|  |
| --- |
| http://blog.gfx47.com/wp-content/uploads/2011/02/unity3d1.jpg |
| Unity 3D Server for CAVE Rendering  Pflichtenheft  **Julien Villiger, Daniel Inversini**  **V0.2, 18.09.2015** |
| **Berner Fachhochschule**  Technik und Informatik  Informatik |

Inhaltsverzeichnis

[1 Allgemeines 3](#_Toc430534034)

[1.1 Zweck dieses Dokumentes 3](#_Toc430534035)

[1.2 Lesekreis 3](#_Toc430534036)

[1.3 Ausgangslage 3](#_Toc430534037)

[1.4 Umfang der Bachelor Thesis 3](#_Toc430534038)

[1.5 Ziele der Arbeit 3](#_Toc430534039)

[1.6 Abgrenzungen 3](#_Toc430534040)

[1.6.1 Technische Abgrenzungen 3](#_Toc430534041)

[1.6.2 Weitere Abgrenzungen 4](#_Toc430534042)

[1.7 Voraussetzungen und Ressourcen 4](#_Toc430534043)

[2 Funktionale Anforderungen 5](#_Toc430534044)

[2.1 Adaption Unity Anwendung für den CAVE 5](#_Toc430534045)

[2.2 Kompatibilität 5](#_Toc430534046)

[2.2.1 Dynamic Linked Library (.dll) 5](#_Toc430534047)

[2.2.2 Asset Store (Packages) 5](#_Toc430534048)

[2.2.3 Source Code API 5](#_Toc430534049)

[2.3 Plattformunabhängigkeit 5](#_Toc430534050)

[2.4 WorldViz Tracking 5](#_Toc430534051)

[2.4.1 Head Tracking 5](#_Toc430534052)

[2.4.2 Wand Tracking 6](#_Toc430534053)

[2.5 Setup 6](#_Toc430534054)

[2.6 Demoapplikation 6](#_Toc430534055)

[3 Nicht funktionale Anforderungen 7](#_Toc430534056)

[3.1 Wiederverwendbarkeit 7](#_Toc430534057)

[3.2 Ergonomie 7](#_Toc430534058)

[3.3 Kompatibilität 7](#_Toc430534059)

[4 Testing 7](#_Toc430534060)

[5 Administratives 7](#_Toc430534061)

[5.1 Projektorganisation 7](#_Toc430534062)

[5.1.1 Projektteam 7](#_Toc430534063)

[5.1.2 Betreuer 7](#_Toc430534064)

[5.1.3 Experte 7](#_Toc430534065)

[5.2 Projektplan 8](#_Toc430534066)

[5.3 Projektsitzungen 8](#_Toc430534067)

[5.4 Meilensteine 8](#_Toc430534068)

[5.4.1 Prototyp Unity Plugin 8](#_Toc430534069)

[5.4.2 Tracking 8](#_Toc430534070)

[5.4.3 Unity Plugin / Handbuch / Dokumentation 8](#_Toc430534071)

[5.4.4 Präsentation 9](#_Toc430534072)

[6 Versionskontrolle 9](#_Toc430534073)

[7 Abbildungsverzeichnis 9](#_Toc430534074)

[8 Tabellenverzeichnis 9](#_Toc430534075)

[9 Glossar 9](#_Toc430534076)

[10 Literaturverzeichnis 9](#_Toc430534077)

[11 Anhang 10](#_Toc430534078)

# 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 den Betreuer dieser Arbeit, Prof. Urs Künzler, die BFH-TI Abteilung CPVR und an die Studenten, welche diese Projektarbeit durchführen.

## Ausgangslage

Das cpvrLab besitzt eine CAVE Installation (Cave Automated Virtual Environment) 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.

## Umfang der Bachelor Thesis

Es ist geplant, einen lauffähigen Prototyp zu erstellen. Dies kann ein Spiel oder eine Tech Demo sein, basierend auf der Game Engine Unity. Wenn der Prototyp / die Prototypen innerhalb der Projektarbeit in den CAVE portiert werden können, ist es möglich, noch weitere VR Aspekte wie Tracking zu überprüfen.

Falls es nicht komplett abgeschlossene Punkte / Features gibt, welche für einen voll funktionsfähigen Prototypen zwingend sind, werden diese aufgenommen.

Im Rahmen einer weiteren Projektarbeit oder einer Bachelorarbeit können fehlende Komponenten und Features realisiert oder wie erwähnt zusätzliche VR Aspekte implementiert werden.

## Ziele der Arbeit

Folgende Ziele sind für das Projekt 2 definiert:

1. Einarbeiten in Unity
2. Einarbeiten in die Theorie der stereoskopischen Projektion
3. Entwicklung eines Demospiels / Anwendung / Demo für den cpvrLab CAVE mit Stereoprojektion
4. Dokumentierte API der schlussendlich verwendeten Lösung
5. Einarbeiten in die Features von Unity Pro Edition

Wobei Punkt 5 sekundär zu betrachten ist, denn u.U. ist eine Pro Edition von Unity nicht nötig.

## Abgrenzungen

### Technische Abgrenzungen

Da wir eine saubere, wartbare und moderne Applikation / Lösung bieten wollen, möchten wir uns falls möglich auf Unity, respektive C# begrenzen. Low-Level Implementationen in C, C++ (auch FreeGLut) möchten wir keine vornehmen.

### Weitere Abgrenzungen

Im Rahmen der Projekt 2 Arbeit sind lauffähige Prototypen genügend. Komplette Setups und Schulungen sind nicht vorgesehen.

## Voraussetzungen und Ressourcen

Folgende zwei Voraussetzungen sind zwingend für die Studenten, welche dieses Projekt durchführen:

1. Besuch des Unity Kurses an der BFH (BTI7527a – Game Development)
2. Besuch der Vertiefungsrichtung CPVR (<http://www.cpvrlab.ti.bfh.ch/>)

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 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 Head Tracking. Diese Translation und Rotation ist aber nur die Veränderung des Kopfes in Relation zur Änderung, die ohnehin von der Applikation gegeben ist.

Entscheidend ist, dass die zwei Kamera-Rotationen und -Translationen, also Input der Unity Anwendung und des WorldViz Trackings, klar getrennt werden, damit die in der Unity Anwendung definierten Kamerascripts nicht beeinträchtig werden. 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:

* Position (Würd ich nicht interpretieren? Sähe keinen Anwendungsfall.)
* **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.

## 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 erstellt.

## Demoapplikation

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

* **Eigene Demoapp**

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.

* **Demoapp einer Dreittpartei**

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.

Als Applikation dient das Autorennspiel, welches beim Downloaden von Unity 5.1 automatisch mitgeliefert wird.

# Nicht funktionale Anforderungen

## Wiederverwendbarkeit

## Ergonomie

## Kompatibilität

# Testing

# 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)

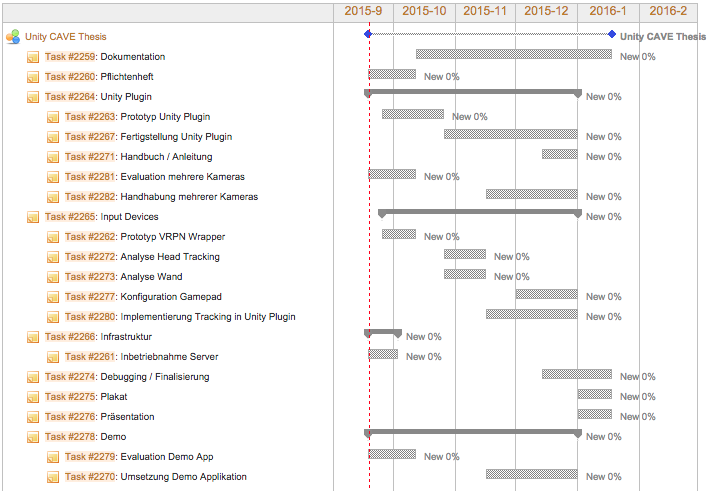
### Betreuer

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

### Experte

tbd

## 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 |
|  |  |  |  |

# Abbildungsverzeichnis

Abbildung 1: Et ut aut isti repuditis qui ium 4

# Tabellenverzeichnis

Tabelle 1: Et ut aut isti repuditis qui ium 4

# Glossar

**Auinweon**

Et ut aut isti repuditis qui ium 7

**Batnwpe**

Et ut aut isti repuditis qui ium 9

**Cowoll**

Et ut aut isti repuditis qui ium 11

# Literaturverzeichnis

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 7

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 9

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 11

# Anhang

Et ut aut isti repuditis qui ium nonsecturia quis incientiae laborem elliquis et quatur, sitiur aut od moluptatur aut ea conseque peri sim erro essequisit remporia dem et landi dest, cone poris quunt volecab ipidero quatur ad quibusamus.