Unity in CAVE

Pflichtenheft

Julien Villiger, Daniel Inversini
V 2.1, 16.03.2015

Berner Fachhochschule
Technik und Informatik
Informatik

Inhaltsverzeichnis

1 Allgemeines

1.1 Zweck dieses Dokumentes

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

1.2 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.

1.3 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.

1.4 Umfang der Projekt 2 Arbeit

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.

1.5 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.

1.6 Abgrenzungen

1.6.1 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.

1.6.2 Weitere Abgrenzungen

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

1.7 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.

2 Voranalysen

2.1 Equalizer

Equalizer ist ein Open Source Framework für skalierbares, paralleles Rendering basierend auf OpenGL, welches ein API zur Verfügung stellt um graphische Applikationen zu entwickeln. Es verwendet verschiedenste Wrapperklassen in C++, um die Systemressourcen abstrahiert darzustellen. Equalizer wird hier analysiert, da es bereits im CAVE der BFH in Verwendung ist.

2.2 Chromium Bibliothek

Mittels der Chromium Bibliothek wird ein OpenGL Command nicht direkt in ein Rasterbild umgewandelt, sondern kann manipuliert und an andere OpenGL Implementationen weitergeschickt werden. Somit sollte es möglich sein, auf jedem Client den Command entsprechend für die spezifische Leinwand anzupassen. Die Machbarkeit dieser Methode wird geprüft.

2.3 MiddleVR

Der französische Hersteller Creative Valley hat ein Plugin entwickelt, um Unity Anwendungen in einem CAVE stereoskopisch darstellen zu können. Die Machbarkeit dieser Methode wird geprüft.

2.4 Unity Standalone

Möglicherweise bietet kein Hilfsmittel genügend Möglichkeiten und es muss eine eigene Lösung erarbeitet werden.

3 Prototyping

Basierend auf den Voranalysen werden ein oder mehrere Prototypen erstellt, welche zu einem späteren Zeitpunkt auch im CAVE lauffähig installiert werden.

Die Ausarbeitung der Prototypen kann unterschiedlich ausfallen. Einerseits ist der Aufwand der Installation im CAVE schlecht abschätzbar und andererseits kann die Art des Prototyps je nach gewähltem Ansatz (eigene Lösung oder die Verwendung eines Frameworks / API) stark variieren.

3.1 Basisfunktionalität

Gewisse Basisfunktionalitäten müssen aber unabhängig des Typs gegeben sein und werden bei jedem Prototypen integriert.

3.1.1 Stereoskopie

Die Verwandlung einer Standardprojektion in eine stereoskopische Darstellung wird umgesetzt.

3.1.2 Multi-screen Rendering

Sämtliche Leinwände oder eine Selektion unterteilen die Projektion. Jeder Rendering-Client liefert also nur für einen bestimmten Ausschnitt Bildinformationen. Im Falle des CAVEs der BFH wird das Bild viermal unterteilt, damit jede Leinwand (links, rechts, geradeaus und unten) das spezifische Bild anzeigen kann und für den User eine räumliche Illusion entsteht.

3.1.3 Input Devices

Typische Eingabegeräte wie Maus und Tastatur gehören zur Basisfunktionalität und werden durch Unity selbst abgedeckt. Zusätzliche Inputinformationen, sei dies mittels Tracking oder speziellen Geräten wie eine „MagicWand“ oder ein „Glove“, sprengen den Rahmen der Basisfunktionalität und werden vorerst nicht integriert.

3.2 Anwendung: Basic

Um die besagten Basisfunktionalitäten darstellen zu können, wird eine simple 3D Welt erstellt, die aus wenigen Primitiven wie Würfeln, Flächen und Kegeln besteht. Auf ein komplexes Lighting und Texturing wird bewusst verzichtet. So kann weitgehend ohne Performancebeeinflussung die Basisfunktionalität sichergestellt und getestet werden.

Dieser Prototyp liefert primär folgende Erkenntnis:

3.2.1 Basisfunktionalität

Ob die Hauptanforderungen an das Produkt den Praxistest bestehen, wird an dieser Stelle ersichtlich.

3.2.2 Ergonomie

Der Vorgang des Einpflegens einer Unity Anwendung mittels dem erarbeiteten Produkt soll mit wenigen Clicks möglich sein. Wie aufwändig, intuitiv und stabil der Prozess ist, wird an dieser Stelle ersichtlich und kann nach der Auswertung verbessert werden.

3.3 Anwendung: Game

Sollten sämtliche Basisfunktionalitäten sichergestellt worden sein, wird in einem nächsten Schritt ein ausprogrammiertes Unity Game in den CAVE implementiert. Voraussetzung dafür ist der vorhandene Source-Code.

Dieser Prototyp liefert Erkenntnis über folgende Punkte:

3.3.1 Kompatibilität

Wie im Endeffekt angestrebt, müssen alle Unity Games Version 4.6 kompatibel sein. Ein Prototyp dieser Art liefert Erkenntnis darüber, ob unsere Lösung auch Spiele mit Custom-Scripts problemfrei integrieren und die Basisfunktionalitäten gewährleisten kann. Ein wichtiger Punkt dabei ist, ob die Synchronisierung bei Zufallsbasierten Scripts zwischen den Rendering-Clients und der Hauptinstanz funktioniert.

3.3.2 Performance

Mit der gegebenen Infrastruktur sollten auch rechenintensive, grafisch ansprechende Games und Anwendungen mit hoher Netzwerkkommunikation fliessend laufen. Die mentale Immersion des Users darf nicht durch technische Defizite beeinträchtigt werden. Entsprechende Games werden auf diese Punkte getestet.

4 Funktionale Anforderungen

4.1 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.

4.2 Kompatibilität

Sämtliche, quelloffene Unity Anwendungen Version 4.6 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.

4.2.1 Unity Plugin

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.

4.2.2 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.

4.2.3 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.

4.3 Plattformunabhängigkeit

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

5 Nicht Funktionale Anforderungen

5.1 Presence

Die mentale Immersion wird durch den CAVE im Vergleich zur normalen Ausführung der Unity Anwendungen deutlich gesteigert. Der Benutzer fühlt sich geistig in die virtuelle Welt versetzt und soll möglichst gering durch Input-Devices beeinträchtig werden.

5.2 Wiederverwendbarkeit

Damit zukünftige Entwickler effizient eigene Anwendungen in den CAVE einpflegen können, wird viel Wert auf die Wiederverwendbarkeit gelegt.

Anwender aus verschiedenen Bereichen wie Architektur, Autoindustrie, Game Development usw. können ihre Simulationen in den CAVE einpflegen und ausführen.

5.3 Ergonomie

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

6 Testing

6.1 Systemtests

Während der Prototypingphase werden laufend Tests auf unabhängigen Rechnern sowie im CAVE durchgeführt um sicherzustellen, dass während der Entwicklung mögliche Probleme sofort erkannt werden und Massnahmen ergriffen werden können.

6.2 Usabilitytests

Abhängig vom Fortschritt der Prototypen werden Tests mit potenziellen Anwendern durchgeführt um die Usability der Lösung abschätzen und optimieren zu können. Sowohl die Inbetriebnahme des CAVEs wie auch die Adaption der eigenen Unity Anwendungen werden berücksichtigt.

7 Administratives

7.1 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.

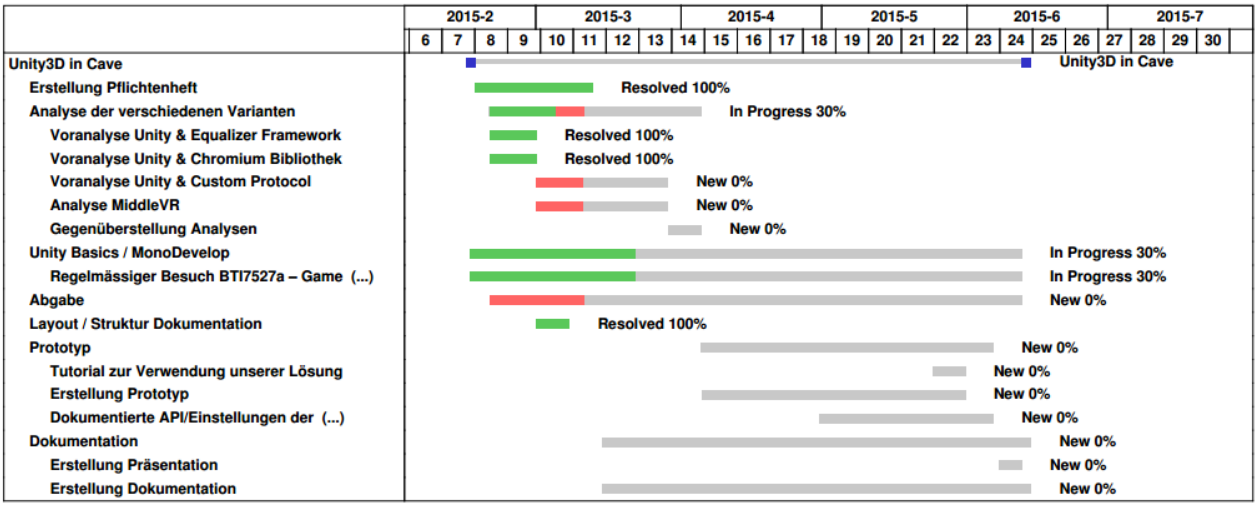
7.1.1 Projektteam

Daniel Inversini daniel.inversini@students.bfh.ch
Julien Villiger julien.villiger@students.bfh.ch

7.1.2 Betreuer

Prof. Urs Künzler urs.kuenzler@bfh.ch

7.2 Projektplan



https://pm.ti.bfh.ch/projects/unity3d-in-cave/issues/gantt

7.3 Projektsitzungen

In einer ersten Phase wird alle zwei Wochen ein Projektmeeting des Teams mit Betreuer durchgeführt. Startend ab dem 17. Februar 2015.

(17.02.2015; 03.03.2015; 17.03.2015; 31.03.2015; 14.04.2015; 28.04.2015; 12.05.2015; 26.05.2015; 09.06.2015)

7.4 Meilensteine

Folgende Tasks aus dem Projekt wurden als Meilensteine definiert:

7.4.1 Voranalyse

Stichtag 29.03.2015
Analyse und Gegenüberstellung der verschiedenen Varianten, Entscheidung für eine dieser Varianten.

7.4.2 Prototyp

Stichtag 31.05.2015
Lauffähiger Prototyp vorhanden im CAVE der BFH.

7.4.3 Dokumentierte API / Einstellungen der Lösung / Prototyp

Stichtag 06.06.2015
Der Prototyp sollte modular und flexibel sein, dass eine nachfolgende Arbeit (Bachelorarbeit oder andere Projektarbeiten) einfach eingebunden werden kann. Als weiteres Hilfsmittel dazu dient die Dokumentation.

8 Versionskontrolle

Version	Datum	Beschreibung	Autor
0.1	04.03.2015	Dokument erstellt / Struktur definiert	Daniel Inversini
0.2	04.03.2015	Struktur überarbeitet	Julien Villiger
1.0	10.03.2015	Updates	Julien Villiger
1.1	12.03.2015	Updates	Daniel Inversini
1.2	12.03.2015	Korrekturlauf / Dokument fertiggestellt	Julien Villiger
2.0	15.03.2015	Überarbeitung gemäss Feedback	Julien Villiger
2.1	16.03.2015	Überarbeitung gemäss Feedback	Daniel Inversini