|  |
| --- |
| Gegenüberstellung Analysen, Variantenentscheid  **Daniel Inversini**  **V1.01 25.03.2015** |
| **Berner Fachhochschule**  Technik und Informatik  Informatik |

Inhaltsverzeichnis

[1 Voranalyse Varianten 4](#_Toc415061289)

[2 Chromium 4](#_Toc415061290)

[2.1 Architektur 4](#_Toc415061291)

[2.2 Anwendung 5](#_Toc415061292)

[2.2.1 Multi-Monitor Displays 5](#_Toc415061293)

[2.2.2 Delegation 5](#_Toc415061294)

[2.2.3 Manipulation 5](#_Toc415061295)

[2.2.4 Stereoskopie 5](#_Toc415061296)

[2.2.5 Command Stream Aufteilung 5](#_Toc415061297)

[2.3 Argumentation 6](#_Toc415061298)

[2.3.1 Pro 6](#_Toc415061299)

[2.3.1.1 Stereoskopie 6](#_Toc415061300)

[2.3.1.2 Infrastruktur 6](#_Toc415061301)

[2.3.1.3 Aufteilung der Monitore 6](#_Toc415061302)

[2.3.2 Kontra 6](#_Toc415061303)

[2.3.2.1 Plattform 6](#_Toc415061304)

[2.3.2.2 Maintenance 6](#_Toc415061305)

[2.3.2.3 OpenGL Support 6](#_Toc415061306)

[2.3.2.4 Kompatibilität Unity3D 6](#_Toc415061307)

[2.3.2.5 Netzwerkauslastung 6](#_Toc415061308)

[2.3.2.6 Community 7](#_Toc415061309)

[2.4 Zusammenfassung 7](#_Toc415061310)

[3 Equalizer – The parallel rendering framework 8](#_Toc415061311)

[3.1 Warum Voranalyse Equalizer? 8](#_Toc415061312)

[3.2 Architektur 8](#_Toc415061313)

[3.3 Anwendung 9](#_Toc415061314)

[3.4 Argumentation 10](#_Toc415061315)

[3.4.1 Pro 10](#_Toc415061316)

[3.4.1.1 Open Source, Dokumentation 10](#_Toc415061317)

[3.4.1.2 Know-How BFH 10](#_Toc415061318)

[3.4.2 Kontra 10](#_Toc415061319)

[3.4.2.1 Unity3D Bezug 10](#_Toc415061320)

[3.4.2.2 Architektur 10](#_Toc415061321)

[3.5 Zusammenfassung 10](#_Toc415061322)

[4 Eigene Lösung 11](#_Toc415061323)

[4.1 Idee 11](#_Toc415061324)

[4.1.1 Unity 11](#_Toc415061325)

[4.1.2 Eigenes Protokoll 11](#_Toc415061326)

[4.2 Architektur 11](#_Toc415061327)

[4.2.1 Aufgabe Server 12](#_Toc415061328)

[4.2.2 Aufgabe Client 12](#_Toc415061329)

[4.3 Funktionale Anforderungen 12](#_Toc415061330)

[4.3.1 Multi-Monitor Displays 12](#_Toc415061331)

[4.3.2 Stereoskopie 12](#_Toc415061332)

[4.3.3 Implementierung bestehender Unity Anwendungen 12](#_Toc415061333)

[4.4 Argumentation 13](#_Toc415061334)

[4.4.1 Pro 13](#_Toc415061335)

[4.4.1.1 Keine Abhängigkeit 13](#_Toc415061336)

[4.4.1.2 Community 13](#_Toc415061337)

[4.4.1.3 Netzwerkauslastung 13](#_Toc415061338)

[4.4.2 Kontra 13](#_Toc415061339)

[4.4.2.1 From Scratch 13](#_Toc415061340)

[4.4.2.2 Aufwand 13](#_Toc415061341)

[4.5 Zusammenfassung 13](#_Toc415061342)

[5 middleVR 14](#_Toc415061343)

[5.1 Idee 14](#_Toc415061344)

[5.2 Warum middleVR? 14](#_Toc415061345)

[5.3 Abdeckung middleVR 14](#_Toc415061346)

[5.3.1 Stereoskopie 14](#_Toc415061347)

[5.3.2 Unsere funktionalen Anforderungen 15](#_Toc415061348)

[5.3.3 Unsere nicht funktionalen Anforderungen 15](#_Toc415061349)

[5.4 Konzept middleVR 15](#_Toc415061350)

[5.4.1 Beschreibung 15](#_Toc415061351)

[5.4.1.1 Beispiel 16](#_Toc415061352)

[5.5 Argumentation 16](#_Toc415061353)

[5.5.1 Pro 16](#_Toc415061354)

[5.5.1.1 Kein From Scratch 16](#_Toc415061355)

[5.5.1.2 Aufwand 16](#_Toc415061356)

[5.5.1.3 Support 16](#_Toc415061357)

[5.5.2 Kontra 16](#_Toc415061358)

[5.5.2.1 Abhängigkeit 16](#_Toc415061359)

[5.5.2.2 Eigenanteil 16](#_Toc415061360)

[5.6 Zusammenfassung 16](#_Toc415061361)

[6 Gegenüberstellung 17](#_Toc415061362)

[7 Entscheid 18](#_Toc415061363)

[8 Abbildungsverzeichnis 18](#_Toc415061364)

[9 Tabellenverzeichnis 18](#_Toc415061365)

[10 Glossar 18](#_Toc415061366)

[11 Literaturverzeichnis 18](#_Toc415061367)

[12 Anhang 19](#_Toc415061368)

[13 Versionskontrolle 19](#_Toc415061369)

# Voranalyse Varianten

Dieses Dokument dient dazu, unsere vier Möglichkeiten zusammen zu betrachten und gegenüberzustellen. Darauf beruht dann der Variantenentscheid, welcher am Schluss des Dokumentes zu finden ist.

# Chromium

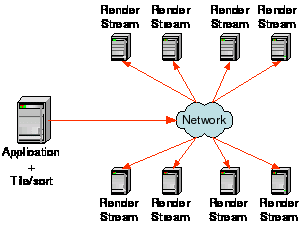
## Architektur

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 weiterschicken

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.

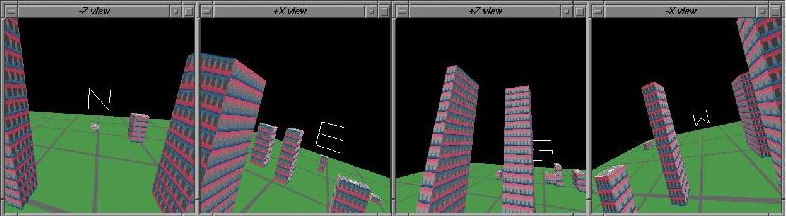
Die Chromium-Instanz läuft auf dem sogenannten „Mothership“ und managed die SPU Kette und Netzwerkverbindungen. Die laufende Applikation setzt die Drawcalls an die Hauptinstanz (Mothership) ab.

**Abbildung 1.0**, Architektur von Chromium mit Mothership und SUPs Quelle: http://chromium.sourceforge.net/doc/index.html

## Anwendung

### Multi-Monitor Displays

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



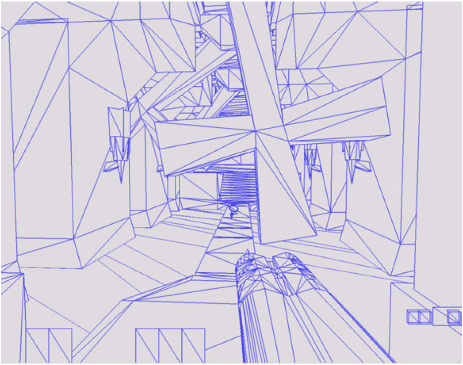
**Abbildung 1.1**, Ansicht der Leinwände in vier Himmelsrichtungen, Quelle: http://chromium.sourceforge.net/presentations/SantaFe-BrianPaul/siframes.html

### 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.

### Manipulation

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



### Stereoskopie

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

### Command Stream Aufteilung

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

**Abbildung 1.2**, Implementation eigener Rendering Styles,

Quelle: (<http://chromium.sourceforge.net/doc/index.html>)

## Argumentation

### Pro

#### Stereoskopie

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

#### Infrastruktur

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

#### Aufteilung der Monitore

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

### Kontra

#### 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.

#### 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*.”

#### 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).

#### 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.

#### 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.

#### Community

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

## 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.

# 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.

## 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.

## Architektur

Equalizer verwendet verschiedene Wrapperklassen, um die Systemressourcen abstrahiert darzustellen.

Die gesamte Liste der Ressourcenkassen finden sie hier:

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

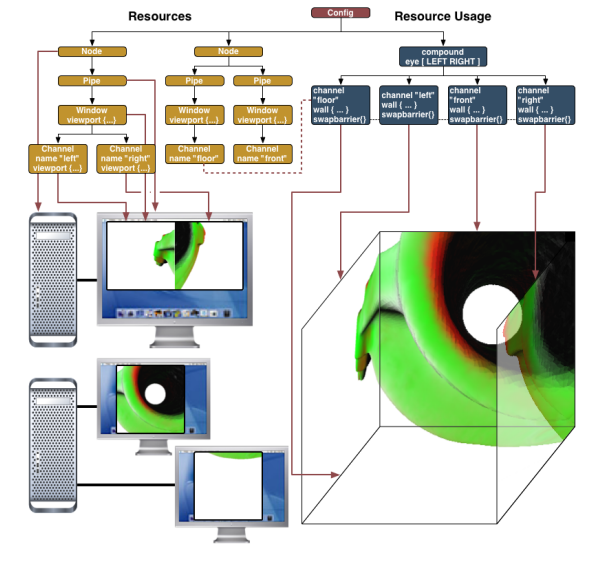


Abbildung 1: Equalizer Beispiel

(<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

## 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>.

* Multi Displays

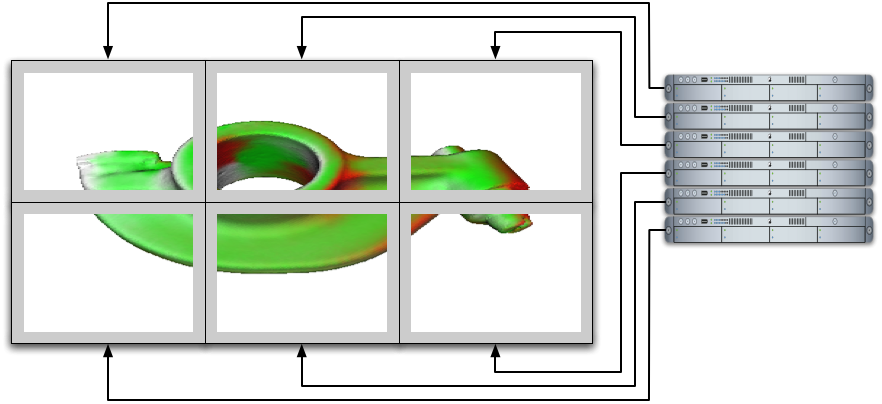


Abbildung 2: Display Wall (<http://www.equalizergraphics.com/useCases.html>)

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

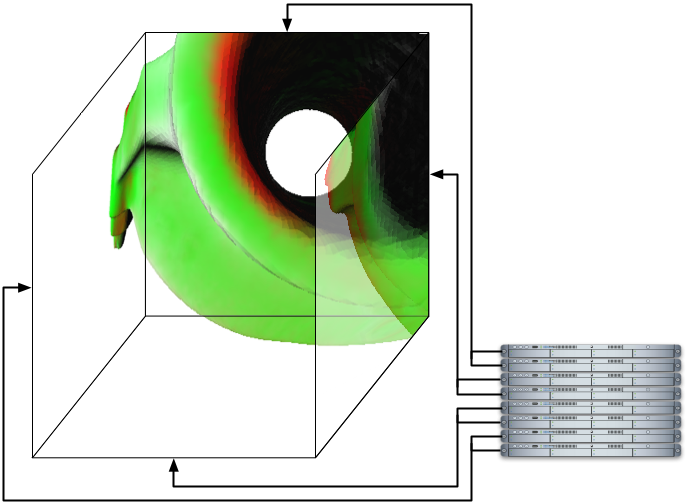


Abbildung 3: Vierseiten VR Installation (<http://www.equalizergraphics.com/useCases.html>)

## Argumentation

### Pro

#### Open Source, Dokumentation

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

#### 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)

### Kontra

#### Unity3D Bezug

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

#### 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 genau so und nur so einzubinden.

Weiter ist unklar, ob über die Objektklassen alles andere von Unity (KI, Physik, etc) auch abbilden lässt.

## 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.

1. 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.

# Eigene Lösung

## 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.

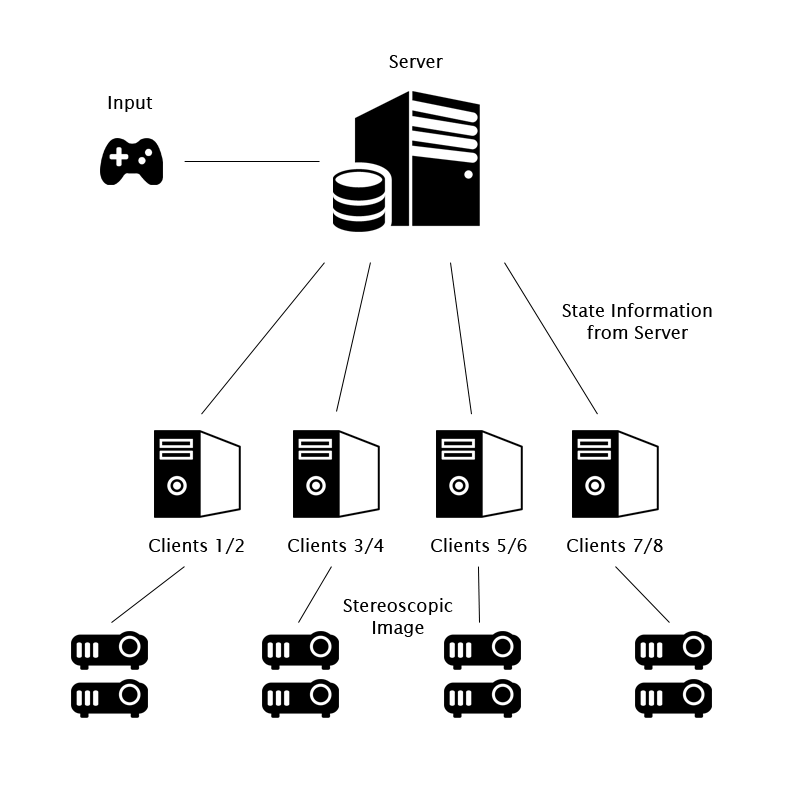
### Unity

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.

### 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.



## 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.

Grafik 1, Client-Server Architektur, Quelle: <http://www.flaticon.com/>

### 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.

### 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.

## Funktionale Anforderungen

### 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.

### Stereoskopie

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

### 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.

## Argumentation

### Pro

#### 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.

#### 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.

#### 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.

### Kontra

#### 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.

#### 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.

## 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.

# middleVR

## 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 Fremdsoftwaren, und hier nur ein vielversprechendes Produkt vorstellen.

## Warum middleVR?

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

*MiddleVR adds the following capabilities to Unity:*

* *Scale one visualization with user-centric perspective,*
* *Support for 3D interaction devices such as 3D trackers (see full list on the right),*
* *S3D – Stereoscopy (active, passive),*
* *Multi-screens / multi-computers synchronization for higher-resolutions and impressive VR systems,*
* *3D interactions: navigation, manipulation,*
* *Immersive menus,*
* *Custom graphical user interfaces (in HTML5),*
* *Display any webpage inside your virtual world.*

*Thanks to the simple and powerful MiddleVR plugin for Unity, create and experience interactive & immersive VR applications in minutes!*

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

## Abdeckung middleVR

middleVR besteht auf zwei Hauptkomponenten:

* Es vereinfacht die Erstellung und Programmierung von VR Anwendungen
* Es ist adaptierbar auf verschiedenste VR Hardware und 3D Anwendungen

### 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.

### http://www.middlevr.com/doc/current/images/MouseTracker.pngUnsere funktionalen Anforderungen

Abbildung 4: middleVR Konfigurator

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

### Unsere nicht funktionalen Anforderungen

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

## http://www.middlevr.com/doc/current/images/workflow.pngKonzept 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 5: Basiskonzept middleVR

### Beschreibung

Die Beschreibung umfasst folgendes :

* Die Devices des VR Systems
* Eine Beschreibung, wie das diese Devices mit der reelen Welt interagieren.

(Beispielsweise Tracker A folgt dem Kopf des Benutzers, Tracker B folgt dessen linker Hand)

* Die Positionen der Screens
* Welche Kameras was wo renden müssen

#### Beispiel

Diese Konfiguration hier 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 6: Beispielskonfiguration CAVE

## Argumentation

### Pro

#### Kein From Scratch

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

#### 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.

#### Support

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

### Kontra

#### 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.

#### Eigenanteil

Für den Stolz der CPVR-Studenten ist es auch wichtig, einen signifikanten Eigenanteil beizusteuern.

Im günstigsten Fall funktioniert die Software nach kurzer Konfiguration.

## Zusammenfassung

middleVR bietet sich sicherlich 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.

# Gegenüberstellung

Aufgrund der Analysen wurden die Pros und Kontras zusammengefasst, damit sie vergleichbar werden. Folgende Kategorien wurden erstellt, sowie kurze Erklärungen und der vergebene Wert dazu:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Support |  | Keyfeatures |  | Unity |  | Architektur |  |
| Pros | Chronium |  |  | Steroskopie gegeben;  Aufteilung der Monitore; | 0.3 |  | 0.1 |  | 0.3 |
| Equalizer | Open Source | 0.5 | Graphikklassen vorhanden (KI & Physik nicht) | 0.3 |  | 0.2 | Graphikklassen vorhanden | 0.3 |
| Eigene Lösung |  | 0 | Sollte prinzipiell alles machbar sein | 0.8 | Nahe an Unity | 0.8 | Nicht abschätzbar | 0.5 |
| middleVR | Professionelle, aktuelle Software | 0.8 | Problemlos | 1 | Unity integriert | 1 | Auf Unity aufbauend | 1 |
| Kontras | Chronium | Seit 2006 praktisch tot | 1 |  |  | OpenGL, Unity Support unklar.  (OpenGL nur bis 1.5) | 0.9 | Netzwerkauslastung könnte Engpass sein | 0.7 |
| Equalizer | Open Source | 0.5 | Wahrscheinlich 2/3 nicht vorhanden | 0.7 | C++ Implementation | 0.8 | Starre Struktur gegeben durch die Wrapperklassen | 0.7 |
| Eigene Lösung | Eigen- implementation | 1 |  | 0.2 |  | 0.2 | Nicht abschätzbar | 0.5 |
| middleVR |  | 0.2 |  | 0 |  | 0 |  | 0 |

Dies ergibt dann folgende Diagramme.

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

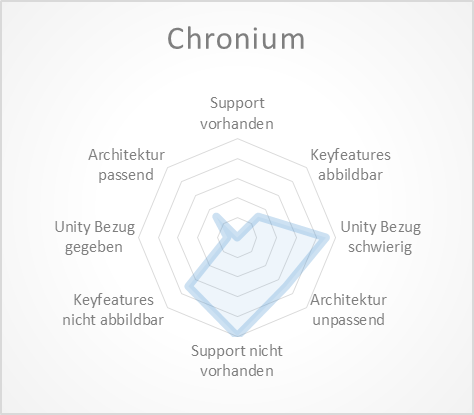


Abbildung 7: Pro-Kontra Grid Chronium



Abbildung 10: Pro-Kontra Grid Eigene Lösung

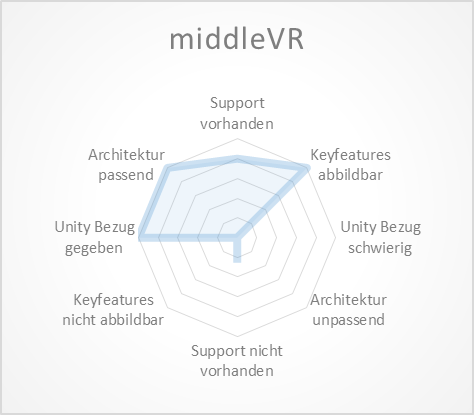


Abbildung 9: Pro-Kontra Grid middleVR

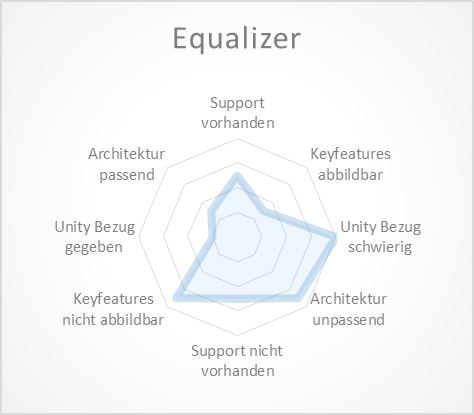


Abbildung 8: Pro-Kontra Grid Equalizer

# Entscheid

# Abbildungsverzeichnis

Abbildung 1: Equalizer Beispiel 8

Abbildung 2: Display Wall (http://www.equalizergraphics.com/useCases.html) 9

Abbildung 3: Vierseiten VR Installation (http://www.equalizergraphics.com/useCases.html) 9

Abbildung 4: middleVR Konfigurator 15

Abbildung 5: Basiskonzept middleVR 15

Abbildung 6: Beispielskonfiguration CAVE 16

# Tabellenverzeichnis

Tabelle 1: Wichtige Equalizer Klassen 9

Tabelle 2: Gegenüberstellung 17

# 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

# Versionskontrolle

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Beschreibung** | **Autor** |
| 1.00 | 24.03.2015 | Dokument erstellt | Daniel Inversini |
| 1.01 | 25.03.2015 | Dokument überarbeitet | Daniel Inversini |