|  |
| --- |
| http://blog.gfx47.com/wp-content/uploads/2011/02/unity3d1.jpg |
| Unity 3D Server for CAVE  Rendering  Pflichtenheft  **Julien Villiger, Daniel Inversini**  **V0.3, 04.10.2015** |
| **Berner Fachhochschule**  Technik und Informatik  Informatik |

Inhaltsverzeichnis

[1 Allgemeines 3](#_Toc431752884)

[1.1 Zweck dieses Dokumentes 3](#_Toc431752885)

[1.2 Lesekreis 3](#_Toc431752886)

[1.3 Ausgangslage 3](#_Toc431752887)

[1.4 Unity 3](#_Toc431752888)

[1.5 Infrastruktur 4](#_Toc431752889)

[1.5.1 Inputs und Tracking 4](#_Toc431752890)

[1.5.2 Rendering 5](#_Toc431752891)

[1.5.3 Darstellung im CAVE 5](#_Toc431752892)

[1.6 Umfang und Ziele der Bachelor Thesis 5](#_Toc431752893)

[1.7 Abgrenzungen 5](#_Toc431752894)

[1.7.1 Technische Abgrenzungen 5](#_Toc431752895)

[1.7.2 Weitere Abgrenzungen 5](#_Toc431752896)

[1.8 Voraussetzungen und Ressourcen 5](#_Toc431752897)

[2 Funktionale Anforderungen 6](#_Toc431752898)

[2.1 Adaption Unity Anwendung für den CAVE 6](#_Toc431752899)

[2.2 Kompatibilität 6](#_Toc431752900)

[2.2.1 Dynamic Linked Library (.dll) 6](#_Toc431752901)

[2.2.2 Asset Store (Packages) 6](#_Toc431752902)

[2.2.3 Source Code API 6](#_Toc431752903)

[2.3 Plattformunabhängigkeit 6](#_Toc431752904)

[2.4 WorldViz Tracking 6](#_Toc431752905)

[2.4.1 Head Tracking 6](#_Toc431752906)

[2.4.2 Wand Tracking 7](#_Toc431752907)

[2.5 Setup 7](#_Toc431752908)

[2.6 Demoapplikation 7](#_Toc431752909)

[2.7 Features 8](#_Toc431752910)

[3 Nicht funktionale Anforderungen 8](#_Toc431752911)

[3.1 Wiederverwendbarkeit 8](#_Toc431752912)

[3.2 Ergonomie 9](#_Toc431752913)

[4 Testing 9](#_Toc431752914)

[4.1 Systemtests 9](#_Toc431752915)

[4.2 Usabilitytests 9](#_Toc431752916)

[4.3 Szenarien 9](#_Toc431752917)

[5 Administratives 9](#_Toc431752918)

[5.1 Projektorganisation 9](#_Toc431752919)

[5.1.1 Projektteam 9](#_Toc431752920)

[5.1.2 Betreuer 9](#_Toc431752921)

[5.1.3 Experte 9](#_Toc431752922)

[5.2 Projektplan 10](#_Toc431752923)

[5.3 Projektsitzungen 10](#_Toc431752924)

[5.4 Meilensteine 10](#_Toc431752925)

[5.4.1 Prototyp Unity Plugin 10](#_Toc431752926)

[5.4.2 Tracking 10](#_Toc431752927)

[5.4.3 Unity Plugin / Handbuch / Dokumentation 10](#_Toc431752928)

[5.4.4 Präsentation 11](#_Toc431752929)

[6 Versionskontrolle 11](#_Toc431752930)

# Allgemeines

## Zweck dieses Dokumentes

Mit diesem Pflichtenheft wird der Rahmen, die Vorgehensweise und die Ziele der Projekt 2 – Arbeit dokumentiert.

## 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 CPVR und an die Studenten, welche diese Projektarbeit durchführen.

## Ausgangslage

Das cpvrLab besitzt eine CAVE (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 in stereoskopisch projiziert, sodass eine nahezu perfekte Raumwahrnehmung entsteht. Die Entwicklung von virtuellen 3D-Welten mit Basis-APIs wie OpenGL oder OpenSceneGraph 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 Software MiddleVR, die den Einsatz von Unity in einem CAVE deutlich vereinfachen sollte. 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
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 Mosaic 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.

## 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)>)

## 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:

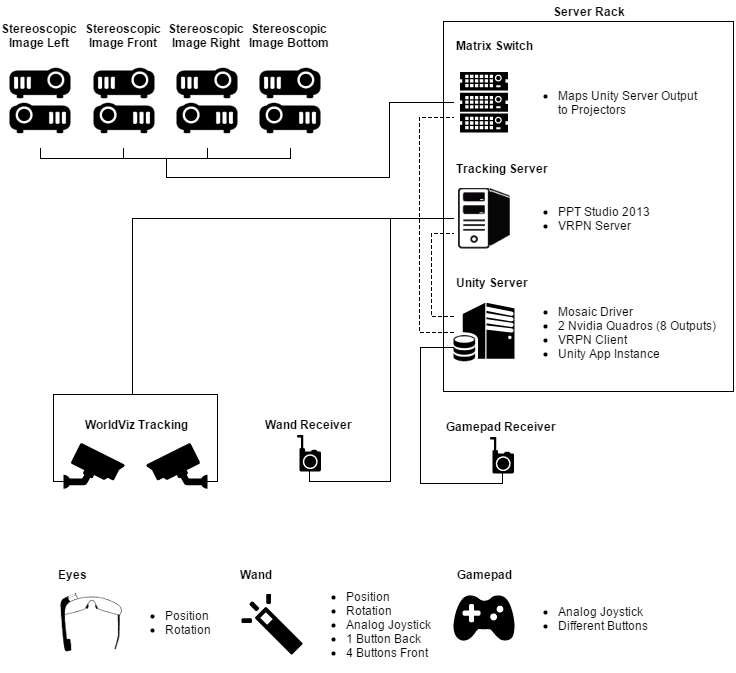


Abbildung 1: Infrastruktur CAVE

### Inputs und Tracking

Die beiden Tracking Devices (Eyes und Wand) werden durch 10 Infrarotkameras (WorldViz 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 VRPN 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.

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

### Rendering

Auf dem Unity Server läuft die Unity Applikation und die über das VPRN Protokoll gelieferten Informationen 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 gerendert wird, auf die 8 vorhandenen Grafikkarten Outputs verteilt und an den Matrix Switch überliefert.

### 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.

## 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 Plugin).
3. Implementierung einer Unity-Demoapplikation, 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-Applikationin kurzer Zeit im CAVE der BFH zu verwenden.

## Abgrenzungen

### Technische Abgrenzungen

Um eine möglichst saubere, wartebare 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.

### Weitere Abgrenzungen

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

## 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.

# Funktionale Anforderungen

## 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.

## Kompatibilität

Sämtliche, quelloffene Unity Anwendungen ab Version 5.0 müssen 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 Linked Library (.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 API

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.

## Plattformunabhängigkeit

Durch die plattformunabhängige Architektur von Unity können die Anwendungen im Rahmen der Möglichkeiten von Unity umgesetzt werden.

## WorldViz Tracking

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

### 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 Applikationslogik 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 uns 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.

## 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.

## Demoapplikation

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

* **Schiessbude**

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

Das Setting dieses Demospiels 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, leichte Bewegungen ausführen und die abzuschiessenden Objekte aus verschiedenen Perspektiven betrachten. Das Wand Device steuert das Luftgewehr, um die Zielscheiben anzuvisieren und mit Hilfe des Joysticks kann sich der Spieler in der Schiessbude frei bewegen und Drehungen ausführen. Die Buttons des Wands werden gebraucht um das Gewehr abzufeuern und nachzuladen.

* **Statische Welt**

Es wird ein Operationssaal dargestellt, wo sich der Benutzer im CAVE wie ein Arzt darin bewegen kann. Als Alternative zum Operationssaal könnten auch Gebäude, Städte und sonstige Modelle Verwendung finden.

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**

Eine nicht von uns, 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.

## 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 | ? |  |
| Gamepad | X | ? |  |
| Vordefinierte Kamerasteuerung | X | X |  |
| Vordefinierte Spiellogik | X | X |  |
| Sekundäre Kameras |  | ? | X |

X: Vorhanden

?: Abhängig der Unity Applikation

# Nicht funktionale Anforderungen

## 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.

## Ergonomie

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

# Testing

## Systemtests

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.

## Usabilitytests

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.

## 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.

# Administratives

## 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.

### Projektteam

Daniel Inversini [daniel.inversini@students.bfh.ch](mailto:daniel.inversini@students.bfh.ch)

Julien Villiger [julien.villiger@students.bfh.ch](mailto:julien.villiger@students.bfh.ch)

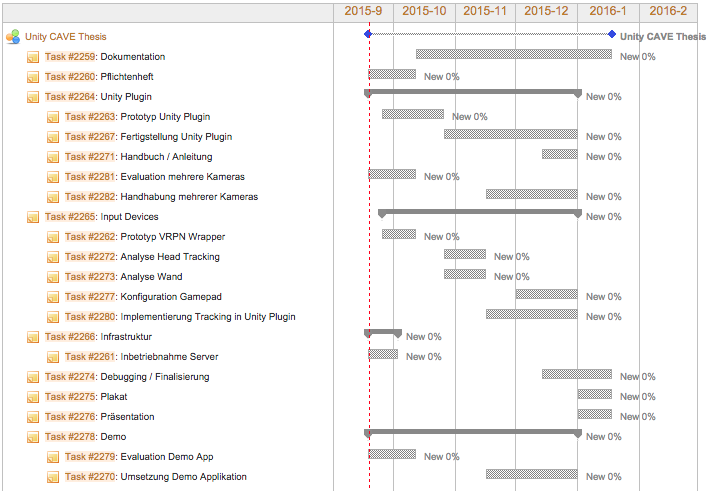
### Betreuer

Prof. Urs Künzler [urs.kuenzler@bfh.ch](mailto:urs.kuenzler@bfh.ch)

### Experte

Dr. Harald Studer

## Projektplan



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

## Projektsitzungen

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

## Meilensteine

Folgende Tasks aus dem Projekt wurden als Meilensteine definiert:

### Prototyp Unity Plugin

Stichtag 29.10.2015

Eine erste Implementierung des Unity Plugins mit grundlegender Funktionalität wurde umgesetzt, damit die Analysephase des Trackings gestartet werden kann.

### Tracking

Stichtag 15.11.2015

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

### 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.

### Präsentation

Stichtag 17.01.2016

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

# 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 |