Exposé

CINEMA 4D als grafischer Editor mit Scriptunterstützung für Fusee

Alexander Scheurer
Baumannstraße 37
78120 Furtwangen

alexander.scheurer@hs-furtwangen.de
Matr.-Nr. 248073

Hochschule Furtwangen University
Fakultät Digitale Medien
Studiengang Medieninformatik Master

Forschungsseminar/Wissenschaftliches Arbeiten
Prof. Dr. Oliver Ruf

Wintersemester 2014/15

Version 1
Stand: 7. Februar 2015

Inhaltsverzeichnis

1 Einleitung

1.1 Übersicht

Aktuelle 3D-Grafik-Engines oder auch Spiele-Engines, zum Beispiel Unity, Unreal Engine oder CryEngine bieten Spiele-Entwicklern eine grafische Oberfläche (Editor). Im Vergleich zu 3D-Grafiksoftware wie CINEMA 4D oder 3ds Max, bieten sie allerdings nur sehr beschränkte Möglichkeiten 3D-Objekte zu modellieren/erstellen [GL14, S.54]. Das heißt auch, dass je nach dem welche Kombination an 3D-Grafik-Engine und 3D-Grafiksoftware verwendet wird eine Schnittstelle geschaffen werden muss um die Daten (zum Beispiel Geometrie, Texturen, Materialien und Animationen) auszutauschen.

Diese Editoren bieten meist auch eine direkte Schnittstelle zur Programmierung indem Programmcode mit 3D-Objekten verknüpft werden kann, beziehungsweise der Programmcode organisiert werden kann. Seltener existiert auch eine Oberfläche für grafische Programmierung wie zum Beispiel Blueprint von Unreal.

Einen eigenen Editor für eine 3D-Grafik-Engine bereitzustellen ist ein zusätzlicher Entwicklungsaufwand, der zur Benutzung der Software nicht zwingend notwendig ist. Es gibt 3D-Grafik-Engines die keinen Editor bereitstellen (zum Beispiel Fusee), was bei der Benutzung des Engines bedeutet, dass ein Programmierer im Programmcode die 3D-Objekte einer Szene setzen muss.

1.2 Aufgabe/Fragestellung

Die Idee dieser Arbeit ist, eine 3D-Grafiksoftware als Editor für eine 3D-Grafik-Engine prototypisch anzubinden und in dem Umfang zu erweitern, dass auch eine Schnittstelle zum Programmcode existiert. Konkret soll hier CINEMA 4D an Fusee angebunden werden. Es soll einerseits möglich sein Programmcode an 3D-Objekte zu binden, als auch die grafische Programmieroberfläche von CINEMA 4D, XPresso, zu verwenden, um daraus lauffähigen Programmcode für Fusee zu erzeugen. Dies soll über ein Plugin für CINEMA 4D geschehen, dass die Kommunikation beider Programme bereitstellt und die Übersetzung

zwischen verschiedenen Datentypen übernimmt.

Hier soll geprüft werden ob die Datenstruktur zur Kommunikation an den OpenGEX Spezifikationen orientieren lässt um eine größtmögliche Kompatibilität auch mit anderer 3D-Grafiksoftware und anderen 3D-Grafik-Engines zu ermöglichen.

Ziel ist es, einen Editor zu schaffen, der alle Fähigkeiten einer 3D-Grafiksoftware besitzt und gleichzeitig direkt an eine 3D-Grafik-Engine angebunden ist. Es soll auch gezeigt werden, ob es praktikabel ist, einen nicht speziell für die 3D-Grafik-Engine entwickelten Editor, in Form einer 3D-Grafiksoftware, zu nutzen.

Als Vorbild für die Umsetzung soll die Funktionalität des Unity Editors dienen.

2 Theoretischer Hintergrund

2.1 Fusee

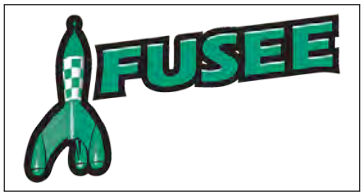

Abbildung 1: Fusee Logo

Die Furtwangen University Simulation and Entertainment Engine (Fusee[1]) ist eine 3D-Grafik-Engine, die an der Fakultät Digitale Medien der Hochschule Furtwangen University seit dem Wintersemester 13/14 entwickelt wird. Fusee wird derzeit überwiegend von Studenten im Projektstudium, verschiedenen Wahlpflichtmodulen, Bachelor- sowie Master-Thesen weiter entwickelt. Fusee ist speziell für das Lernen und Lehren im Bereich der 3D-Grafik, Spiele und Simulationen gedacht und ermöglicht in dieser Hinsicht Zugriff auf tiefliegende Funktionen (vergleiche [MG14]).

Grundsätzlich versucht Fusee so viele Plattformen wie möglich zu bedienen. Hierunter zum Beispiel Windows, Linux, Android, iOS und auch HTML5/WebGL fähige Browser. Fusee verfolgt einen hoch modularen Ansatz, jegliche extern anprogrammierte Software und auch teilweise interne bereitgestellte Methoden sollen vom Benutzer der Engine ausgetauscht werden können. Beispiele für Module sind die Grafikanbindung von OpenGL über OpenTK oder die Physikanbindung von Bullet über BulletSharp (siehe [Sch14]).

Seinen modularen Aufbau erreicht Fusee dadurch, dass der Programmcode in drei Bereiche aufgeteilt wird. Der 'Core' beinhaltet alle grundlegenden internen Funktion, der 'Platform Adapter' beinhaltet die Anprogrammierung externen Codes und 'Common' stellt die Funktionalität der Implementierung über Interfaces bereit (siehe Abbildung 2).

[1] http://www.fuse3d.org

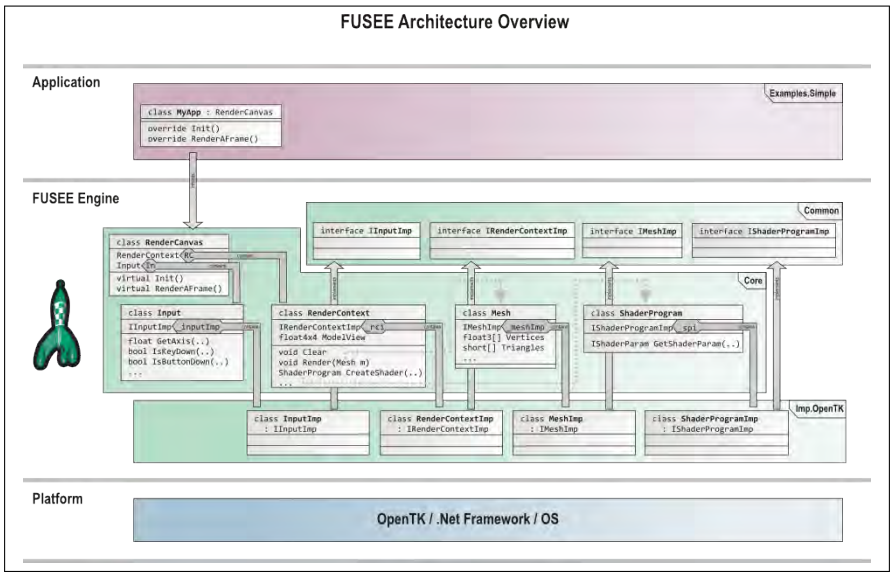

Abbildung 2: Fusee Aufbau

2.2 CINEMA 4D

CINEMA 4D[2] ist eine kostenpflichtige 3D-Grafiksoftware von der MAXON Computer GmbH. Für den Bildungsbereich stellt MAXON kostenlose Lizenzen zur Verfügung. CINEMA 4D wird seit 1990 entwickelt und ist mit ihrem aktuellen (2014) Update in Version R16 für Windows und Max OS X verfügbar. Mit CINEMA 4D unterstützt unter anderem Modellierung, Texturierung, Animation und eine Möglichkeit der grafischen Programmierung (XPresso).

XPresso ist Node basiert, es ermöglicht also über Verbindungslinien Abhängigkeiten zwischen Objekten zu definieren. Zum Beispiel die Anzahl der Segmente des einen Objekts wird zur Position des anderen Objektes (siehe Abbildung 3). XPresso stellt alle relevanten Logik- und Rechenoperationen zur Verfügung, um auch komplexere Probleme zu lösen.

CINEMA 4D bietet eine umfangreiche Programmierschnittstelle (API) die es ermöglicht die Funktionalität von CINEMA 4D von außen, mit selbst erstellten Plugins, zu erweitern.

[2]http://www.maxon.de/

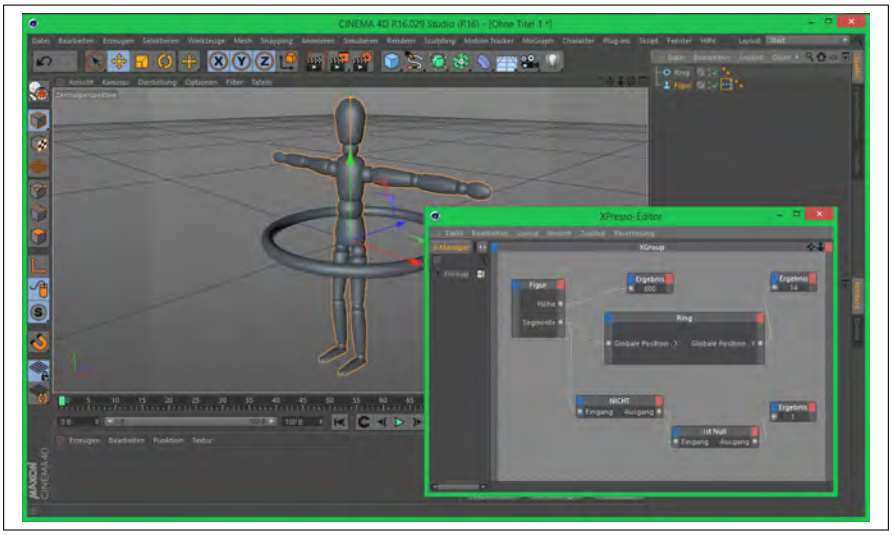

Abbildung 3: CINEMA 4D mit XPresso-Editor

2.3 OpenGEX

Open Game Engine Exchange (OpenGEX[3]) ist ein Dateiformat, das zum Austauschen von Daten zwischen 3D-Grafiksoftware und 3D-Grafik-Engine gedacht ist und wird seit 2013 von Eric Lengyel entwickelt. Es basiert in seiner Struktur auf COLLADA[4], welches schon seit Längerem ein etabliertes Format im 3D-Grafik Bereich ist. OpenGEX erweitert COLLADA um interaktive Strukturen im es geeigneter für Spiele und Simulationen zu machen. [Len14]

2.4 Unity

Unity[5] ist eine 3D-Grafik-Engine die von Unity Technologies entwickelt wird und sowohl in einer Kostenlosen als auch Kostenpflichtigen-Version verfügbar ist. Unity besteht aus einer Laufzeitumgebung und einem grafischen Editor [GL14, S.29]. Im Editor können Objekte im 3D-Raum platziert werden oder er kann als Vorschau für die aktuelle Szene zur Laufzeit verwendet werden. Er ist

[3]http://opengex.org/
[4]https://www.khronos.org/collada/
[5]http://www.unity3d.com/

nicht geeignet Objekte zu modellieren, dies geschieht in der Regel mit anderen Werkzeugen, zum Beispiel mit CINEMA 4D.

Unity ist komponentenbasiert, das heißt, die Möglichkeiten ein von Objekten zu beeinflussen werden dadurch verändert, dass eine Komponente dem Objekt hinzugefügt wird. Auf Abbildung 4 im rechten Teilfenster, dem *Inspector*, ist zu sehen, dass dem Objekt *Ship* eine *Transform*-Komponente und eine *Script*-Komponente vom Typ *ShipScript* zugewiesen ist.

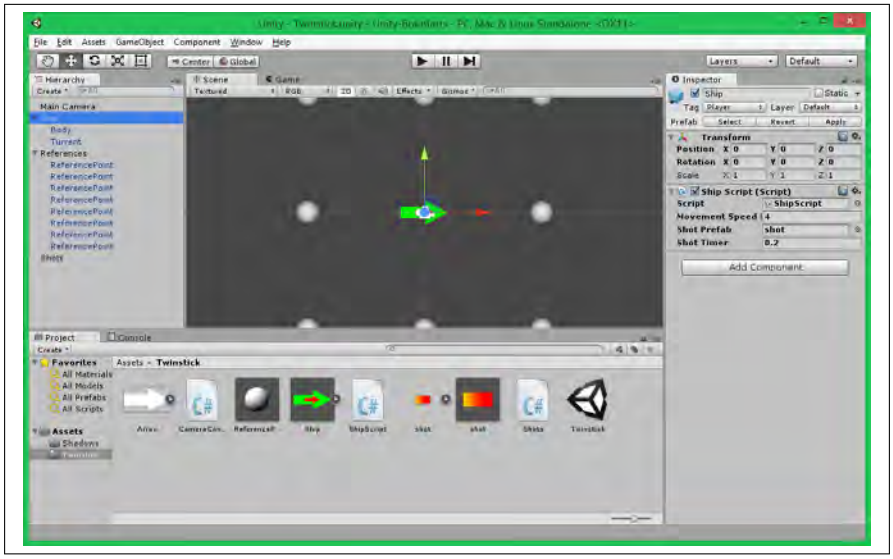

Abbildung 4: Unity Editor

3 Methodik und Vorgehensweise

3.1 Allgemein

In der Softwaretechnik gibt es etablierte Vorgehensweisen zur Entwicklung von Software. Da hier keine alleinstehende Software entwickelt werden soll, sondern in Abhängigkeit von anderer Software steht, müssen beide Seiten analysiert und passend verbunden werden. Die CINEMA 4D API ist objektorientiert in C++ entwickelt. Fusee hingegen ist zwar auch objektorientiert aufgebaut jedoch in C# entwickelt. Dadurch liegt nahe, auch das Plugin objektorientiert zu entwickeln. Es muss jedoch die Brücke geschlagen werden zwischen C++ und C#. Hier muss evaluiert werden in welcher der beiden Sprachen das Plugin entwickelt werden soll und wie es an die jeweils andere Sprache angebunden (gewrappt) werden kann.

Weiterführende Literatur: [FKC06], [BHS07], [GR04], [Bal01]

3.2 Werkzeuge

3.2.1 Visual Studio

Visual Studio[6] ist eine Entwicklungsumgebung von Microsoft die mehrere Programmiersprachen, darunter C# und C++, umfasst. Sowohl Fusee als auch CINEMA 4D werden mit Visual Studio entwickelt.

3.2.2 ReSharper

ReSharper[7] ist eine Erweiterung für Visual Studio von JetBrains. Es erlaubt die Analyse der Programmcode-Qualität und erzwingen eines definierten Programmierstil. Dies macht es einfacher Fehler auch schon während der Entwicklung zu erkennen und durch einen einheitlichen Programmierstil den Programmcode lesbarer und nachvollziehbarer zu machen.

[6] http://www.visualstudio.com/
[7] https://www.jetbrains.com/resharper/

3.2.3 Git

Bei der Softwareentwicklung ist von umfangreicheren Projekten eine Versionskontrolle unabdingbar. Sie bietet die Möglichkeit den Entwicklungsvorgang nachzuvollziehen und Fehler bei der Entwicklung, auch weit nach deren Entstehen, zu erkennen. Da das zu erweiternde Fusee Projekt ebenfalls auf Git[8] basiert soll auch in dieser Arbeit geht verwendet werden. [LM12]

3.2.4 SWIG

Mit SWIG[9] (Simplified Wrapper and Interface Generator) existiert bereits eine automatisierte Möglichkeit eine Anbindung zwischen C++ und C# zu ermöglichen. Ohne SWIG würde allein das manuelle Verbinden den Umfang der Arbeit sprengen, es muss lediglich auf beiden Seiten die Voraussetzung für die Anbindung geschaffen werden.

[8] http://git-scm.com/
[9] http://www.swig.org/

4 Wissenschaftliche Einordung

4.1 Wissenschaftsgebiet

Im Hauptteil der Arbeit soll eine Software erweitert werden um die Anbindung an eine zweite Software, dies ist eine überwiegend praktische Ausrichtung und gehört damit ins Feld der praktischen Informatik, genauer der Softwaretechnik.

4.2 Stand der Forschung

Im Bereich der Softwaretechnik allgemein ist es nichts Neues, dass Software auch über die Grenzen von Programmiersprachen hinweg miteinander interagiert. Dies kann zum Beispiel Textverarbeitungssoftware sein, die mit einer Spracherkennungssoftware erweitert wird oder Virtualisierungssoftware die es ermöglicht ganze Betriebssysteme miteinander zu verbinden.

In diesem sehr speziellen Bereich der 3D-Computergrafik waren jedoch keine Beispiele zu finden.

5 Zeitplan

- 1 Monat - Literaturrecherche und Auseinandersetzung mit dem Thema
- 1 Monat - Prototyping der Implementierung
- 2 Monate - Fertigstellung der Implementierung
- 1 Monat - Performancetests, Optimierungen und Auswertung
- 1 Monat - Fertigstellung der Arbeit

Literaturverzeichnis

[Bal01] H. Balzert. *Lehrbuch der Software-Technik*. 2. Aufl. Heidelberg und Berlin: Spektrum Akad. Verl, 2001. ISBN: 3-8274-0480-0 (siehe S. 9).

[BHS07] F. Buschmann, K. Henney und D. C. Schmidt. *Pattern-oriented software architecture, Volume 5, On patterns and pattern languages*. Wiley series in software design patterns. Chichester, England: J. Wiley, c2007. ISBN: 0-471-48648-5. URL: `http://proquest.tech.safaribooksonline.de/9780471486480` (siehe S. 9).

[FKC06] P. Forbrig, I. O. Kerner und J. Cleve, Hrsg. *Programmierung: Paradigmen und Konzepte*. Lehr- und Übungsbuch Informatik. München: Fachbuchverl. Leipzig, 2006. ISBN: 9783446403017. URL: `http://deposit.ddb.de/cgi-bin/dokserv?id=2618111&prov=M&dok_var=1&dok_ext=htm` (siehe S. 9).

[GL14] J. Gregory und R. Lemarchand. *Game engine architecture*. 2. ed. Boca Raton, Fla.: CRC Press, 2014. ISBN: 9781466560017 (siehe S. 3, 7).

[GR04] E. Gamma und D. Riehle. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Programmer's choice. München: Addison-Wesley, 2004. ISBN: 3-8273-2199-9 (siehe S. 9).

[Len14] E. Lengyel. *Open Game Engine Exchange Specification: Version 1.1.2*. Hrsg. von Terathon Software LLC. Roseville, California, 2014. URL: `http://opengex.org/OpenGEX.1.1.2.pdf` (besucht am 06. 02. 2015) (siehe S. 7).

[LM12] J. Loeliger und M. McCullough. *Version control with Git: [powerful tools and techniques for collaborative software development]*. 2. ed., covers GitHub. Beijing: O'Reilly, 2012. ISBN: 9781449316389 (siehe S. 10).

[MG14] C. Müller und F. Gärtner. »Student Project – Portable Real-Time 3D Engine«. In: *EG 2014 - Education Papers*. Hrsg. von Jean-Jacques Bourdin, Joaquim Jorge und Eike Anderson. Strasbourg, France: Eurographics Association, 2014, S. 37–38. URL: `http://fusee3d.org/wp-content/uploads/2014/04/FUSEE_Educational.pdf` (besucht am 27. 06. 2014) (siehe S. 5).

[Sch14] L. Schey. »Einbindung einer Physik-Engine in FUSEE«. Bachelor-Thesis, Hochschule Furtwangen University, Fakultät Digitale Medien. Furtwangen im Schwarzwald, 2014 (siehe S. 5).