|  |
| --- |
| Equalizer Framework  Voranalyse  **Daniel Inversini**  **V1.00 01.03.2015** |
| **Berner Fachhochschule**  Technik und Informatik |

Inhaltsverzeichnis

[1 Equalizer – The parallel rendering framework 3](#_Toc412996685)

[1.1 Warum Voranalyse Equalizer? 3](#_Toc412996686)

[1.2 Architektur 3](#_Toc412996687)

[1.3 Anwendung 4](#_Toc412996688)

[2 Argumentation 5](#_Toc412996689)

[2.1 Pro 5](#_Toc412996690)

[2.1.1 Open Source, Dokumentation 5](#_Toc412996691)

[2.1.2 Know-How BFH 5](#_Toc412996692)

[2.2 Kontra 5](#_Toc412996693)

[2.2.1 Unity3D Bezug 5](#_Toc412996694)

[2.2.2 Architektur 5](#_Toc412996695)

[3 Zusammenfassung 6](#_Toc412996696)

[4 Literaturverzeichnis 6](#_Toc412996697)

[5 Anhang 6](#_Toc412996698)

[6 Versionskontrolle 6](#_Toc412996699)

# 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 Projekt 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>

Als Beispiel sieht dann dies so aus:

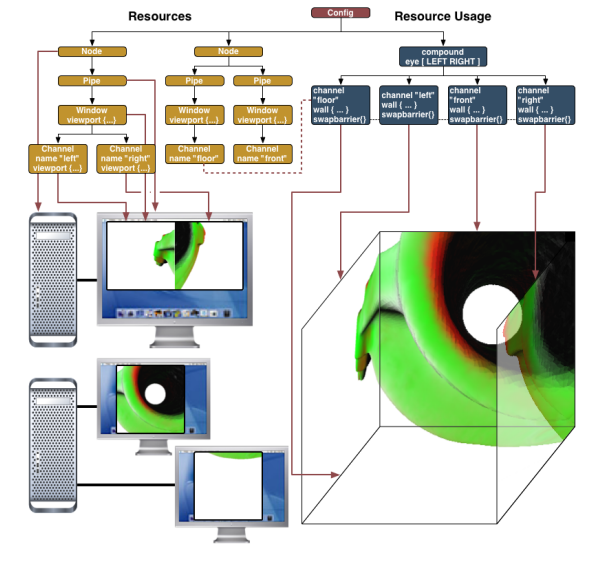


Abbildung 1: Equalizer Beispiel (<http://www.equalizergraphics.com/documents/Developer/eqPly.pdf>)

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

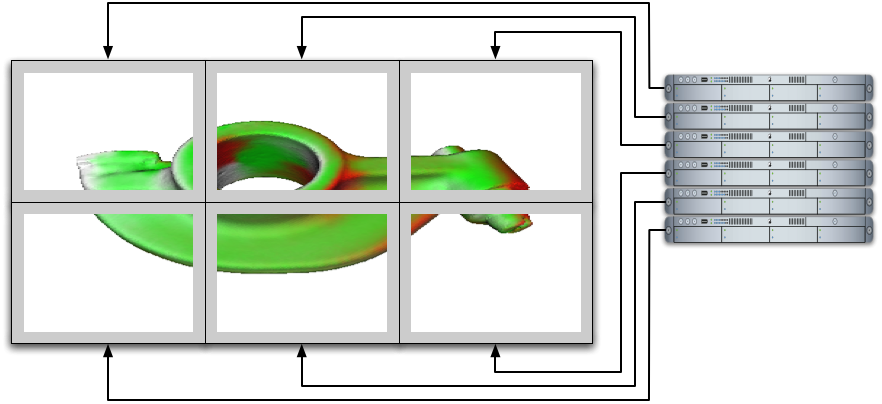


Abbildung 2: Display Wall (<http://www.equalizergraphics.com/useCases.html>)

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

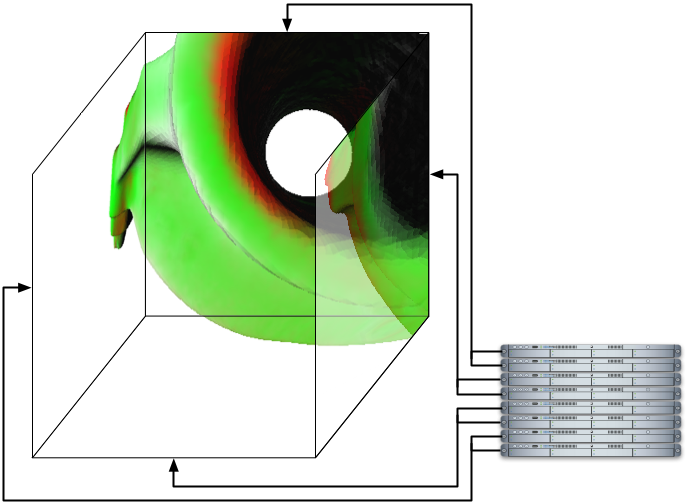


Abbildung 3: Vierseiten VR Installation (<http://www.equalizergraphics.com/useCases.html>)

# 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.

# Literaturverzeichnis

**Anwendungsfälle Equalizer**

[*http://www.equalizergraphics.com/useCases.html*](http://www.equalizergraphics.com/useCases.html)