



# Unity in CAVE

## Dokumentation

**Julien Villiger, Daniel Inversini**  
**V 1.2, 20.04.2015**

# Inhaltsverzeichnis

|         |   |    |
|---------|---|----|
| 1       | Einführung                                    | 5  |
| 2       | Voranalyse Varianten                          | 6  |
| 3       | Chromium                                      | 7  |
| 3.1     | Architektur                                   | 7  |
| 3.2     | Anwendung                                     | 8  |
| 3.2.1   | Multi-Monitor Displays                        | 8  |
| 3.2.2   | Delegation                                    | 8  |
| 3.2.3   | Manipulation                                  | 8  |
| 3.2.4   | Stereoskopie                                  | 8  |
| 3.2.5   | Command Stream Aufteilung                     | 8  |
| 3.3     | Argumentation                                 | 8  |
| 3.3.1   | Pro   | 8  |
| 3.3.1.1 | Stereoskopie                                  | 8  |
| 3.3.1.2 | Infrastruktur                                 | 9  |
| 3.3.1.3 | Aufteilung der Monitore                       | 9  |
| 3.3.2   | Kontra  | 9  |
| 3.3.2.1 | Plattform                                     | 9  |
| 3.3.2.2 | Maintenance                                   | 9  |
| 3.3.2.3 | OpenGL Support                                | 9  |
| 3.3.2.4 | Kompatibilität Unity3D                        | 9  |
| 3.3.2.5 | Netzwerkauslastung                            | 9  |
| 3.3.2.6 | Community                                     | 9  |
| 3.4     | Zusammenfassung                               | 9  |
| 4       | Equalizer – The parallel rendering framework  | 11 |
| 4.1     | Warum Voranalyse Equalizer?                   | 11 |
| 4.2     | Architektur                                   | 11 |
| 4.3     | Anwendung                                     | 12 |
| 4.3.1   | Multi Displays                                | 12 |
| 4.4     | Argumentation                                 | 13 |
| 4.4.1   | Pro   | 13 |
| 4.4.1.1 | Open Source, Dokumentation                    | 13 |
| 4.4.1.2 | Know-How BFH                                  | 13 |
| 4.4.2   | Kontra  | 13 |
| 4.4.2.1 | Unity3D Bezug                                 | 13 |
| 4.4.2.2 | Architektur                                   | 13 |
| 4.5     | Zusammenfassung                               | 13 |
| 5       | Eigene Lösung                                 | 14 |
| 5.1     | Idee  | 14 |
| 5.1.1   | Unity   | 14 |
| 5.1.2   | Eigenes Protokoll                             | 14 |
| 5.2     | Architektur                                   | 14 |
| 5.2.1   | Aufgabe Server                                | 15 |
| 5.2.2   | Aufgabe Client                                | 15 |
| 5.3     | Funktionale Anforderungen                     | 15 |
| 5.3.1   | Multi-Monitor Displays                        | 15 |
| 5.3.2   | Stereoskopie                                  | 15 |
| 5.3.3   | Implementierung bestehender Unity Anwendungen | 16 |
| 5.4     | Argumentation                                 | 16 |
| 5.4.1   | Pro   | 16 |
| 5.4.1.1 | Keine Abhängigkeit                            | 16 |

|   |    |
|---|----|
| 5.4.1.2 Community                             | 16 |
| 5.4.1.3 Netzwerkauslastung                    | 16 |
| 5.4.2 Kontra                                  | 16 |
| 5.4.2.1 From Scratch                          | 16 |
| 5.4.2.2 Aufwand                               | 16 |
| 5.5 Zusammenfassung                           | 16 |
| 6 middleVR                                    | 17 |
| 6.1 Idee                                      | 17 |
| 6.2 Warum middleVR?                           | 17 |
| 6.3 Abdeckung middleVR                        | 17 |
| 6.3.1 Stereoskopie                            | 17 |
| 6.3.2 Unsere funktionalen Anforderungen       | 18 |
| 6.3.3 Unsere nicht funktionalen Anforderungen | 18 |
| 6.4 Konzept middleVR                          | 18 |
| 6.4.1 Beschreibung                            | 18 |
| 6.4.1.1 Beispiel                              | 18 |
| 6.5 Argumentation                             | 19 |
| 6.5.1 Pro                                     | 19 |
| 6.5.1.1 Bereits ausgearbeitete Software       | 19 |
| 6.5.1.2 Aufwand                               | 19 |
| 6.5.1.3 Support                               | 19 |
| 6.5.2 Kontra                                  | 19 |
| 6.5.2.1 Abhängigkeit                          | 19 |
| 6.5.2.2 Eigenanteil                           | 19 |
| 6.6 Zusammenfassung                           | 19 |
| 7 Gegenüberstellung                           | 20 |
| 8 Entscheid                                   | 22 |
| 9 middleVR Prototyping                        | 23 |
| 9.1 Prototypen                                | 23 |
| 9.1.1 Basic                                   | 23 |
| 9.1.2 AngryBots                               | 23 |
| 9.1.3 Car Tutorial                            | 24 |
| 9.2 Variantenentscheid                        | 25 |
| 9.3 Aktuelle Infrastruktur                    | 25 |
| 9.4 Variante Multi-GPU                        | 26 |
| 9.4.1 Lizenzen                                | 26 |
| 9.4.2 Adaption Infrastruktur                  | 26 |
| 9.4.3 Vorteile                                | 26 |
| 9.4.4 Nachteile                               | 26 |
| 9.4.5 Fazit                                   | 27 |
| 9.5 Variante Cluster                          | 27 |
| 9.5.1 Lizenzen                                | 28 |
| 9.5.2 Vorteile                                | 28 |
| 9.5.3 Nachteile                               | 28 |
| 9.5.4 Fazit                                   | 28 |
| 9.6 Variante Nvidia Mosaic                    | 28 |
| 9.6.1 Was ist Nvidia Mosaic?                  | 28 |
| 9.6.2 Limitierungen                           | 29 |
| 9.6.3 Kosten und Lizenzen                     | 29 |
| 9.6.4 Vorteile                                | 30 |
| 9.6.5 Nachteile                               | 30 |
| 9.6.6 Fazit                                   | 30 |
| 10 Abbildungsverzeichnis                      | 31 |
| 11 Tabellenverzeichnis                        | 31 |

|                         |    |
|-------------------------|----|
| 12 Glossar              | 31 |
| 13 Literaturverzeichnis | 32 |
| 14 Anhang               | 32 |
| 15 Versionskontrolle    | 32 |

# 1 Einführung

## 2 Voranalyse Varianten

Basierend auf einer Besprechung mit unserem Betreuer wurden vier Möglichkeiten in Erwägung gezogen, wie die Integration von Unity in den CAVE erfolgen kann.

1. Chromium
2. Equalizer
3. Eigene Lösung
4. MiddleVR

Um eine standfeste Entscheidung treffen zu können und die Umsetzung eines Prototyps anzugehen, wurde für jede Variante eine detaillierte Voranalyse vorgenommen. Die Analysen dienen als Gegenüberstellung und sind Basis für den Variantenentscheid.

## 3 Chromium

### 3.1 Architektur

Chromium ist eine OpenGL Implementation. Doch entgegen üblicher Implementationen, wird der OpenGL Command nicht in ein Rasterbild umgewandelt, sondern wird manipuliert und an andere OpenGL Implementationen weitergeschickt.

Die Chromium Bibliothek unterstützt eine Server / Client Architektur. Die Verarbeitungskette ist unterteilt in mehrere Stream Processing Units, kurz SPU.

Für jeden OpenGL Command hat eine SPU folgende Möglichkeiten:

- Modifizieren
- Ablehnen
- An eine weitere SPU weiter schicken

Die letzte SPU hat die Wahl, den OpenGL Command an eine lokale OpenGL Implementation zu überreichen um ein Rasterbild zu generieren, oder über ein Netzwerk an einen oder mehrere Chromium Servers zu schicken.

Die Chromium-Instanz läuft auf dem sogenannten „Mothership“ und managed die SPU Kette und Netzwerkverbindungen. Die laufende Applikation setzt die Drawcalls an die Hauptinstanz (Mothership) ab.

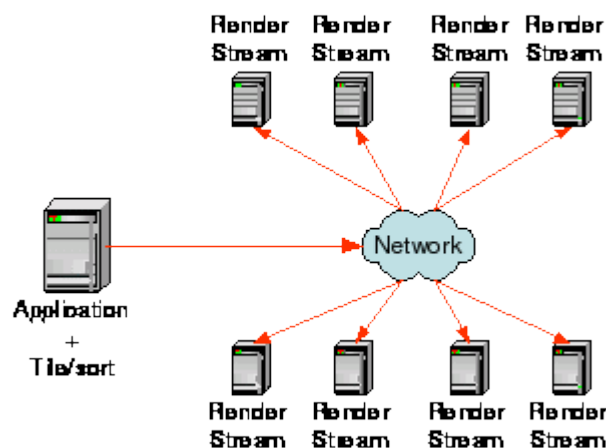


Abbildung 1: Architektur von Chromium mit Mothership und SUPs Quelle: <http://chromium.sourceforge.net/doc/index.html>

## 3.2 Anwendung

### 3.2.1 Multi-Monitor Displays

Darstellung der OpenGL Commands auf mehreren Displays. Konzipiert für einen CAVE mit mehreren Leinwänden.

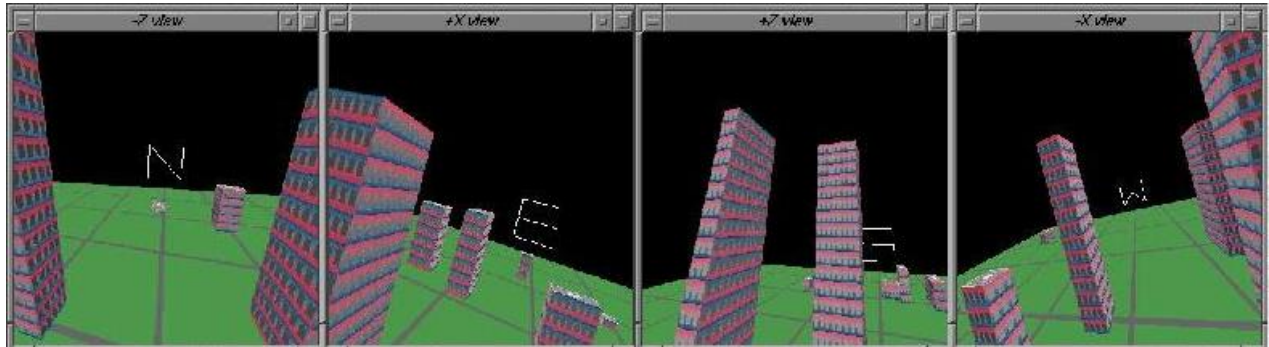


Abbildung 2: Ansicht der Leinwände in vier Himmelsrichtungen, Quelle: <http://chromium.sourceforge.net/presentations/SantaFe-BrianPaul/siframes.html>

### 3.2.2 Delegation

Der gesamte OpenGL Stream kann von einem Rechner auf den anderen verschoben werden. Hat eine Maschine keine dedizierte Grafikkarte, kann diese Aufgabe an einen besser ausgerüsteten Rechner delegiert werden.

### 3.2.3 Manipulation

Polygone eines OpenGL Streams können manipuliert werden. Sogar eigene Rendering Styles können dank der komplett programmierbaren Rendering Pipeline von Chromium implementiert werden.

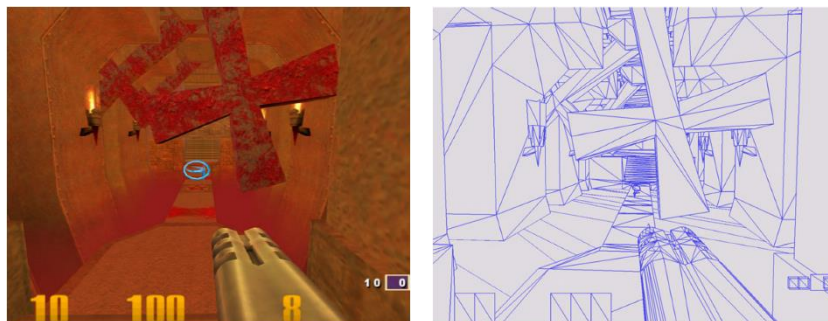


Abbildung 3: Implementation eigener Rendering Styles, Quelle: <http://chromium.sourceforge.net/doc/index.html>

### 3.2.4 Stereoskopie

Nicht-Stereoskopische Anwendungen können in Stereoskopische umgewandelt werden. Aktive (Shutter Glasses) sowie passive (Polarisierte Lichtprojektion) Stereoskopie werden unterstützt.

### 3.2.5 Command Stream Aufteilung

Ein OpenGL Command Stream kann aufgeteilt werden, damit versch. Rechner einen Teil des Renderings übernehmen können. Vergleichbar mit nVidia's SLI.

## 3.3 Argumentation

### 3.3.1 Pro

#### 3.3.1.1 Stereoskopie

Das wichtigste Feature, die Umwandlung in eine stereoskopische Darstellung, wird angeboten.



#### 3.3.1.2 Infrastruktur

Die benötigte Infrastruktur ist gegeben. Das Rendering kann auf mehrere Rechner verteilt werden.

#### 3.3.1.3 Aufteilung der Monitore

Jeder Projektor im CAVE hat einen eigenen Viewport und diese Aufteilung wird unterstützt.

### 3.3.2 Kontra

#### 3.3.2.1 Plattform

Chromium wurde auf Linux entwickelt und sollte auf diesem System ausgeführt werden. Unter Windows und OSX gibt es bekannte Probleme. Das verlangte Linuxwissen müsste zusätzlich erarbeitet werden.

#### 3.3.2.2 Maintenance

Seit 2006 gab es kein Update mehr. In einer Präsentation aus dem Jahre 2004 wird als nächste Phase der Support von OpenGL 2.0 angestrebt (Quelle: Slide 27, <http://chromium.sourceforge.net/presentations/SantaFe-BrianPaul/siframes.html>). Bis anhin wurde dieser Task nicht umgesetzt.

Falls Anwender von Chromium auf ein Problem stossen, können Feature Requests auf sourceforge.net abgesetzt werden (<http://sourceforge.net/p/chromium/feature-requests/>). Die letzten Requests wurden im Jahre 2002 bearbeitet und geschlossen. Neu erstellte Einträge

Zitat auf der offiziellen Sourceforge-Seite (<http://sourceforge.net/projects/chromium/>), 22.01.2015:  
„UPDATE: Chromium is no longer updated or maintained. The project is frozen.“

#### 3.3.2.3 OpenGL Support

Die letzte noch unterstützte OpenGL Version war 1.5 mit Chromium Release 1.5 (Dezember 2003). Die aktuelle Version von OpenGL ist 4.5 (Release August 2014).

#### 3.3.2.4 Kompatibilität Unity3D

Etliche Features, die über die OpenGL Version 1.5 hinausgehen und von Unity3D verwendet werden, könnten bei Chromium zu schwerwiegenden Problemen führen.

Der Output von Unity3D könnte inkompatibel mit den SUPs sein. Eine Modifikation des Outputs müsste in Betracht gezogen werden, wobei der Aufwand sehr schwer abschätzbar und nur bedingt zielführend ist.

#### 3.3.2.5 Netzwerkauslastung

Engpässe könnten entstehen, weil der gesamte OpenGL Stream übers Netz geschickt wird. Bei simplen Anwendungen mit wenigen Primitiven sollte die Performance ausreichen, in Anbetracht dessen, dass wir komplexe Unity3D Spiele rendern wollen, würde sicherlich die Netzwerkkapazität nicht ausreichen. Vorgängige Tests müssten durchgeführt werden.

#### 3.3.2.6 Community

Chromium hat keine aktive Community mehr, die bei Problemen bei der Installation Hilfestellung bieten könnte.

## 3.4 Zusammenfassung

Basierend auf der Gegenüberstellung der Pro- und Kontra-Argumentation und obwohl unsere geforderten Key-Features von der Chromium Graphics Library abgedeckt werden, sind die Nachteile massiv überwiegend.

Ausschlaggebend sind in erster Linie der eingestellte Support und die fehlende Weiterentwicklung der Bibliothek. Solange Unity3D und OpenGL sich am Weiterentwickeln sind, müsste Chromium laufend nachziehen und die neu entwickelten Features unterstützen.

Das Ziel der Thesis ist der Einsatz moderner und zukunftsorientierter Technologien. Wird auf ein Relikt gesetzt, ist der Erfolg der Umsetzung fraglich und keinesfalls eine robuste Basis, um zeitgemässe Anwendungen laufen zu lassen.

## 4 Equalizer – The parallel rendering framework

Equalizer ist ein Open Source Framework für skalierbares, paralleles Rendering basierend auf OpenGL, welches ein API zur Verfügung stellt um solche graphischen Applikationen zu entwickeln.

### 4.1 Warum Voranalyse Equalizer?

Heute wird das Equalizer Framework im CAVE der BFH bereits verwendet.

Das Ziel unseres Projekts ist es, Möglichkeiten aufzuzeigen, wie das Unity3D im aktuellen CAVE verwendet werden kann. Da unter Umständen Komponenten wiederverwendet werden können, wollen wir dieses API überprüfen.

### 4.2 Architektur

Equalizer verwendet verschiedene Wrapperklassen, um die Systemressourcen abstrahiert darzustellen. Die gesamte Liste der Ressourcenklassen ist hier zu finden:

<http://www.equalizergraphics.com/documents/Developer/API-1.0/internal/annotated.html>

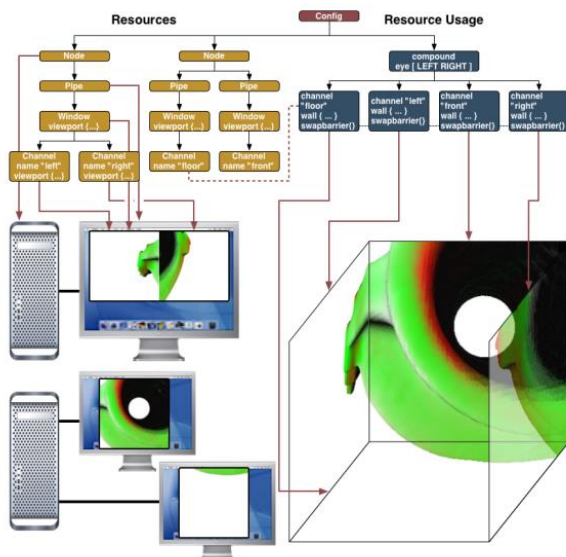


Abbildung 4: Equalizer Beispiel, Quelle: <http://www.equalizergraphics.com/documents/Developer/eqPly.pdf>

Nodes identifizieren einzelne Computer innerhalb des Clusters, wovon jeder mehrere Graphikkarten besitzen kann, Pipe. Dies definiert sich dann weiter zu Window, welche die einzelnen OpenGL Drawables und Context verwalten.

Abbildung 1 stellt ein Beispiel mit zwei Computern und drei Graphikkarten dar, welche vier Wände eines CAVEs rendern (Cave Automatic Virtual Environment)

Als Überblick noch die wichtigsten Equalizer Klassen:

| Equalizer Klasse | Information                                    |
|------------------|--|
| eq::Config       | Beschreibung der vorhandenen Ressourcen        |
| eq::Node         | Ein Client (Computer) im Rendering Cluster     |
| eq::Pipe         | Graphikkarte des Knoten (Node)                 |
| eq::Window       | OpenGL Drawable auf der entsprechenden Pipe    |
| eq::Channel      | Viewport im Window                             |
| eqNet::Object    | Verteiltes Objekt für gemeinsam genutzte Daten |

Tabelle 1: Wichtige Equalizer Klassen

### 4.3 Anwendung

Equalizer kann auf alle Applikationen angewendet werden, welche Quellcode offen sind und auf OpenGL basieren. Vorzugsweise sollte die Applikation in C++ wie Equalizer programmiert sein,

Da Equalizer sehr flexibel ist, sind verschiedene Anwendungen möglich, hier nur Auszüge, welche uns interessieren:

Komplette Liste unter <http://www.equalizergraphics.com/useCases.html>.

#### 4.3.1 Multi Displays

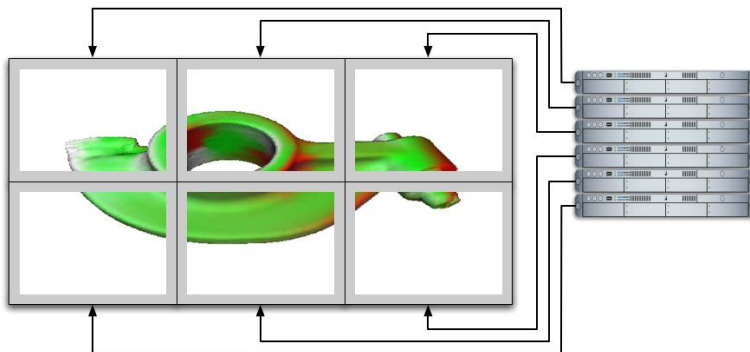


Abbildung 5: Display Wall, Quelle: <http://www.equalizergraphics.com/useCases.html>

Konfigurationen möglich wie Display Wall (oben) und CAVE Anwendungen (unten).

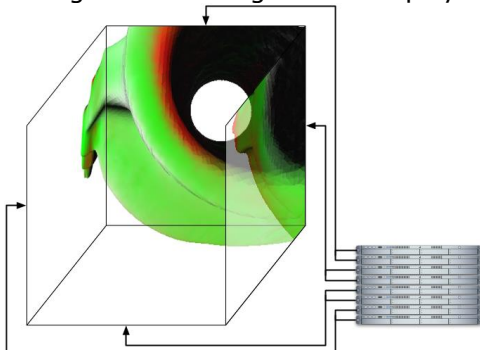


Abbildung 6: Vierseiten VR Installation, Quelle: <http://www.equalizergraphics.com/useCases.html>

## 4.4 Argumentation

### 4.4.1 Pro

#### 4.4.1.1 *Open Source, Dokumentation*

Equalizer ist Open Source. Es ist eine relativ aktuelle Dokumentation vorhanden.

#### 4.4.1.2 *Know-How BFH*

Durch verschiedene Projektarbeiten und eine bereits vorhandene Installation des gesamten Frameworks ist an der Berner Fachhochschule BFH in Biel Know-How vorhanden. Da wir aber eigenes Knowhow erarbeiten möchten, ist dieser Punkt sehr tief zu priorisieren.

(Wir möchten nicht das Rad nochmals neu erfinden mit der Anwendung des Equalizer Frameworks über beispielsweise ein Glut Interface)

### 4.4.2 Kontra

#### 4.4.2.1 *Unity3D Bezug*

Unser Projekt sollte klar auch eine Einarbeitung in Unity sein. Es sollte nicht die Hauptarbeit sein, Implementationen eines anderen Frameworks, Equalizer, anzupassen.

#### 4.4.2.2 *Architektur*

Equalizer hat durch seine Wrapperklassen eigentlich eine sehr starre Struktur vorgegeben. Man müsste im Unity sehr tief eingreifen, um diese Klassen einzubauen. Wir möchten nicht eine komplizierte Version, « einen Hack », herstellen, um dies auf Biegen und Brechen genauso und nur so einzubinden. Weiter ist unklar, ob über die Objektklassen alles andere von Unity (KI, Physik, etc) auch abbilden lässt.

## 4.5 Zusammenfassung

Basierend auf der kurzen Analyse der Dokumentation von Equalizer, unserer Projektanforderungen und den kurzen Pro- und Kontra Argumenten kommen wir zu folgenden Schlussfolgerungen:

Eine Verwendung des Equalizer Frameworks wäre denkbar, eventuell technisch sogar möglich, aber folgende Contra-Argumente wiegen zu schwer:

1. Wrapperklassen sind möglich im Unity3D. Da aber Equalizer OpenGL Aufbauend ist, fehlen uns wichtige Element wie für die Physik, KI, etc.  
So ist unser Ziel – ein Unity3D Projekt/Spiel/Techdemo einfach und bequem im CAVE anzubieten, nicht möglich.
2. Abschweifung von Unity3D hin zu C++.  
Wir nehmen an, dass der Grossteil der Arbeit/Prototypen dann direkt auf C++ Ebene durchgeführt werden müsste.  
Dies ist durchwegs denkbar, aber das Hauptaugenmerk für Unity3D (mit C#) würde somit verfehlt.

## 5 Eigene Lösung

### 5.1 Idee

Eine Variante besteht darin, auf Frameworks von Drittparteien zu verzichten und eine eigene Lösung zu entwickeln. Dieser Ansatz wird nur in Erwägung gezogen, falls sämtliche Möglichkeiten mit einem bestehenden System als unzureichend oder nicht umsetzbar eingestuft wurden.

Eines der Hauptprobleme wird die Synchronisierung des Servers und der verschiedenen Clients sein. Sobald Scripts Zufallskomponenten beinhalten, darf die Berechnung des Verlaufs des Spiels oder der Simulation nur zentral an einem Ort geschehen. Ansonsten sind die Stationen nicht mehr synchron und es kann kein einheitliches Bild mehr im CAVE dargestellt werden.

Zwei Ansätze sind in Betracht zu ziehen.

#### 5.1.1 Unity

Unity bietet Module, um Multiplayerspiele zu entwickeln. Das Problem der Synchronisierung tritt vor allem in diesem Bereich auf und ist somit ein zentrales Anliegen der Unity-Entwickler und Anwender. Auf deren Erfahrung und der bereits umgesetzten Module kann zurückgegriffen werden.

#### 5.1.2 Eigenes Protokoll

Falls Unity mit den Standardfunktionen zu wenige Möglichkeiten bietet und auch im Asset Store keine hilfreichen Bibliotheken vorhanden sind, müsste ein eigenes System zum Synchronisieren erarbeitet werden, welches die Kommunikation zwischen den Clients und dem Server unabhängig von Unity betreibt.

Zum Einsatz käme das User Datagram Protocol (UDP), welches die nötigen Informationen vom Hauptserver an die Clients schickt.

### 5.2 Architektur

Das geplante System basiert auf einer Client-Server Architektur. Die Hauptinstanz der Unity Anwendung wird auf einem eigenen Server laufen, welche den Rendering-Clients die nötigen Informationen zukommen lässt, um die Synchronisierung zu gewährleisten.

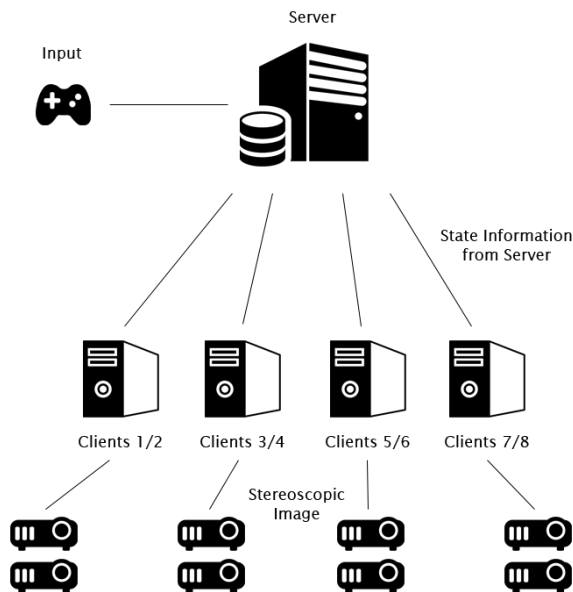


Abbildung 7: Client-Server Architektur, Quelle: <http://www.flaticon.com/>

### 5.2.1 Aufgabe Server

Die Hauptinstanz berechnet den Ablauf des Spiels oder der Simulation. User-Inputs werden hier verwaltet und entsprechende Geräte sind deshalb an diesem Rechner angeschlossen.

Nachdem die Anwendung gestartet wurde, ist sie bereit für die regelmässige Auslieferung der Informationen zum Synchronisieren der Clients und wartet auf deren Verbindungsaufbau.

Die Position der Kamera und der aktuelle State des Spiels oder der Simulation werden laufend an die Clients übermittelt, damit die Darstellung im CAVE entsprechend angepasst werden kann.

### 5.2.2 Aufgabe Client

Sobald der Server gestartet wurde, können sich die Clients beim Server anmelden und erhalten in regelmässigen Abständen Informationen über den aktuellen Status des Spiel-, bzw. Simulationsablaufs. Weil jeder Client für eine Sicht (ein Auge) auf eine Leinwand im CAVE zuständig ist, wird der Bildausschnitt dementsprechend reduziert und die Bildinformationen werden für die stereoskopische Darstellung auf einen Projektor geschickt. Jeder Client rendert jeweils nur eine Sicht der Anwendung, so dass zwei Clients zusammen die 3D Illusion, mit je einem eigenen Projektor, darstellen können.

Der Ablauf des Spiels oder der Simulation wird hier nicht berechnet. Lediglich die Informationen, die nötig sind um die virtuelle Welt darzustellen, werden empfangen und interpretiert.

## 5.3 Funktionale Anforderungen

### 5.3.1 Multi-Monitor Displays

Weil jede Leinwand nur einen Viertel des kompletten Bildes darstellt, liefert jeder einzelne Client selektiv Bildinformationen. Die erarbeitete Lösung sollte also in der Lage sein, basierend auf den Einstellungen des Clients, nur den entsprechenden Bildanteil zu projizieren.

### 5.3.2 Stereoskopie

Nicht-Stereoskopische Anwendungen sollten in Stereoskopische umgewandelt werden können, wobei passive (Polarisierte Lichtprojektion) Stereoskopie unterstützt wird. Unity unterstützt diese Funktion.

### 5.3.3 Implementierung bestehender Unity Anwendungen

Eine eigene Unity Anwendung in den CAVE zu implementieren sollte mit wenigen Clicks möglich sein. Die Rendering-Clients sind bereits für die jeweiligen Leinwände vorkonfiguriert. Es gilt lediglich, die erstellte Lösung in eine Unity 4.6 Anwendung mit vorhandenem Sourcecode einzubinden. Das Unity Projekt wird anschliessend für Windows exportiert und auf die Clients / den Server verteilt.

Nachdem die Anwendung auf dem Server gestartet wurde, können die Clients einzeln den Bootvorgang durchlaufen und sich mit dem Server verbinden, um die nötigen Informationen zur Darstellung der Projektion zu erhalten.

## 5.4 **Argumentation**

### 5.4.1 Pro

#### *5.4.1.1 Keine Abhängigkeit*

Weil das gesamte System eine Eigenentwicklung ist, bestehen bis auf die Verbindung zu Unity und der benutzten Infrastruktur keinerlei Abhängigkeiten. Die maximale Flexibilität ist somit gewährleistet. Ausserdem entfällt das Einarbeiten in ein bestehendes Framework.

#### *5.4.1.2 Community*

Unity ist ein sich stetig weiterentwickelndes Framework und geniesst eine immer grösser werdende Community, welche sich aktiv in Foren beteiligt. Sollten Probleme auftauchen, ist das Internet eine grosse Informationsquelle, die täglich an neuen Informationen reicher wird.

#### *5.4.1.3 Netzwerkauslastung*

Durch die Client-Server Architektur, bekannt aus Multiplayerspielen, die auch übers Internet gespielt werden, stellt das interne Netzwerk keinen Flaschenhals dar. Die nötigen Informationen für die Clients sind lediglich der Status des aktuellen Hauptspielablaufs.

### 5.4.2 Kontra

#### *5.4.2.1 From Scratch*

Weil auf keine Bibliothek oder Framework zurückgegriffen werden kann, muss alles von Grund auf selber programmiert werden. Lediglich die Funktionen von Unity können und sollten Verwendung finden.

#### *5.4.2.2 Aufwand*

Der Aufwand und die möglichen Probleme sind schwer abzuschätzen. Das gesamte System der Synchronisierung, der Stereoskopie und des Einpflegens in den CAVE muss geplant, umgesetzt und Debugged werden. Die Verwendung eines fertigen Frameworks, welches von einem Entwicklerteam stammt und sich in der Praxis bewährt hat, kann auf eine Robustheit zurückgreifen, die bei einer eigenen Entwicklung nicht per se gegeben ist.

## 5.5 **Zusammenfassung**

Einen eigenen Lösungsansatz zu verfolgen besticht durch seine Flexibilität. Mit der Game-Engine Unity wird auf das richtige Pferd gesetzt und ist zukunftsorientiert. Der Aufwand, ein eigenes Produkt zu erstellen ist im Gegensatz zur Implementierung eines fertigen Frameworks um einiges höher und birgt Gefahren. Deshalb ist die sorgfältige Analyse der bestehenden und geprüften Lösungen entscheidend. Erst wenn durch stichhaltige Argumente die anderen Möglichkeiten ausgeschlossen werden können, wird dieser Ansatz weiterverfolgt. Die Implementierung mittels einer eigenen Lösung ist aber, mit entsprechendem Aufwand, vielversprechend.



## 6 middleVR

### 6.1 Idee

Da Unity bereits auf einer breiten Community abgestützt ist, und diese auch fördert mit beispielsweise dem Asset Store, kann die Evaluation eines bestehenden Produktes auch in die Lösungsvorschläge fließen.

Jedoch möchten wir uns Abgrenzen von einer Analyse mehrerer Fremdsoftwares, und hier nur ein vielversprechendes Produkt vorstellen.

### 6.2 Warum middleVR?

middleVR (middleVR for Unity) hat uns dank seiner gut lesbaren Dokumentation und dessen Internetauftritt überzeugt, das Produkt konkreter zu beurteilen. Kurz und prägnant stellt es sich wie folgt vor:

middleVR erweitert Unity um folgende Funktionalitäten:

- Skalierung einer Visualisierung mit benutzerzentrierter Perspektive
- Support für 3D Interaktionsdevices wie 3D Trackers
- S3D – Aktive und Passive Stereoskopie
- Multiscreen- / Multicomputersynchronisierung für höhere Auflösungen und eindrucksvolle VR Systeme
- 3D Interaktionen: Navigation, Manipulation
- Immersive Menus
- Eigene grafische User Interfaces (in HTML5)
- Anzeige von beliebiger Website in der virtuellen Welt

Kreieren und erleben Sie interaktive & immersive VR Applikationen in wenigen Minuten dank dem simplen und mächtigen middleVR Plugin für Unity!

(Quelle: <http://www.middlevr.com/middlevr-for-unity/> )

### 6.3 Abdeckung middleVR

middleVR besteht aus zwei Hauptkomponenten:

- Vereinfachte Erstellung und Programmierung von VR Anwendungen
- Adaption auf verschiedenste VR Hardware und 3D Anwendungen

#### 6.3.1 Stereoskopie

middleVR weist darauf hin, wenn aktive Stereoskopie verwendet wird, dass eine Unity Pro – Lizenz vorhanden sein muss, und nur eine Handvoll GPUs unterstützt wird. (<http://www.middlevr.com/doc/current/#stereoscopy---s3d>)

Da der CAVE der BFH aber mit passiver Stereoskopie arbeitet, sollte dies uns hier nicht betreffen.

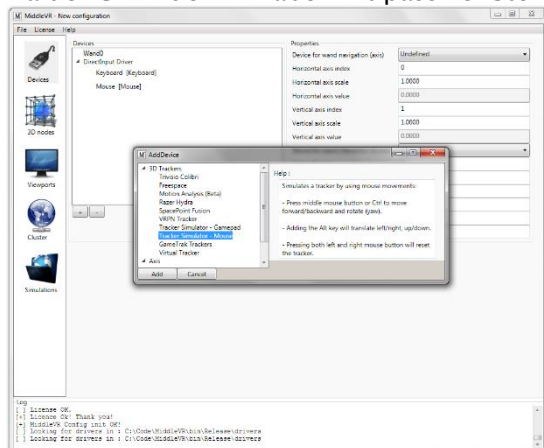


Abbildung 8: middleVR Konfigurator

### 6.3.2 Unsere funktionalen Anforderungen

Unsere definierten funktionalen Anforderungen (siehe Pflichtenheft), werden komplett abgedeckt, da es bereits als Unity Asset (Plugin) funktioniert.

### 6.3.3 Unsere nicht funktionalen Anforderungen

Bei den nicht funktionalen Anforderungen sind wir bei der Ergonomie, der Anwendung des Plugins, und der Wiederverwendbarkeit an middleVR gebunden.

## 6.4 Konzept middleVR

middleVR funktioniert wie folgt:

1. Beschreibung des VR Systems
2. Generierung der Konfiguration für das beschriebene VR System
3. Verwendung dieser Konfiguration durch das Unity Plugin

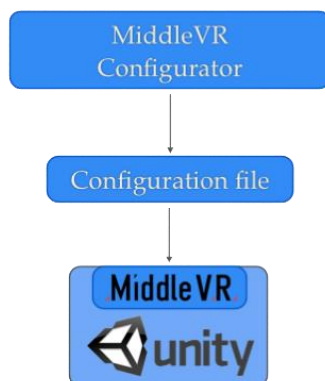


Abbildung 9: Basiskonzept middleVR

### 6.4.1 Beschreibung

Die Beschreibung umfasst folgende Punkte:

- Die Devices des VR Systems
- Eine Beschreibung, wie das diese Devices mit der realen Welt interagieren.  
(Beispielsweise Tracker A folgt dem Kopf des Benutzers, Tracker B folgt dessen linker Hand)
- Die Positionen der Screens
- Welche Kameras was wo rendern müssen

#### 6.4.1.1 *Beispiel*

Die Beispielkonfiguration umfasst einen 5-Seiten CAVE mit 5 Clustern.

(Anmerkung: das Setting in der BFH wird sich unterscheiden bei den Anzahl Clustern (1 Cluster pro Viewport), sowie der Ausrichtung (hier von Innen nach Aussen), und natürlich den ganzen Parametern.

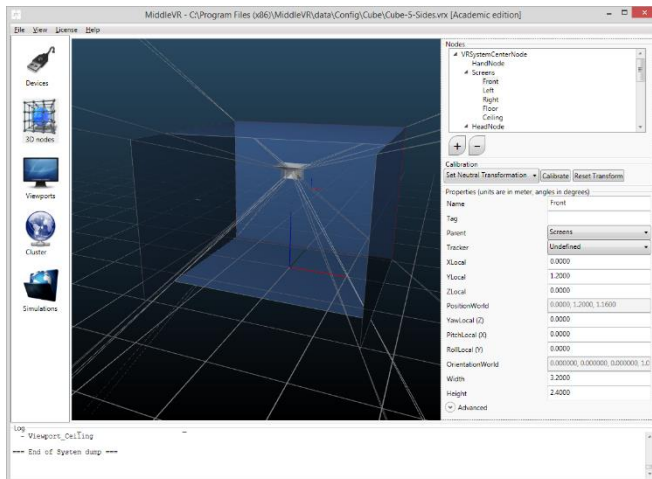


Abbildung 10: Beispielskonfiguration CAVE

## 6.5 Argumentation

### 6.5.1 Pro

#### 6.5.1.1 Bereits ausgearbeitete Software

Da es sich um eine professionelle Software handelt, würde viel oder fast alles an Programmierung ausserhalb von Unity entfallen. Ob Scripts (Kamera oder Assets) angepasst oder selbst geschrieben werden müssen, oder eventuell sogar ein eigener Konfigurator (als Alternative zum middleVR Konfigurator) erstellt werden müsste, ist unbekannt.

#### 6.5.1.2 Aufwand

Da alle nötigen Parameter bereits vorhanden sind für den CAVE zu konfigurieren, fällt ein grosser Teil des Designs weg. Dies aber nur im optimalen Fall, wenn das komplette middleVR verwendet werden kann.

#### 6.5.1.3 Support

middleVR ist ein Produkt, welches in mehreren Lizenzmodellen kommt. Somit wäre auch ein aktueller Support vorhanden.

### 6.5.2 Kontra

#### 6.5.2.1 Abhängigkeit

Mit dieser Lösung sind wir fast oder komplett an middleVR gebunden. Falls während der Prototyping-phase unüberbrückbare Probleme auftreten, könnte das Projekt nicht realisiert werden.

#### 6.5.2.2 Eigenanteil

Für den Stolz der CPVR-Studenten ist es auch wichtig, einen signifikanten Eigenanteil beizusteuern. Im günstigsten Fall funktioniert die Software nach kurzer Konfiguration.

## 6.6 Zusammenfassung

middleVR bietet sich als Lösung an. Es kann jedoch erst beim Prototyping entschieden werden, ob die vorhandenen Features, Möglichkeiten und Dokumentationen ausreichend sind, um den CAVE der BFH abzubilden.

## 7 Gegenüberstellung

Basierend auf den Analysen wurden die Pros und Kontras zusammengefasst, um einen Vergleich aufzustellen. Die verschiedenen Kategorien werden kurz erläutert und mit einer Relevanz gewichtet.



|   |               | Support  |     | Keyfeatures  |     | Verbindung zu Unity   |     | Architektur                                  |     |
|---|---------------|--|-----|--|-----|---|-----|--|-----|
|    | Chromium      |  | 0   | - Stereoskopie<br>- Aufteilung der Monitore                | 0.7 | Keine   | 0.1 | Gegebene Client / Server Struktur im CAVE    | 0.3 |
|   | Equalizer     | Dank Open Source vorhandene Community            | 0.5 | Graphikklassen vorhanden (KI & Physik nicht)               | 0.3 | Keine   | 0.2 | Vorhandene Graphikklassen                    | 0.3 |
|   | Eigene Lösung | Viele aktive Unity Entwickler kontaktierbar      | 0.3 | Mit entsprechendem Aufwand nahezu alles machbar            | 0.8 | Umsetzung direkt in Unity                                     | 0.8 | Gut an CAVE adaptierbar                      | 0.5 |
|   | middleVR      | Professionelle und aktuelle Software             | 0.8 | - Stereoskopie<br>- Aufteilung der Monitore<br>- Tracking  | 1   | Als Package für Unity vorhanden                               | 0.8 | Auf Unity basierend                          | 0.9 |
|  | Chromium      | Eingestellt                                      | 1   | - Kein Tracking<br>- Mittels Unity keinen grossen Einfluss | 0.3 | - Kein direkter Unity Support<br>- OpenGL nur bis Version 1.5 | 0.9 | Engpass Netzwerkauslastung                   | 0.7 |
|   | Equalizer     | Open Source                                      | 0.5 | Wahrscheinlich grosser Teil nicht vorhanden                | 0.7 | - Kein direkter Unity Support<br>- C++ Implementation         | 0.8 | Starre Struktur gegeben durch Wrapperklassen | 0.7 |
|   | Eigene Lösung | Auf eigenes Unity-Wissen angewiesen              | 0.7 | Nur mit entsprechendem Aufwand möglich                     | 0.2 | Ausschliesslich Möglichkeiten von Unity                       | 0.2 | Eigene Architektur notwendig                 | 0.5 |
|   | middleVR      | Nur über Internet und mit höherer Latenz möglich | 0.2 |  | 0   | Geknüpft an Unity Version / Features                          | 0.2 | Geringe Flexibilität                         | 0.1 |

Tabelle 2: Gegenüberstellung Pro und Kontra

Daraus ergeben sich folgende Diagramme.

(Info: Falls eine Kategorie nicht genau bekannt war, wurde mit der vorhandenen Erfahrung geschätzt)

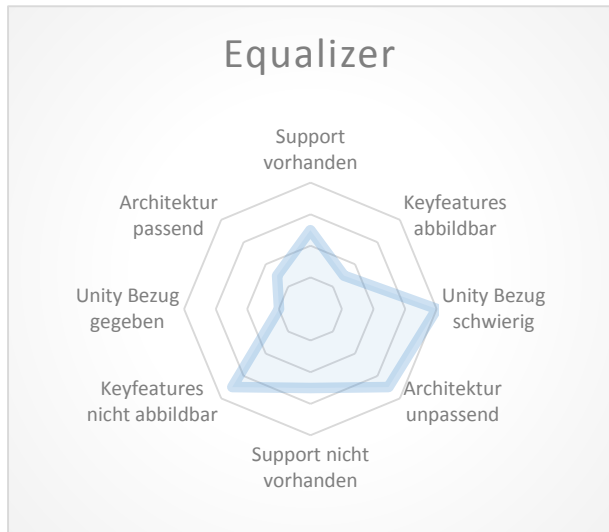


Abbildung 14: Pro-Kontra Grid Equalizer

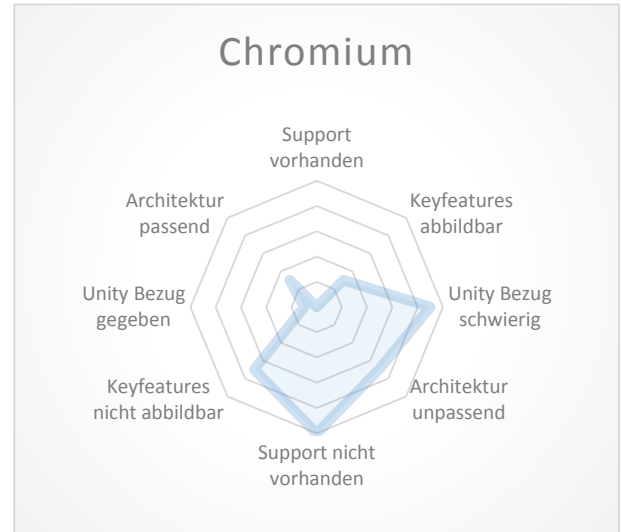


Abbildung 14: Pro-Kontra Grid Chromium



Abbildung 14: Pro-Kontra Grid Eigene Lösung

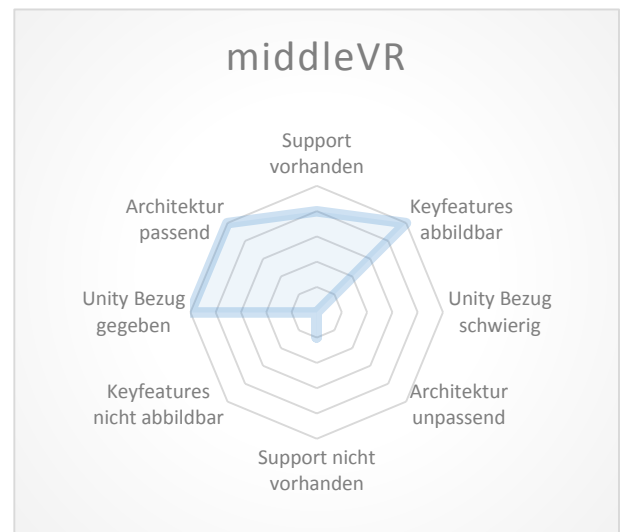


Abbildung 14: Pro-Kontra Grid middleVR

## 8 Entscheid

Ausgehend von den Pros und Kontras der verschiedenen Möglichkeiten hat sich ein klarer Trend abgezeichnet. Auf Kosten der grösstmöglichen Flexibilität, welche sich bei einer eigenen Umsetzung ergeben hätte, konnte sich MiddleVR klar von den übrigen Methoden abheben und bietet eine sichere, geprüfte und supportete Lösung um Unity in den CAVE zu integrieren.

Dank den Analysen konnten mögliche Probleme weitgehend ausgeschlossen werden, bei der Umsetzung der ersten Prototypen besteht aber nach wie vor die Gefahr, auf ungeahnte und unlösbare Probleme zu stossen. Darum muss als Fallback-Plan auch eine zweite Möglichkeit nicht aus den Augen gelassen werden und schnell reagieren zu können.

Mit Absprache unseres Betreuers wird die Variante middleVR verfolgt und in einem nächsten Schritt erfolgt die Auswahl und Umsetzung der Prototypen.

## 9 middleVR Prototyping

### 9.1 Prototypen

Ein wichtiger Faktor bei middleVR ist, je nach Setting, die Synchronisierung der verschiedenen Clients, bzw. GPUs. Um möglichst viele Fälle zu testen, werden Prototypen mit unterschiedlichen Anforderungen an middleVR und den CAVE integriert.

Kritische Punkte bezüglich Synchronisierung bei der Darstellung mit middleVR sind unter anderem:

- Die Kamera
- Dynamischen Objekte
- Shader-Effekte
- Grosser Datentransfer über das Netzwerk

Mit den Prototypen soll eruiert werden, wo möglicherweise Abstriche gemacht werden müssen und welche Features von Unity problemlos integriert werden können.

#### 9.1.1 Basic

Der Basic-Prototyp dient lediglich als Basisanwendung, um die grundlegende Lauffähigkeit von middleVR testen zu können. Bei der Synchronisierung soll kein grosser Datentransfer entstehen.

In der statischen virtuellen Welt befindet sich nur ein Cube, eine Ebene als Terrain ohne jegliches Material und ein First-Person-Controller, um die Transformationen des Cubes und der rudimentären Landschaft zu visualisieren. Ein Directional Light dient als Lichtquelle.

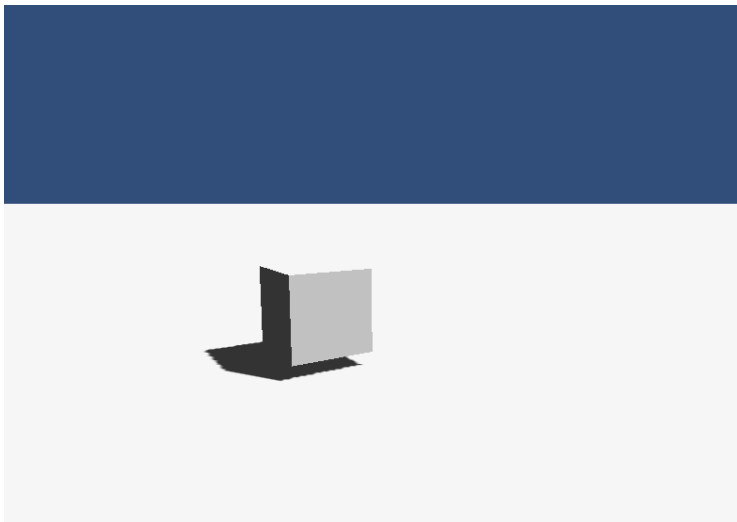


Abbildung 15: Basic-Prototyp

#### 9.1.2 AngryBots

Mit Unity wurde bis und mit Version 4 das Beispielprojekt AngryBots mitgeliefert, welcher einen geeigneten Showcase der Möglichkeiten von Unity bietet. Etliche Effekte, detaillierte Models, Kampfhandlungen und weitere Keyfeatures von Games sind darin enthalten.

Weil middleVR zum aktuellen Zeitpunkt Unity5 nur mit einem Beta-Release unterstützt, wird mit diesem Prototypen noch auf Unity4 gesetzt.



Abbildung 16: AngryBots, Quelle: <http://www.androidmag.de/>

Sämtliche kritischen Punkte, mit Hauptaugenmerk auf visuellen Effekten und der Kamera, können mittels diesem Prototypen eruiert werden und ist gewissermassen ein Härtetest für middleVR und vertritt optimal die Bedingungen eines realen Games, wie es später eingesetzt werden kann.

### 9.1.3 Car Tutorial

Ein weiteres Beispielprojekt ist ein Autorennspiel, genannt « Car Tutorial ». Es basiert auf Unity 3.3. Dieses Tutorial besitzt ein realistisches Autofahrverhalten und eine verbesserte Physik.



Abbildung 17: Car Tutorial, Quelle: <http://blogs.unity3d.com/2010/04/24/car-tutorial/>

Das Hauptaugenmerk liegt wie erwähnt auf der Physik, sowie den speziellen Wheelcollidern.



## 9.2 Variantenentscheid

Mit middleVR sind verschiedene Varianten denkbar, um einen CAVE zu implementieren. Jede dieser Methoden hat gewisse Vor- und Nachteile bezüglich Aufwand, Performance, Infrastruktur und Lizenzierungskosten. Nun gilt es in einem nächsten Schritt die Varianten gegenüberzustellen, um eine gute Basis für den anschliessenden Entscheid zu schaffen. Ein wichtiger Faktor sind die entstehenden Kosten. Diese können durch Aufrüstung der Infrastruktur oder Lizenzierung entstehen und es muss abgeschätzt werden, ob die Mehrkosten zu den entstandenen Vorteilen tragbar sind.

Die aktuelle Ausgangslage bietet drei verschiedene middleVR-Varianten an, welche detailliert eruiert werden.

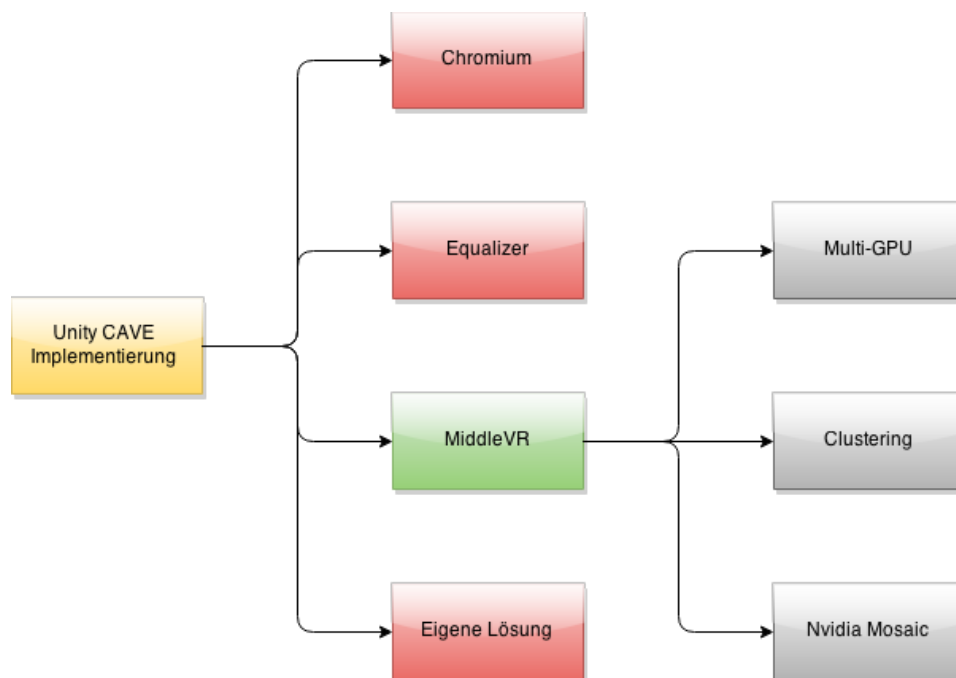


Abbildung 18: Evaluierung middleVR Varianten, Quelle: <http://draw.io>

## 9.3 Aktuelle Infrastruktur

Mit dem aktuellen Setting des CAVEs ist die Art der Implementierung von middleVR gegeben. Es sind 8 Clients und 1 Server im Einsatz, je mit einer Grafikkarte mit 2 Outputs. Nur einer dieser Outputs führt zu einem Beamer, der andere dient als Admin-Konsole. So können zur Laufzeit, ohne störende Unterbrüche auf der CAVE-Leinwand, Adaptionen vorgenommen werden.

Das bedeutet, dass auf jedem Client eine Unity- und middleVR-Instanz laufen muss und über das Netzwerk erfolgt die Synchronisierung mittels Clustering. Ein Client ist somit zuständig für einen Teil der stereoskopischen Projektion (Z.B. das Bild für das linke Auge auf der linken Leinwand).

Die nachfolgend diskutierten Ansätze haben einen möglichen Einfluss auf die aktuelle Infrastruktur. Um aber auch ohne Anpassungen des Settings eine Implementierung sicherstellen zu können, wird unabhängig vom schlussendlich gewählten Ansatz ein Prototyp dieser Art umgesetzt. MiddleVR setzt bei der Verwendung eines Cluster-Systems eine Academic oder Pro Lizenz voraus.

**Damit keine unnötigen Kosten entstehen, wird in einem Zeitfenster von 30 Tagen die Testversion gelöst und ein Prototyp eingebaut.**

## 9.4 Variante Multi-GPU

MiddleVR bietet die Möglichkeit, 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 die GPUs müssen intern abgeglichen werden. 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.

*„Nevertheless the Multi-GPU option comes at a price because it forces the use of clustering and so to manage its difficulties“ (Zitat middleVR Dokumentation, Quelle: <http://www.middlevr.com/doc/current/>).*

Jede GPU hat eine eigene Unity-Instanz am Laufen. Erste Prototypen haben zu Tage gebracht, dass die Synchronisierung je nach Feature Probleme verursachen kann und Anpassungen am Quellcode erfordert werden.

### 9.4.1 Lizenzen

Pro GPU muss eine middleVR Academic-, bzw. Pro-Lizenz gelöst werden.

| Komponente                | Anzahl | Preis pro Stück € | Preis gesamt €  |
|---------------------------|--------|-------------------|-----------------|
| middleVR Academic Licence | 4      | 3000.-            | 12'000.-        |
|                           |        |                   | <b>12'000.-</b> |

Tabelle 3: Geschätzte Kosten middleVR Lizenzen Multi-GPU

### 9.4.2 Adaption Infrastruktur

Ein komplett neuer Rechner mit 4 Grafikkarten muss angeschafft und am CAVE angeschlossen werden.

| Komponente                         | Anzahl | Preis pro Stück CHF | Preis gesamt CHF |
|------------------------------------|--------|---------------------|------------------|
| Grafikkarte - GeForce GTX 760      | 4      | 260.-               | 1040.-           |
| RAM - 16GB DDR3-1600               | 1      | 120.-               | 120.-            |
| Mainboard - ASUS Z9PE-D8 WS        | 1      | 439.-               | 439.-            |
| CPU - Intel Core i7 4820K          | 2      | 347.-               | 694.-            |
| SSD - Samsung 850 EVO Basic 500 GB | 1      | 214.-               | 214.-            |
| Server Case                        | 1      | 200.-               | 200.-            |
| Netzteil                           | 1      | 150.-               | 150.-            |
| Verkabelung                        |        |                     | 100.-            |
|                                    |        |                     | <b>2957.-</b>    |

Tabelle 4: Geschätzte Kosten Infrastruktur Multi-GPU, Quelle: <http://digitec.ch> (19.04.2015)

### 9.4.3 Vorteile

- Maximale Performance dank Verwendung mehrerer GPUs
- Keine Netzwerk-Einschränkungen
- Ein einzelner Rechner

### 9.4.4 Nachteile

- Anschaffung neuer Komponenten
- Synchronisierung mehrerer Unity-Instanzen
- Lösen mehrerer Lizenzen

- Inbetriebnahme eines neuen Systems in den CAVE
- Relativ hohe Lizenzkosten

#### 9.4.5 Fazit

Das primäre Ziel der Arbeit ist nicht ein hochperformanter CAVE, um sehr rechenintensive Anwendungen eindrucksvoll darstellen zu können, sondern vielmehr ein Showroom, um Spiele, die hauptsächlich am eigenen Computer Zuhause Verwendung finden, stereoskopisch darzustellen. Der dominante Vorteil, nämlich die maximale Performance, wird in unserem Projekt dementsprechend degradiert und hat nicht die stärkste Gewichtung. Denn kann ein Unity-Spiel am eigenen Rechner flüssig gespielt werden, reicht im CAVE eine einzelne Grafikkarte theoretisch auch.

Die zu tragenden Kosten sind verhältnismässig hoch und falls nur ein Rechner zum Einsatz kommt, ist es wünschenswert, die Problematik der Synchronisierung eliminieren zu können, was da leider nicht der Fall ist.

### 9.5 Variante Cluster

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.

Unity unterscheidet zwischen zwei verschiedenen Konzepten:

|                                  |   |
|----------------------------------|---|
| <u>Authorative Server:</u>       | Verwalten die gesamte Spielszene, Clients nehmen keine Änderungen vor. Sie senden diese Änderungen (Tastatureingaben, etc) an den Server. |
| <u>Nicht authorative Server:</u> | Alle Clients berechnen ihre Spielwelt selbst und bekommen nur neue Positionen für die Objekte von den anderen Klienten.                   |

Diese verschiedenen Architekturen müssen bei den Prototypen berücksichtigt werden.

### 9.5.1 Lizenzen

Pro Cluster muss eine middleVR Academic-, bzw. Pro-Lizenz gelöst werden. Je nach Serversystem muss noch eine zusätzliche Lizenz gelöst werden.

| Komponente                | Anzahl | Preis pro Stück € | Preis gesamt €  |
|---------------------------|--------|-------------------|-----------------|
| middleVR Academic Licence | 8      | 3000.-            | 24'000.-        |
| middleVR Academic Licence | 1      | 3000.-            | 3000.-          |
|                           |        |                   | <b>27'000.-</b> |

Tabelle 5: Geschätzte Kosten middleVR Lizenzen Cluster 8 Clients

| Komponente                | Anzahl | Preis pro Stück € | Preis gesamt €  |
|---------------------------|--------|-------------------|-----------------|
| middleVR Academic Licence | 4      | 3000.-            | 12'000.-        |
| middleVR Academic Licence | 1      | 3000.-            | 3000.-          |
|                           |        |                   | <b>15'000.-</b> |

Tabelle 6: Geschätzte Kosten middleVR Lizenzen Cluster 4 Clients

### 9.5.2 Vorteile

- Aktuelle Infrastruktur muss nicht verändert werden (Abgesehen von eigenen Harddisks)
- Skalierbar  
Es können auch nur 4 Clients eingesetzt werden
- Keine Limitierung bei CPUs  
Jede Unity Instanz hat vollen Zugriff auf den CPU, welchen es u.U. benötigt.

### 9.5.3 Nachteile

- Kosten
- Netzwerksynchronisierung  
(Kann u.U. aufwändig werden, aber prinzipiell möglich)

### 9.5.4 Fazit

Da diese Variante keine Umbauarbeiten benötigt, wird sie auch für das Prototyping verwendet. Technisch sicherlich nicht die trivialste Variante, aber Unity und middleVR sind für solche Settings ausgelegt. Weiter können sich hier doch noch einige Fehlerquellen offenbaren.

Wie ist das Netzwerksetting genau im CAVE? Sind die aktuellen Rechner und Graphikkarten 100% kompatibel? In unseren Augen sind aber die Lizenzkosten der springende Punkt. Dies entzieht sich aber unserer Projektarbeit und wird nur präsentiert.

## 9.6 Variante Nvidia Mosaic

### 9.6.1 Was ist Nvidia Mosaic?

Nvidia Mosaic ist eine Technologie, um mehrere Graphikkarten zu verlinken und auf einem Desktop darzustellen.

„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> )

Der Treiber simuliert dann dem System nur einen GPU/Viewport vor, der selbst angepasst werden kann.

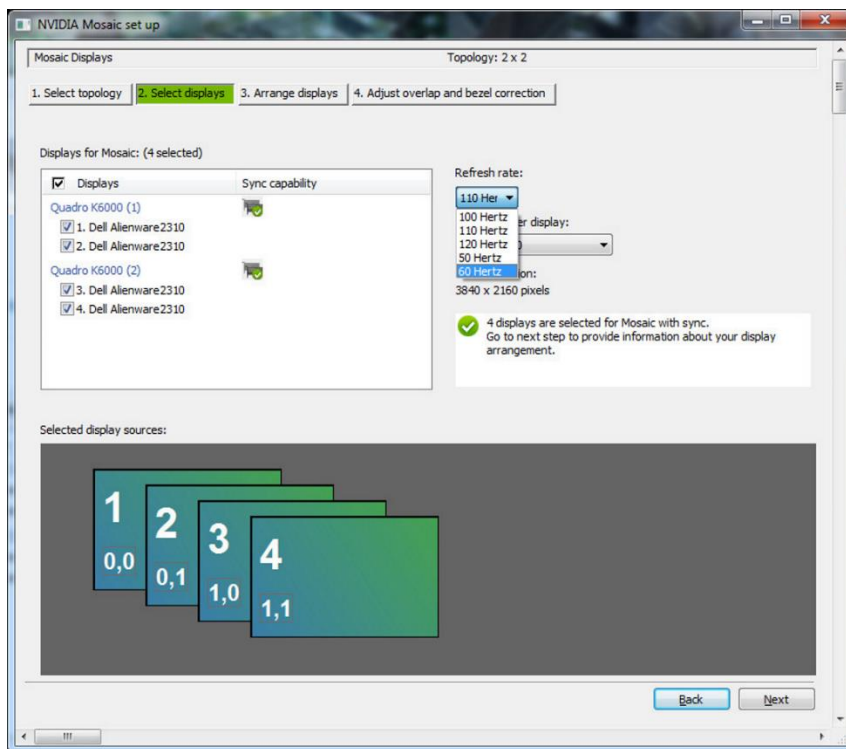


Abbildung 19: Nvidia Mosaic Einstellungen, Quelle: <http://nvidia.custhelp.com/rnt/rnw/img/enduser/3568-7.png>

## 9.6.2 Limitierungen

Aus dem Factsheet von Nvidia entnehmen wir folgende Limitierungen:

1. Auflösung  
(Quadro K5000/5000/6000, Quadro Plex 7000)  
Maximal 16384\*16384 Pixel. (Dies entspricht 4x einer 4K Auflösung horizontal und vertikal)
2. Maximale Displayanzahl  
(Quadro K5000)  
Maximal 16 Displays mit vier Quadrokarten.

(Quelle: [http://www.nvidia.com/docs/IO/40049/NVMosaic\\_UC\\_v05.pdf](http://www.nvidia.com/docs/IO/40049/NVMosaic_UC_v05.pdf) )

## 9.6.3 Kosten und Lizenzen

Quadrokomponenten müssten zusätzlich angeschafft werden. Hier ein Beispielsetting:

| Komponente                        | Anzahl | Preis pro Stück CHF | Preis gesamt CHF |
|-----------------------------------|--------|---------------------|------------------|
| Grafikkarte - Nvidia Quadro K5000 | 4      | 1797.-              | 7188.-           |
| RAM – 16GB DDR3-1600              | 1      | 120.-               | 120.-            |
| Mainboard - ASUS Z9PE-D8 WS       | 1      | 439.-               | 439.-            |

|                                    |   |       |               |
|------------------------------------|---|-------|---------------|
| CPU - Intel Core i7 4820K          | 2 | 347.- | 694.-         |
| SSD - Samsung 850 EVO Basic 500 GB | 1 | 214.- | 214.-         |
| Server Case                        | 1 | 200.- | 200.-         |
| Netzteil                           | 1 | 150.- | 150.-         |
| Verkabelung                        |   |       | 100.-         |
|                                    |   |       | <b>9105.-</b> |

Tabelle 7: Nvidia Quadro K5000 Beispielaufbau

| Komponente                | Anzahl | Preis pro Stück € | Preis gesamt € |
|---------------------------|--------|-------------------|----------------|
| middleVR Academic Licence | 1      | 3000.-            | 3000.-         |
|                           |        |                   | <b>3'000.-</b> |

Tabelle 8: Nvidia Quadro K5000 middleVR Lizenz

#### 9.6.4 Vorteile

- Nur eine middleVR Lizenz nötig (1 GPU, resp. Rechner)
- Konfigurierbar

#### 9.6.5 Nachteile

- Neue Hardware benötigt (Nvidia Quadro GPUs)
- Kann ein Würfel dargestellt werden?

#### 9.6.6 Fazit

Eine Nvidia Mosaic Lösung wäre denkbar, man würde sich sogar ganz vom herausfordernden Clustersystem verabschieden. Schwierigkeiten könnte es allenfalls mit dem Seitenverhältnis des Würfels geben. Es stellt sich die Frage, ob beim nVidia Mosaic Treiber die notwendigen Einstellungen vorgenommen werden können. Die Dokumentationen zeigen jeweils nur „Wall-Settings“. Laut Aussage von middleVR sollte diese Variante jedoch möglich sein, aber es ist mit Performanceeinbussen zu rechnen, weil (wiederum laut middleVR) angeblich nur eine Grafikkarte die Berechnungen vornimmt und mit Hilfe der „sekundären“ GPUs das Bild schlussendlich dargestellt wird. Diese Aussage deckt sich aber nicht mit der Dokumentation von Nvidia.

## 10 Abbildungsverzeichnis

|  |    |
|--|----|
| Abbildung 1: Architektur von Chromium mit Mothership und SUPs Quelle:<br><a href="http://chromium.sourceforge.net/doc/index.html">http://chromium.sourceforge.net/doc/index.html</a>   | 7  |
| Abbildung 2: Ansicht der Leinwände in vier Himmelsrichtungen, Quelle:<br><a href="http://chromium.sourceforge.net/presentations/SantaFe-BrianPaul/siframes.html">http://chromium.sourceforge.net/presentations/SantaFe-BrianPaul/siframes.html</a> | 8  |
| Abbildung 3: Implementation eigener Rendering Styles, Quelle:<br><a href="http://chromium.sourceforge.net/doc/index.html">http://chromium.sourceforge.net/doc/index.html</a>   | 8  |
| Abbildung 4: Equalizer Beispiel, Quelle:<br><a href="http://www.equalizergraphics.com/documents/Developer/eqPly.pdf">http://www.equalizergraphics.com/documents/Developer/eqPly.pdf</a>  | 11 |
| Abbildung 5: Display Wall, Quelle: <a href="http://www.equalizergraphics.com/useCases.html">http://www.equalizergraphics.com/useCases.html</a>   | 12 |
| Abbildung 6: Vierseiten VR Installation, Quelle: <a href="http://www.equalizergraphics.com/useCases.html">http://www.equalizergraphics.com/useCases.html</a>   | 12 |
| Abbildung 7: Client-Server Architektur, Quelle: <a href="http://www.flaticon.com/">http://www.flaticon.com/</a>  | 15 |
| Abbildung 8: middleVR Konfigurator   | 18 |
| Abbildung 9: Basiskonzept middleVR   | 18 |
| Abbildung 10: Beispielskonfiguration CAVE  | 19 |
| Abbildung 14: Pro-Kontra Grid Chromium   | 21 |
| Abbildung 14: Pro-Kontra Grid Eigene Lösung  | 21 |
| Abbildung 14: Pro-Kontra Grid middleVR   | 21 |
| Abbildung 14: Pro-Kontra Grid Equalizer  | 21 |
| Abbildung 15: Basic-Prototyp   | 23 |
| Abbildung 16: AngryBots, Quelle: <a href="http://www.androidmag.de/">http://www.androidmag.de/</a>   | 24 |
| Abbildung 17: Car Tutorial, Quelle: <a href="http://blogs.unity3d.com/2010/04/24/car-tutorial/">http://blogs.unity3d.com/2010/04/24/car-tutorial/</a>  | 24 |
| Abbildung 18: Evaluierung middleVR Varianten, Quelle: <a href="http://draw.io">http://draw.io</a>  | 25 |
| Abbildung 19: Nvidia Mosaic Einstellungen, Quelle:<br><a href="http://nvidia.custhelp.com/rnt/rnw/img/enduser/3568-7.png">http://nvidia.custhelp.com/rnt/rnw/img/enduser/3568-7.png</a>  | 29 |

## 11 Tabellenverzeichnis

|  |    |
|--|----|
| Tabelle 1: Wichtige Equalizer Klassen  | 12 |
| Tabelle 2: Gegenüberstellung Pro und Kontra  | 20 |
| Tabelle 3: Geschätzte Kosten middleVR Lizenzen Multi-GPU   | 26 |
| Tabelle 4: Geschätzte Kosten Infrastruktur Multi-GPU, Quelle: <a href="http://digitec.ch">http://digitec.ch</a> (19.04.2015) | 26 |
| Tabelle 5: Geschätzte Kosten middleVR Lizenzen Cluster 8 Clients   | 28 |
| Tabelle 6: Geschätzte Kosten middleVR Lizenzen Cluster 4 Clients   | 28 |
| Tabelle 7: Nvidia Quadro K5000 Beispielaufbau  | 30 |
| Tabelle 8: Nvidia Quadro K5000 middleVR Lizenz   | 30 |

## 12 Glossar

|   |    |
|---|----|
| <b>Auinweon</b><br>Et ut aut isti repuditis qui ium | 7  |
| <b>Batnwpe</b><br>Et ut aut isti repuditis qui ium  | 9  |
| <b>Cowoll</b><br>Et ut aut isti repuditis qui ium   | 11 |

## 13 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

## 14 Anhang

## 15 Versionskontrolle

| Version | Datum      | Beschreibung          | Autor                             |
|---------|------------|-----------------------|-----------------------------------|
| 1.00    | 24.03.2015 | Dokument erstellt     | Daniel Inversini                  |
| 1.01    | 25.03.2015 | Dokument überarbeitet | Daniel Inversini                  |
| 2.00    | 20.04.2015 | Dokument erweitert    | Daniel Inversini, Julien Villiger |