Wahrnehmung sowie die Anpassung des Frustums, um eine realistische Perspektive zu gewährleisten. Weiter bietet das Plugin verschiedene Parameter und Einstellungsmöglichkeiten über ein API an.

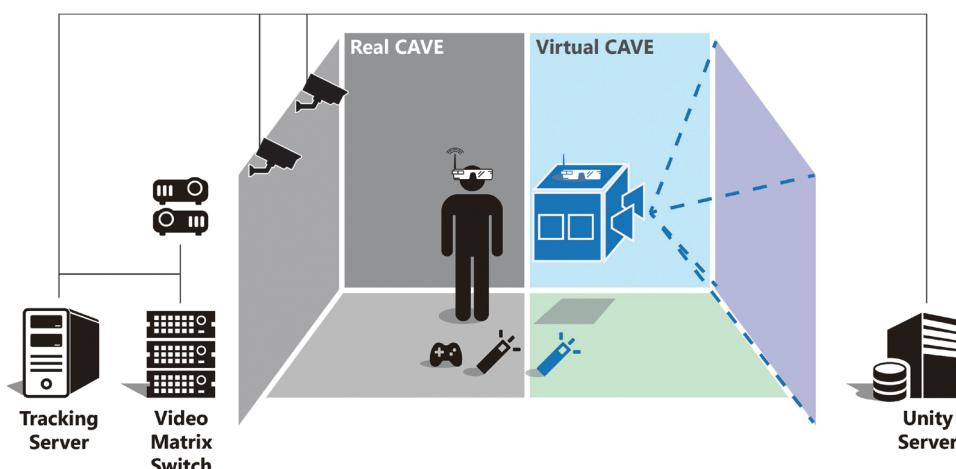
Um die Möglichkeiten des CAVEs zusammen mit Unity, dem Plugin und dem Trackingsystem zu demonstrieren, stehen zwei eigens erstellte Beispielapplikationen zur Verfügung. Durch die Einfachheit des Plugins können erstellte 3D-Welten innerhalb kurzer Zeit hautnah erlebt werden, was auch für andere Abteilungen der BFH von grossem Nutzen sein kann. Der Einsatz moderner Technologien, Hardware und Programmiersprachen bietet nun gute Zukunftsperspektiven für den CAVE der BFH.



Daniel Inversini
+41 79 712 52 57
daniel@inversini.com



Julien Villiger
julien.villiger@gmail.com

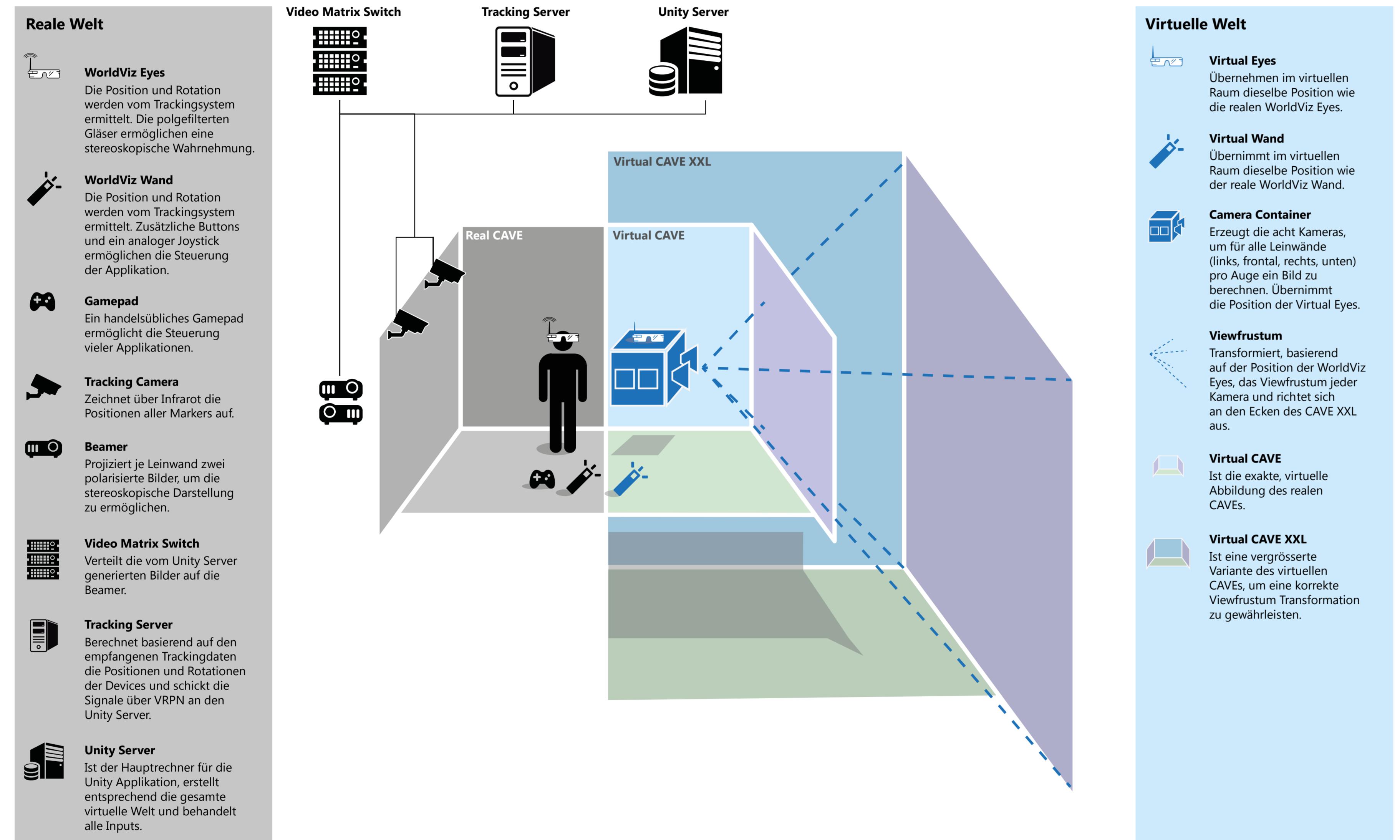


Ausgangslage

Neben der bestehenden CAVE Cluster-Rendering Lösung soll ein Unity 3D Render-Server in Betrieb genommen werden, um der zunehmenden Bedeutung von Unity im CPVRLab und im Unterricht Rechnung zu tragen. Bestehende oder neue Unity Applikationen sollen möglichst einfach in das Multi-Screen Rendering Setup des CAVEs integriert werden können, um Anwendungen in 3D zu erleben. Das Trackingsystem von WorldViz, welches die Ermittlung der Position des Benutzers im CAVE ermöglicht, soll ebenfalls zum Einsatz kommen.

Umsetzung

Der Hauptbestandteil der Umsetzung ist die virtuelle Abbildung der Komponenten in der Anwendung. Mit Hilfe von Unity wird die Weiterverarbeitung und Interpretation vereinfacht und ist somit Basis für sämtliche Manipulationen der Applikation.

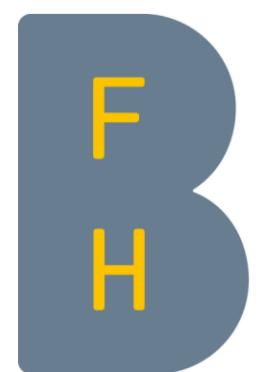


Ergebnis

Das Resultat der Arbeit ist ein Unity Package, welches mit wenigen Klicks in die eigene Applikation integriert werden kann und mit Drag & Drop aktiviert wird. Um eine möglichst grosse Bandbreite an Unity Anwendungen abdecken zu können, werden etliche Einstellungsmöglichkeiten zur Verfügung gestellt. So können beispielsweise zusätzliche Kameras individuell platziert oder die Darstellung der GUI-Elemente auf eine CAVE-Leinwand fixiert werden. Weiter lassen sich die Buttons auf dem WorldViz Wand, dem primären Inputgerät, frei zuordnen, um der eigenen Applikation zu entsprechen. Falls auf Wunsch nur spezifische Achsen bei der Weiterverarbeitung der Devices beachtet werden sollen, können diese auch frei eingeschaltet werden.

Die Viewfrustum-Transformation, welche basierend auf der Position des Benutzers im CAVE berechnet wird, gewährleistet eine realistische Perspektive im virtuellen Raum und lässt den Benutzer in die künstliche Welt eintauchen.

Zwei eigens erstellte Beispielapplikationen stehen zur Verfügung, um die Möglichkeiten des CAVEs zusammen mit Unity, dem Plugin und dem Trackingsystem zu demonstrieren. Dank der Einfachheit des Plugins können erstellte 3D Modelle innerhalb kurzer Zeit hautnah erlebt werden, was auch für andere Abteilungen der BFH von grossem Nutzen sein kann.



Bachelorthesis-Aufgabe

Unity 3D Server for CAVE Rendering

ID IKLU1-2-15

Studierende Daniel Inversini
Julien Villiger

Betreuer Urs Künzler

Experten Dr. Harald Studer

Aufgabe Neben der bestehenden CAVE Cluster-Rendering Lösung soll ein Unity 3D Render-Server in Betrieb genommen werden, um der zunehmenden Bedeutung von Unity im CPVRLab und im Unterricht Rechnung zu tragen. Es ist dabei eine Lösung zu entwickeln, welche eine einfache Integration von neuen oder bestehenden Unity Applikationen in das Multi-Screen Rendering Setup des CAVEs ermöglicht.

Folgende Anforderungen sollen dabei umgesetzt werden:

- Entwicklung eines geeigneten Setups oder APIs für ein einfaches Unity Multi-Screen Rendering.
- Integration des WorldWiz Tracking Systems für User Head Tracking und Kamerasteuerung.
- Implementation einer Demo-Applikation welche die Features des Frameworks demonstriert.
- Erstellen eines Entwickler-Handbuches mit detaillierter Anleitung und Tutorial.

Verwendete Technologien:

- OpenGL, C++, Unity 3D, Tracking System, CAVE