

Bearbeitungsbeginn: 01.09.2014

Vorgelegt am: TBA

Thesis

zur Erlangung des Grades

Master of Science

im Studiengang Medieninformatik

an der Fakultät Digitale Medien

Dominik Steffen

Matrikelnummer: 245857

Technical game-authoring process and tool
development

Erstbetreuer: Prof. Christoph Müller

Zweitbetreuer: Prof. Dr. Wolfgang Taube

Abstract

Arbeitsprozesse in heutigen Game Engines verlangen von Entwicklern meist das Erlernen neuer Toolsets und das während eines meist sehr eingeschränkten Projekt Zeitraums. Es wäre für Entwickler einfacher sich mit den bereits bekannten Tools zu beschäftigen und mit diesen großartige Ergebnisse zu erreichen. Designer müssen sich oft in unbekannte Editoren und SDKs einarbeiten während Entwickler sich in Grafische Editoren einarbeiten sollen um ihren Code an der richtigen Stelle des Projekts einzubinden. Diese Arbeit baut eine Brücke zwischen beiden Welten. Durch die Konzeption und Umsetzung eines Software Tools und Entwicklungsprozesses wird eine Trennung der Abhängigkeiten in einem Projekt erreicht. Mit Hilfe eines Plugins ist es möglich, dass Designer oder Entwickler jederzeit mit ihren eigenen Tools in die Entwicklung eines Projektes einsteigen. Es wird ermöglicht mit Cinema 4D und einer IDE wie VS2013 an einem Projekt mit der FUSEE Engine zu arbeiten ohne die bereits bekannte Welt zu verlassen. Ein FUSEE Projektstruktur "managed" durch die Nutzung des entstandenen Cinema 4D Plugins und den generierten Visual Studio Solution Dateien selbst. Das zuerst konzeptionell entworfene Tool wurde während dieser Arbeit umgesetzt und bietet ausreichende Basisfunktionalität um ein Projekt als Entwickler als auch als Artist zu erstellen und zu bearbeiten. Hierzu wurden verschiedene Konzepte betrachtet und andere GameEngines auf Workflow und Anwendbarkeit untersucht. Es wurden einige Kernkonzepte erkannt und für eine Implementierung in FUSEE Uniplug analysiert und weiter entwickelt.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Masterthesis selbstständig und ohne unzulässige fremde Hilfe angefertigt habe. Alle verwendeten Quellen und Hilfsmittel die sowohl zum schreiben dieser Arbeit als auch zum Entwickeln des dazugehörigen Sourcecodes benutzt wurden, habe ich angegeben.

Dominik Steffen, Küssaberg den 13. April 2015

"Hier steht ein wichtiges Zitat zur Entstehung dieser Arbeit."

- TBD.

Dominik Steffen

Matr.-Nr.: 245857

Hochschule Furtwangen

E-Mail:

dominik.steffen@hs-furtwangen.de

dominik.steffen@gmail.com

Inhaltsverzeichnis

1	Anforderungen, Ziele und eine Fragestellung	1
1.1	Motivation	1
1.2	Ziele der Implementierung	2
1.3	Verwendete Software	2
2	Grundlegendes	4
2.1	Entwicklungsprozesse in Interaktiver 3D Software und Games	4
2.1.1	Projektmanagement Modelle	4
2.1.2	Internes Tool Developing anstatt Tool licencing	7
2.2	Mitglieder eines Entwicklerteams	8
2.2.1	Artists	8
2.2.2	Designer	9
2.2.3	Engineer	10
2.2.4	Weitere für diese Arbeit nicht relevante Teammitglieder	11
2.3	Stakeholderanalyse intern	12
2.4	Ein Arbeitsprozess wird entwickelt	12
2.4.1	Game Authoring / Game Development	12
2.4.2	Tool Development	13
2.5	Die Struktur von Game Assets	15
2.5.1	Warum eine Trennung von Code und Content?	15
3	Entwicklung eines Konzeptes	16
3.1	Use Cases der verschiedenen Entwickler	18
3.1.1	Was möchten Artists?	18
3.1.2	Was möchten Designer?	18
3.1.3	Was möchten Entwickler?	18
3.1.4	Projekt bezogen	18
3.1.5	Prozess bezogen	18
3.2	Aktuelle Engines und deren Arbeitsprozesse	18
3.2.1	Prozesse in Game Engines und einem Framework	18

3.2.2	Unreal Engine 4	18
3.2.3	Unity 3D	18
3.2.4	idTech X	18
3.2.5	Weitere	18
3.3	Konzeptentwurf	18
3.3.1	Systemdesign für ein Plugin	18
3.3.2	Systemdesign für einen Project-Handler	18
3.3.3	Entfernen von Abhängigkeiten	18
3.3.4	Zeitersparnis durch bekannte Tools	18
3.3.5	Warum Fusee und Cinema 4D?	18
3.4	Die Implementierung	19
3.4.1	Cinema 4D Plugin API und SDK	19
3.4.2	Uniplug	21
3.4.3	Fusee	22
3.5	Das eigentliche Plugin	22
3.5.1	Visualisierung der Systemarchitektur	22
3.5.2	Generieren eines Fusee Projektes	22
3.5.3	Code Generation und die Vermeidung von Roundtrips (nicht so ganz roundtrips, generierung um generierung etc.)	22
3.5.4	XPresso Schaltungen - Programmieren ohne Program- mieren	22
3.5.5	Partial Classes in .NET	22
4	Ergebnisse und Erkenntnisse	23
4.1	Game Authoring Entwicklungsprozesse jetzt und in Zukunft . .	23
4.2	Wie weit ist die Implementierung fortgeschritten?	23
4.3	Welcher Mehrwert wurde erreicht?	23
4.4	Integration des Systems in den weiteren Projektverlauf von FU- SEE	23
	Verzeichnis der Sourcecode Beispiele	25
	Tabellenverzeichnis	26
	Abbildungsverzeichnis	27
	Literaturverzeichnis	27
	UML Diagramme	30

1 Anforderungen, Ziele und eine Fragestellung

Arbeitsprozesse in heutigen Game Engines verlangen von Entwicklern meist das Erlernen neuer Toolsets und dies während eines meist sehr eingeschränkten Projekt Zeitraums. Es wäre für Entwickler einfacher sich mit den bereits bekannten Tools zu beschäftigen und mit diesen großartige Ergebnisse zu erreichen. Authoring Tools ermöglichen also auch Teammitgliedern ohne weiteres tiefgehend technisches Verständnis für die Programmierung von Spielen, an Projekten in der Entwicklung mitzuarbeiten.

It [Developing “Authoring Tools”, Anmerkung des Autors] is successful due to several factors: it allows non-technical users to work on projects that would otherwise be out of their reach (due to lack of expertise, especially concerning programming languages); it can bring structure into unstructured domains (such as game development) and it can speed up development by streamlining and automating common tasks. Florian, Mehm, 2014

1.1 Motivation

Diese Arbeit beschäftigt sich nun mit der Frage ob es möglich ist ein Tool zu konzipieren welches auf der Basis eines bereits bestehenden Modeling Editors (hier Cinema 4D von Maxon¹) das Erstellen einer “fertigen”² Szene für die 3D Engine Fusee³ ermöglicht. Hierbei wird nach der Konzeption versucht die Basis Funktionalität in Visual Studio mit Hilfe von C# Code und der nach C# gewrappten⁴ Cinema 4D API zu implementieren. Die gewrappte Cinema 4D API basiert auf einem ehemaligen Projekt der Hochschule Furtwangen. Dieses wird als Grundlage für die hier angedachte Implementierung genutzt und bietet einen geringen Umfang an Basisfunktionalität. So bietet es die Möglichkeit

¹MAXON Computer GmbH. (2014-2015). Cinema 4D R16. <http://www.maxon.net/de/products/cinema-4d-studio.html>.

²Build fähige Version einer im Fusee Szenenformat abgespeicherten 3D Szene.

³FUSEE (Furtwangen Simulation and Entertainment Engine - <http://www.fusee3d.org>)

⁴Eine Software welche von einem anderen Stück software umgeben wird.

grundsätzlich Plugins für Cinema 4D in der Programmiersprache C# zu schreiben. Von Haus aus ermöglicht Maxon das Entwickeln von Plugins nur in C++, Python und Coffee (einer von Maxon selbst entwickelten Skriptsprache). Der Vollständigkeit halber sei gesagt, dass Maxon für C++ noch das Framework Melange anbietet welches es ermöglicht Cinema 4D Dateien ohne eine Cinema 4D installation zu erstellen, zu speichern und zu laden. Sollte eine Installation vorhanden sein kann das Plugin auch Szenen rendern.

Szenen in Cinema 4D werden grundsätzlich in einer Art Baumstruktur gespeichert und zur weiteren Verarbeitung im Speicher gehalten. Die hier konzipierte Software möchte diese Tatsache nutzen um Szenen aus einem Modeling Editor (Cinema 4D) in eine Szene des Fusee Szenen Formats (.fus) umzuwandeln. Eine ".fus" Datei ist ebenfalls in einer Baumartigen Struktur gespeichert. Dieses Prinzip der Szenendarstellung ist bereits aus verschiedenen Frameworks und Softwareprojekten für 2D Darstellung bekannt. Das ist zum einen der Übersichtlichkeit als auch verschiedenen Algorithmischen Operationen auf den Daten der Szene geschuldet. In Baumstrukturen organisierte Interfaces werden außerdem bei der Entwicklung des User Interfaces für das Mobile Betriebssystem Android verwendet.

1.2 Ziele der Implementierung

Die während dieser Arbeit implementierte Software hat das Ziel eine Basis für die Verwendung von Cinema 4D als Game Engine Editor aufzubauen. Es werden grundlegende Funktionen in Form einer C# Bibliothek entwickelt die es ermöglichen sollen das Projekt in Zukunft auch für andere 3D Modeling Software anzupassen. Diese Arbeit zielt nicht darauf ab ein komplettes Tool für die Entwicklung von Spielen in der Fusee Engine zu erschaffen. Es wird versucht eine art Grundstein für weitere Forschung und Entwicklung in den Bereich des Game Authoring Toolkit Developments für die Arbeit mit der Akademischen Simulations und Entertainment Software FUSEE zu legen. Das Kernziel ist das Erstellen eines Konzeptes und die Erläuterung der einzelnen Module eines solchen Systems. Verschiedene bereits bestehende Tools und Game Engines werden zu vergleichen herangezogen und wurden im Laufe dieser Arbeit untersucht und getestet.

1.3 Verwendete Software

- Microsoft Visual Studio 2010,
verwendet als Entwicklungsumgebung für das Softwareprojekt.

- Die Erweiterung ReSharper in Version 7.1 für Visual Studio 2010 <http://www.jetbrains.com/resharper>
- Umlet <http://www.umlet.com/>
Ein kostenloses Tool um UML Diagramme zu erstellen.
- GitHub und die GitShell www.github.com
Verwendet als Versionskontrollsystem und als Distributionswerkzeug für den Sourcecode.
- TexWorks www.tug.org/texworks/
Zum Schreiben dieser Arbeit.

2 Grundlegendes

2.1 Entwicklungsprozesse in Interaktiver 3D Software und Games

Um einen Entwicklungsprozess abzubilden und Tools für Entwickler, sogenannte Developer Tools, zu entwickeln bedarf es einer gewissen Organisation. Im Bereich der modernen Spieleentwicklung in kleinen bis mittleren Unternehmen (seltener bei großen AAA Produktionen ¹) wird hierfür ein agiles Modell zur Softwareentwicklung eingesetzt. Hier soll ein kurzer Überblick über aktuelle Modelle entstehen. Diese Modelle ermöglichen zum einen das schnelle Entwickeln von Tools während der knappen Entwicklungszeit eines Spiele Produkts und zum anderen unterstützen sie die Arbeit von kleinen Teams, in welchen meist Tool Development betrieben wird, innerhalb eines großen Entwickler-teams um so gezielt plötzlich auftauchende Aufgaben ohne lange Planung und viel Bürokratie lösen zu können. Damit ist ein fortschreiten des gesamten Projektablaufs gesichert und Entwickler können ihre Zeit hauptsächlich für die Entwicklung der Tools investieren.

2.1.1 Projektmanagement Modelle

Um große Projekte wie Computergames oder Interaktive Software zu entwickeln, bedarf es meist einer detaillierten Planung und einer exakten Rollenverteilung im Entwicklerteam. Es existieren verschiedene Methoden des Projektmanagement auf welche hier kurz im Zusammenhang mit der Arbeit eingegangen werden soll. Einige der Projektmanagement Modelle wirken auf die Arbeitsweise der Teammitglieder aus. Daher wird diese Arbeit hier keinen umfassenden Überblick über Projektmanagement Methoden geben, sondern nur solche Ansprechen die sich direkt oder indirekt stark auf das Tool Development auswirken.

¹Allgemein: Hochqualitative Spiele Software mit großem Entwicklungsbudget und einer Breiten Zielgruppe. Vgl. Cifaldi, 2005

Agile Modelle vs. klassische Modelle

Viele Entwickler (Ubisoft, siehe Schmitz, 2014) setzen heute auf moderne Modelle zum Entwickeln von Software. Die so genannten agilen Modelle (wie beispielsweise Scrum, Extreme Programming und Feature Driven Development) ermöglichen meist das schnelle (agile) reagieren auf plötzlich auftauchende schwierige Situationen. Klassische Modelle (Wasserfallmodell, Spiralmodell) haben hier meist Probleme durch ungleich höhere Bürokratie und Komplexität und benötigen ein Zeitaufwändigeres re-iterieren im Falle von Updates und Umstrukturierungen in Folge von unvorhergesehenen Ereignissen und Problemen. Hochkomplexe Software Projekte die über längere Zeiträume entwickelt werden können meist nur durch klassische Projektmanagement Modelle überblickt und erfasst werden. Allerdings bedeutet der zusätzliche Bürokratische Mehraufwand auch oftmals einen erhöhten Overhead im Personal-, Software- und Knowledge-Bereich. Es ist im Fall des schnell-lebigen Tool Developments also geschickter, sich mit einem ebenso schnell-lebigen und agilen Projektmanagementmodell wie Scrum zu organisieren.

Scrum

Der Scrum Prozess tauchte das erste mal in der Veröffentlichung “The New Product Development Game” von Hirotaka Takeuchi and Ikujiro Nonaka 1986 auf - damals nicht unbedingt in der Software- sondern der allgemeinen Produktentwicklung eingesetzt. Seitdem hat sich das Modell weiter entwickelt und erfreut sich bei innovativen Softwareprojekten im Games und Indie-Games Bereich (auch und meist wohl auch vor allem im Tool Development) sehr großer Beliebtheit. Die Entwickler CCP und Warhorse Studios hatten hierzu eigene Videos und Artikel veröffentlicht, siehe CCP Games, 2009, Schmitz, 2013, Martin Klekner, 2014.

Ein Scrum Entwicklerteam ist mit folgenden Rollen besetzt:

- Product Owner
- Entwicklungsteam
- Scrum Master

Bei diesen Rollen handelt es sich um das interne Scrum Team - das Entwicklungsteam des Produktes. Scrum kann innerhalb eines Projektes und Teams beliebig heruntergebrochen werden, bis die gewünschte gröÙe eines Entwicklerteams erreicht wird. Externe Rollen wie Stakeholder etc. verlagern sich somit auf andere interne Projektleiter oder Teammitglieder. Aus diesem Grund

ist das Model gut für die Entwicklung von Development Tools und Toolkits geeignet. Mit Hilfe des Models, können benötigte Toolkits während einer Projektlaufzeit schnell und effizient entwickelt werden ohne dass ein schwerfälliger Bürokratischer Prozess die Entwicklung blockiert. Somit ergänzt sich dieser Prozess gut mit dem doch eher agilen entwickeln von Development Tools während der Projektlaufzeit - denn in den seltensten Fällen wurde vor dem Beginn des Projekts daran gedacht alle nötigen Tools bereitzustellen. Oftmals ergeben sich auch während der Entwicklung neue Herausforderungen für das Team welche nach neuen Tools verlangen.

Hier soll nurn kurz ein Szenario aufgebaut werden, welches das Tool Development Team eines aktiven Software Entwicklers beschreibt. Zuerst einmal sollen die Rollen verteilt werden:

- Product Owner - Meist der leitende Entwickler des Software Projektes. In diesem Fall meist ein Producer und/oder Game Developer.
- Entwicklungsteam - Das Tool Development Team selbst.
- Scrum Master - Die leitende Person des Tool Development Teams, bzw. sollte sich das Team sehr nah am Scrum Modell bewegen, dann meist ein Entwickler außerhalb des Teams aber mit guten Kontakten zum Team selbst und erhöhter Erreichbarkeit.

Die Stakeholder des Tools wären in diesem Fall die anderen Entwickler des Unternehmens die das Produkt im Produktiefbetrieb einsetzen möchten. Es kann hierbei auch von Vorteil sein, das Tool iterativ in den Arbeitsalltag des Teams zu integrieren um die Entwickler nicht durch einen Berg an neuen Features zu verunsichern und so die Einarbeitungszeit möglichst gering zu halten.

Es soll hier an einem kurzen Beispiel deutlich gemacht werden, wie ein solches Tool eingeführt werden könnte:

Szenario: Ein Team benötigt einen Textur-Editor / Tool um Texturen in das Format der Game-Engine zu transformieren.

- Der Antrag für das Tool vom Producer/Entwickler/oder anderen Personen wird gestellt.
- Das Tool wird bewilligt und das Tool Development Team wird beauftragt.
- Das Team entwickelt designed das Tool und implementiert Basisfunktionalität.

- Das Tool wird mit der Basisfunktionalität an das Produkt Team herausgegeben.
- Die fehlenden Funktionen werden implementiert.
- Das Tool wird mit der erweiterten Funktionalität herausgegeben.
- Es wird mit dem Produkt Team Rücksprache gehalten, welche Funktionen noch benötigt werden.

Dieser Prozess schließt iterativ ab bzw. nicht ab, da während der Entwicklung einer interaktiven Anwendung / Games eventuell auch auf externe Einflüsse wie Third Party Software oder Marktentwicklungen eingegangen wird. Als Beispiel: So könnte sich durch die Veröffentlichung einer neuen GPU Generation oder den vorgezogenen PC Release die Größe der benötigten Textur-Dateien ändern.

2.1.2 Internes Tool Developing anstatt Tool licencing

Internes Tool Development ist ein wichtiger Aspekt im Team eines Games und Software Entwicklerteams. Erich Bethke berichtet in davon, dass Michael Abrash² ihm einst mitteilte, “dass 50% der Entwickler Arbeit bei idSoftware in das Tool Development fliesse.” vgl. Bethke, E. (2003). *Game Development and Production*. Wordware Publishing Inc.; Auflage: Pap/Cdr (Februar 2003), S. 44. An der Relevanz des Themas hat sich trotz des zurückliegenden Zeitraums kaum etwas getan. Sony hat für den Release der Playstation 4³ ein Development Kit⁴ für die internen Entwickler Studios erstellen lassen, welches bereits während der Planung und Entwicklung der Konsole entwickelt wurde. Sony hat diese Prozedur perfektioniert und lässt die eigenen Tools sogar in einem eigens dafür gegründeten Unternehmen für die eigenen Studios erstellen⁵. Sony hat im Herbst 2014 den für Playstation 3 Spiele eigens intern entwickelten Welt Editor “Level Editor”⁶ als Open Source Software veröffentlicht und für Jedermann auf GitHub verfügbar gemacht. Der Editor kommt ohne direkten Enginebezug aus und lässt sich somit für verschiedenste Projekte der Sony Studios anpassen. Verschiedene Entwickler haben darauf ihre eigenen Tool Kits und Editoren auf dem von Sony bereitgestellten ATF Framework und “Level Editor” erstellt um Spiele wie Naughty Dogs Uncharted⁷, Guerilla

²Ehemals idSoftware, ehemals Valve VR, aktuell Oculus VR Chief Scientist

³Playstation 4 - Erschienen im Herbst 2013

⁴Freeman, W. (2014). Kingdom come: deliverance video update 9.

⁵SNSystems <http://www.snsystems.com/>

⁶Wawro, A. (2014). Sony releases level editor that's open source and engine-agnostic.

⁷NaughtyDog. (2007, 2009, 2011). Uncharted. http://www.naughtydog.com/games/uncharted_drakes_fortune/. Sony Computer Entertainment.

Games' Killzone Serie⁸ oder Quantic Dreams Beyond:Two Souls⁹ zu erstellen. Diese Tatsache zeigt, dass selbst in großen Studios immernoch bedarf nach einfach und schnell zu erweiternden Frameworks und Editoren besteht. Das ATF Framework bzw. der "Level Editor" von Sony waren auch ein Anlass diese Arbeit

2.2 Mitglieder eines Entwicklerteams

Es soll hier ein kurzer Überblick über die gängigsten Mitglieder eines Entwicklerteams gegeben werden. Grob können Mitglieder in die folgenden drei Gruppen aufgeteilt werden - Artist, Designer, Engineer. Jede Gruppe arbeitet hierbei interdisziplinär mit den anderen zusammen, kümmert sich aber doch um die ganz eigenen Bestandteile eines Produktes. Es ist jedoch durchaus so, dass jede Gruppe ihre eigenen Tools und Methoden verwendet. Dieser Ansatz wird in der Konzeptionierung dieser Arbeit aufgegriffen und weiter verfolgt.

Bei der Bezeichnung und Aufteilung der verschiedenen Teammitglieder in Fachbereiche orientiert sich diese Arbeit am Werk von Chandler, H. (2006). *The Game Production Handbook*. Thomson Delmar Learning, in welchem er die Produktionsprozesse eines Spiels sowohl in designtechnischer Weise als auch aus technischer Sicht beschreibt.

2.2.1 Artists

Artists sind in einem Games Projekt für jegliche Repräsentation der Spiellogik nach Außen zuständig. Sie erstellen Modelle von Spielfiguren und Umgebungen und kreieren Texturen und User Interfaces. Bei den Artists handelt es sich um eine Kerngruppe für diese Arbeit da sie einen Großteil der Arbeitszeit in den Tools und Editoren des Spiels verbringt. Artists können in mehrere Untergruppen aufgeteilt werden. Dies bedeutet jedoch nicht, dass jedes Unternehmen jede Artists Rolle beschäftigt. Oftmals übernehmen einzelne Mitarbeiter mehrere Rollen je nach dem Entwicklungsstand des Projekts.

⁸Guerrilla Games. (2004, 2007, 2011, 2013, 2013). Killzone 1, Killzone 2, Killzone 3, Killzone: Mercenary, Killzone: Shadow Fall. <http://www.guerrilla-games.com/>. Sony Computer Entertainment.

⁹Quantic Dream. (2013). Beyond: Two Souls. <http://www.quanticroam.com/en/>. Sony Computer Entertainment.

Modeling/Animation Artist

Ein Animation Artist verbringt die meiste Zeit damit Animationen und Modelle (3D, 2D) für die Verwendung im Spiel vorzubereiten - kurz: Assets¹⁰. Programme wie Cinema 4D¹¹, 3DS Max¹², oder Modo¹³ sind Beispiele für Kernsoftware dieser Entwickler. Der Vollständigkeit halber sei hier noch das Open Source Projekt Blender¹⁴ erwähnt.

Environment Artist

2.2.2 Designer

Designer (Gamedesigner) arbeiten eng mit Artists und Engineers zusammen. Meist entwickeln Game Designer das Spielprinzip, den Raum des Spiels und das Regelwerk. Sie schreiben oft Skripte und kleine Implementierungen oder verbessern Grafiken oder Spielfunktionen. Sie verwenden Assets aus der Designabteilung und fügen diese mit Skripten zusammen. Spieltests werden von Ihnen überwacht um den Spielfluss und das Erlebnis des Rezipienten beim Spielen zu optimieren.

Level/World Designer

Level bzw. World Designer erstellen aus den erschaffenen Assets eine oder mehrere zusammenhängende Spielwelten - sogenannte Level. Diese Welten werden durch sie und weitere Artists mit Inhalt nach den Plänen der Game Designer gefüllt. Oft haben diese Welten einen gewissen gestalterischen Anspruch und von den Designern erwünschten Artstyle welche die Atmosphäre des Spiels repräsentiert. Meistens werden diese Welten in einem extra dafür geschaffenen Editor angefertigt und können sich in einem Modeling Tool wie Cinema 4D entwickelt werden. Ein Beispiel für solche Level Editoren ist der GTKRadiant¹⁵ Editor für Spiele basierend auf der idTech3 und idTech4 Engine¹⁶, beide als Open Source auf der Plattform GitHub verfügbar. Weitere Beispiele sind der Level Editor von Sony, auf welchen diese Arbeit später noch eingeht sowie der Unity3d Editor. Der Unity3d beinhaltet eine gesamte Game Engine, jedoch

¹⁰ Assets sind Bestandteile des Produktes welche eine Grafische oder logische Repräsentation im Produkt erfahren. Dazu zählen z.B. Modelle, Texturen und Code Dateien.

¹¹ MAXON Computer GmbH, 2014-2015.

¹² Autodesk, Inc., 2015.

¹³ The Foundry Visionmongers Ltd., 2015.

¹⁴ Blender Foundation, 2014-2015.

¹⁵ Open Source Projekt GTKRadiant <http://icculus.org/gtkradiant/>

¹⁶ Beide Engines und weiterer Source Code von idSoftware herunterladen auf dem Account des Unternehmens auf GitHub unter <https://github.com/id-Software> - geprüft am 08.04.2015

wird direktes Modeling und das erstellen von Texturen von Grund auf nicht unterstützt. Alle Assets, außer primitiver Geometrischer Objekte wie Würfel und Kugeln etc. müssen in externen Programmen erstellt und importiert werden.

Die Konzeptionierung dieser Arbeit wird nun die versuchen einen Ansatz zu entwickeln der es ermöglicht zumindest einen Teil der Level und Welteditor Tools zu beseitigen. Somit könnten Level und World Designer und Environment Artists ihre Arbeit in die bereits bekannten Modeling Tools verlagern und so eine verbesserte Produktivität erreichen.

Scripter

Scripter sind meist dafür Zuständig verschiedene Ereignisse in einer für die Game Engine extra entwickelten Script Sprache zu beschreiben und so die Welt des Spiels interaktiver zu gestalten. Diese Aufgaben unterstützen die Spiellogik oder aber beschreiben die Funktionen ganzer Systeme wie z.B. die eines Aufgabensystems (Quest Systems) welches dem Spieler während des Spiels mitteilt, was er in der Spieltwelt zu tun hat. Hier sind allerdings viele Bestandteile eines Spiels anzuordnen. Meist werden

User Interface Designer

User Interface Designer kümmern sich um das Erstellen von grafischen Schnittstellen welche die Interaktion mit dem Benutzer ermöglichen. Hierfür verwenden sie oft Scriptsprachen wie Actionscript von Adobe (Zur programmierung von Adobe Flash Interfaces) oder gar fertige Middleware wie Scaleform¹⁷ ein Cross Plattform UI Solution Tool¹⁸ von Autodesk. Diese Gruppe der Entwickler wird durch diese Arbeit nur sehr gering beeinflusst. In der Fusee Engine werden Interfaces über Code Dateien eingebunden und daher in externen Grafikprogrammen und Visual Studio angefertigt.

2.2.3 Engineer

Engineers / Ingenieure arbeiten meist am Kern der Applikation und schreiben den Source Code für die Anwendung, Engine, Netzwerkfunktionen, KI, und Tools. Diese Entwickler arbeiten hauptsächlich in einer IDE¹⁹ wie Visual Studio (auf welches sich das zu dieser Arbeit konzeptionierte Tool bezieht) oder

¹⁷Autodesk Inc., 2014-2015.

¹⁸Ermöglicht das erstellen von 2D, 2.5D und 3D Ui Elementen. Wird z.B. von der Unreal Engine 4 verwendet.

¹⁹Integrated Development Environment

XCode ²⁰. Der in der IDE geschriebene Code wird dann von den Engineers selbst oder von Game Designer in der Engine verwendet. Hierbei kann sich das Tätigkeitsfeld ausweiten bis hin zur Entwicklung von Gamelogs ²¹.

Tool Engineer

Diese Arbeit bezieht sich auf den Bereich des Tool Development. Hierbei entwickelt ein kleines Team - meist während oder vor der eigentlichen Arbeit an einem Projekt die Tools für die restlichen Entwickler des Projektes. Diese Tool Palette kann von Textur editoren bis hin zu kompletten Welt editoren fast alles vorstellbare enthalten. Verschiedene Studios haben eigene Tool Developer Teams, welche sich nur um diesen Bereich des Produktes kümmern. Diese Teams betreuen auch meist den Modding Support für ein fertiges veröffentlichtes Produkt. Beispiele für Modding Tools sind z.B. das RedKit von CDProject Red für das Spiel The Witcher 1 und 2, der LevelEditor von Sony der in einer Open Source Version vorliegt oder das Creation Kit von Bethesda Softworks welches einen Modding Support für die Spiele der The Elder Scrolls Reihe bereit stellt.

Computer-Graphics Engineer

Computer Graphics Engineers beschäftigen sich meist mit dem entwickeln der eigentlichen Engine. Oft sind Graphics Ingenieure aber auch an der Tool Entwicklung beteiligt. Gerade in kleineren Unternehmen könnten die eigentlichen Strukturen schnell aufbrechen um Synergien zu nutzen.

2.2.4 Weitere für diese Arbeit nicht relevante Teammitglieder

In diesem Abschnitt sollen noch die restlichen Mitglieder des Teams erwähnt werden, welche Tools nutzen aber meist nicht an der Entwicklung der selbigen beteiligt sind.

- Sound Engineer
- Bla
- Bla
- Bla
- Bla

²⁰X-Code ist nur für MacOSX erhältlich

²¹Logik des Spiels, ermöglicht das interagieren etc. mit und in der Software

2.3 Stakeholderanalyse intern

Um herauszufinden, welche Entwicklergruppen eines Teams von neuen Development Tools im Bezug auf Fusee betroffen wären, wurde eine Analyse durchgeführt, um die Stakeholder innerhalb des Teams abzubilden. In der Praxis wäre für die Konzeption bzw. das Systemdesign eines neuen Tools der Besuch der von der Tool Entwicklung betroffenen Teammitglieder am Arbeitsplatz und das beobachten der jeweils verrichteten Aufgaben sehr aufschlussreich. Durch diese Methode könnten Probleme im Arbeitsablauf frühzeitig identifiziert, behoben und besonders wichtige Features rechtzeitig vor Beginn der Implementierung in Erfahrung gebracht und geplant werden.

Eine Bedarfsanalyse und eine allgemeine Analyse der Aufgabengebiete der jeweiligen Entwickler sollte in das System Design bzw. das Requirements Engineering ebenfalls mit einfließen. Hierzu könnten interviews mit den Entwicklern geführt werden. Dies ist eine schnelle Methode welche sich gut mit einem agilen Prozess wie Scrum vereinbaren lässt.

2.4 Ein Arbeitsprozess wird entwickelt

2.4.1 Game Authoring / Game Development

Game Development beschreibt allgemein das entwickeln einer Interaktiven Software, meist eines Spiels. Hierbei spielt es keine Rolle ob die Software der puren Unterhaltung dient oder sich dem Genre des Serious Games zuordnen lässt. Auch weitere interaktive Anwendungen ohne Unterhaltungsfaktor können in diese Kategorie fallen. Die Produktionsabläufe ähneln sich stark. Im Gegensatz zum Tool Development hat das Game Development meist den Auftrag am Ende ein für den Consumer zugängliches Produkt zu schaffen. Das Tool Development beschäftigt sich in erster Linie mit dem Erstellen der Werkzeuge welche benötigt werden um das Consumerprodukt zu entwickeln.

Es ist aber in der Tat so, dass Tool Development auch auf dem Consumer Markt ankommen kann. Verschiedene Entwickler stellen in der Vergangenheit ihre Tools den Fans und Kunden zur Verfügung. Aus diesen Tools haben sich selbst schon wieder Produkte wie Modifikationen²², Patches oder Erweiterungen entwickelt. Einige Beispiele sind die das Creation Kit von Bethesda Softworks und die daraus entstandenen zahlreichen Fan Modifikationen wie Nehrim oder Enderal des Mod Teams SureAI²³.

²²auch Mods genannt, Verändern das eigentliche Spiel oder ersetzen es in Form einer "Total Conversion" gleich ganz durch neue, eigens entwickelte Inhalte

²³Freies Modifikations Team. <http://www.sureai.net>

2.4.2 Tool Development

Nach Wihlidals Auffassung (Wihlidal, 2006) unterscheidet sich die Planung eines Projektes zur Erstellung eines Developer Tools nicht sehr von der Planung zur allgemeinen Entwicklung von Software. Es gelten hier vier Planungsphasen die den Projektablauf kennzeichnen. Der erste Schritt wäre eine allgemeine Planung des Tools. Diese beinhaltet Funktionen und Umfang des Tools. Eine Beschreibung der Anforderungen bzw. der Ziele des Projektes wird mit den begünstigten Entwicklern abgesprochen.

Die zweite Phase beschreibt eine Bedarfsanalyse der Stakeholder. Es werden Arbeitsabläufe skizziert und mit den Beteiligten durchgesprochen. Je nach Komplexität und Relevanz des Projekts wird Software anderer Hersteller oder anderer Arbeitsbereiche ebenfalls analysiert und das Ergebnis zur Gesamtanalyse hinzugezogen. So könnte eventuell der Einsatz einer Drittanbieter Software in gewissen Situationen vorteilhafter sein als eine komplette interne Neuentwicklung. Diese Entscheidung ist aber sehr Situationsabhängig.

Daraufhin folgt die Designphase in welcher das Entwicklerteam des neuen Tools das Requirements Engineering abschliesst und mit dem Software-/Systemdesign beginnt. Hier wird das Produkt in Form von UML Diagrammen und Veranschaulichungen entwickelt. Die tatsächliche Implementierung folgt als letzte Phase des Projektablaufs. Während der Implementierung kann durch das agile Scrum System immer wieder iterativ an den Features des Tools gearbeitet werden.

Im folgenden Abschnitt wird noch etwas genauer auf die jeweiligen Schritte der Entwicklung eingegangen.

Planung

Es folgt die Planungsphase des Tools. Hier wird zuerst eine Requirementsanalyse ²⁴ durchgeführt. Mit ihr sollen alle wichtigen Kerneigenschaften der Software identifiziert und niedergeschrieben werden. Ein an diese Arbeit angehängtes Designdokument führt diese Requirementsanalyse weiter aus. Die Anforderungsanalyse ist in einer solchen Situation, in welcher ein Produkt unter Zeitdruck für den Produktivbetrieb entwickelt wird, ein wichtiger Bestandteil der Projektplanung. Ein Tool, welches nicht den Anforderungen der Teammitglieder entspricht kann nicht im Betrieb eingesetzt werden und verzögert im schlimmsten Fall die weitere Entwicklung des gesamten großen Softwareprojektes.

²⁴dt. Anforderungsmanagement

Requirements Analyse

- Eine Software soll es ermöglichen, dass Artists, Designer und Developer an ein und dem selben Projekt arbeiten können ohne die gewohnte Arbeitsumgebung (3D-Modellierungssoftware, IDE) zu verlassen und etwas komplett neues (Level-Editor) zu erlernen.
- Das Produkt muss auf Basis der FUSEE Engine entstehen
- Ziel ist es in Cinema 4D ein FUSEE Projekt anzulegen, zu speichern und es zu öffnen
- Assets/Bollen ins Spiel integriert werden können die von Artists, Designern und Entwicklern bearbeitet werden können.
- Das FUSEE C# Projekt sollte aus C4D heraus gebaut werden können.
- Eine Stakeholderanalyse schafft Klarheit, welche Parteien des Teams mit dem zu erstellenden Tool arbeiten müssen.
- Es ist zu analysieren, welche Schritte für welche Art der Arbeit des Teams notwendig sind. Hierzu werden Usecases der verschiedenen Rollen und Aufgaben erstellt.

Requirements Dokumentation

Während der Requirements Dokumentation werden alle Ansprüche an die Software dokumentiert und von den Stakeholdern geprüft. Über dieses Dokument lässt sich der spätere Funktionsumfang des Tools genauestens definieren und verfolgen. Das Dokument stellt sozusagen ein rechtlich relevantes Dokument dar.

System Design / System Modeling

Anschließend an die Analyse folgt das System Modeling in welchem die Anforderungen des Programs zu einem Softwareprodukt modelliert werden. Oft bedient sich das Entwicklerteam hierbei Notationen wie UML ²⁵ in Diagramm und Schrift Form. Das System Design ist ein kritischer Punkt. Hier müssen die Anforderungen des Kunden genau in die geplante Entwicklung des Systems übernommen werden. Oftmals arbeitet das System Design

²⁵Unified Modeling Language

Abgleich des System Designs mit den Anforderungen

Vor der Implementierung muss immer auch geprüft werden ob das geplante System aus dem Designprozess mit den tatsächlichen Anforderungen der Stakeholder zusammen passt.

Implementierung

Die Implementierung ist der praktische Schritt des ganzen Prozesses. Während der Implementierung wird das Tool entwickelt, auf Fehler geprüft eventuell auch durch einen iterativen Entwicklungsprozess erweitert.

Asset Pipeline und Feedback

SEHR KRITISCHE MARKE! Das Asset aus dem Editor in die Engine bekommen, die Darstellung etc kontrollieren. Zeitkritisch beim export etc. Carter Seite 6 und folgende.

2.5 Die Struktur von Game Assets

Assets und das aufbrechen in Bestandteile eines Projektes. Level etc. bestehen aus Assets und mehr.

2.5.1 Warum eine Trennung von Code und Content?

3 Entwicklung eines Konzeptes

3.1 Use Cases der verschiedenen Entwickler

3.1.1 Was möchten Artists?

3.1.2 Was möchten Designer?

3.1.3 Was möchten Entwickler?

3.1.4 Projekt bezogen

Projekt anlegen

Projekt öffnen

Projekt speichern

Projekt bauen

In Projekt einsteigen

Projekt klonen etc.

3.1.5 Prozess bezogen

Gleichzeitig an Projekt arbeiten

Model Datei importieren

Gleichzeitig an einem Objekt arbeiten

3.2 Aktuelle Engines und deren Arbeitsprozesse

3.2.1 Prozesse in Game Engines und einem Framework

3.2.2 Unreal Engine 4

3.2.3 Unity 3D

3.2.4 idTech X

3.2.5 Weitere

3.3 Konzeptentwurf

3.3.1 Systemdesign für ein Plugin

3.3.2 Systemdesign für einen Project-Handler

3.3.3 Entfernen von Abhängigkeiten

3.3.4 Zeitersparnis durch bekannte Tools

- Es soll möglichst wenig "geparsed" oder "konvertiert" werden
- Dateien sollen Version Control kompatibel bleiben (wenig bis keine Binaries)
- Das Projekt muss Strukturiert sein
- Eine Fusee Solution soll gehandelt werden können

3.4 Die Implementierung

Das Projekt "Fusee Authoring Toolkit" basiert auf mehreren Software Projekten. Ein Teil dieser Projekte wurde an der Hochschule Furtwangen entwickelt. Ein anderer Teil stammt von externen Entwicklern. Dieser Abschnitt gibt einen Überblick über die verwendeten Softwarebestandteile des Toolkits und ihrer Einbindung.

3.4.1 Cinema 4D Plugin API und SDK

Cinema 4D (MAXON Computer GmbH, 2014-2015) ist ein kommerzielles proprietäres Produkt des Unternehmens MAXON Computer GmbH und steht nicht als Open Source Projekt zur Verfügung. Maxon stellt für die Entwicklung von Plugins¹ eine API² bereit. Bei dieser API handelt es sich um eine Bibliothek aus C++ Code. Dieser Code kann in Form von Header Files in die eigene Applikation integriert werden. Durch diese C++ Headerfiles ergibt sich eine Schnittstelle zu internen Methoden in der Cinema 4D Software. Die tatsächlichen Methoden können mit der API nicht eingesehen werden, es handelt sich hier lediglich um Header Files ohne den Code der tatsächlichen Implementierung. Das C++ SDK wird bei einer Installation von Cinema 4D automatisch mitinstalliert. Auf einem Windows PC findet sich das SDK und die dazugehörigen Visual Studio Solution Dateien unter folgendem Pfad ausgehend vom Installations Ordner der 64Bit³: "Cinema 4D/plugins/cinema4dsdk"

Maxon stellt jegliche SDKs und Beispiele auf seinem GitHub Account unter der Apache License Version 2.0, January 2004⁴ zur freien Verfügung <https://github.com/PluginCafe>. Das C++ SDK ist unter dieser Adresse einsehbar: https://github.com/PluginCafe/cinema4d_cpp_sdk

¹Plugins in C4D - Erweiterungen für die Software Cinema 4D und deren Funktionen

²Application Programming Interface, z. Dt. Programmierschnittstelle

³Cinema 4D liegt seit der Version R15 nur noch als 64 Bit Version vor und benötigt laut <http://www.maxon.net/?id=311> (Maxon C4D System Requirements) ein 64 Bit System

⁴Apache Licence 2.0 https://github.com/PluginCafe/cinema4d_cpp_sdk/blob/master/LICENSE

Das folgende Schaubild erläutert die Verwendung der API. Es ist in der Tat so, dass der eigene Plugin Code laut Maxons Dokumentation einige Funktionen überschreiben bzw. Klassen vererben muss. Ansonsten ist der Inhalt des Plugin Codes nicht beschränkt.

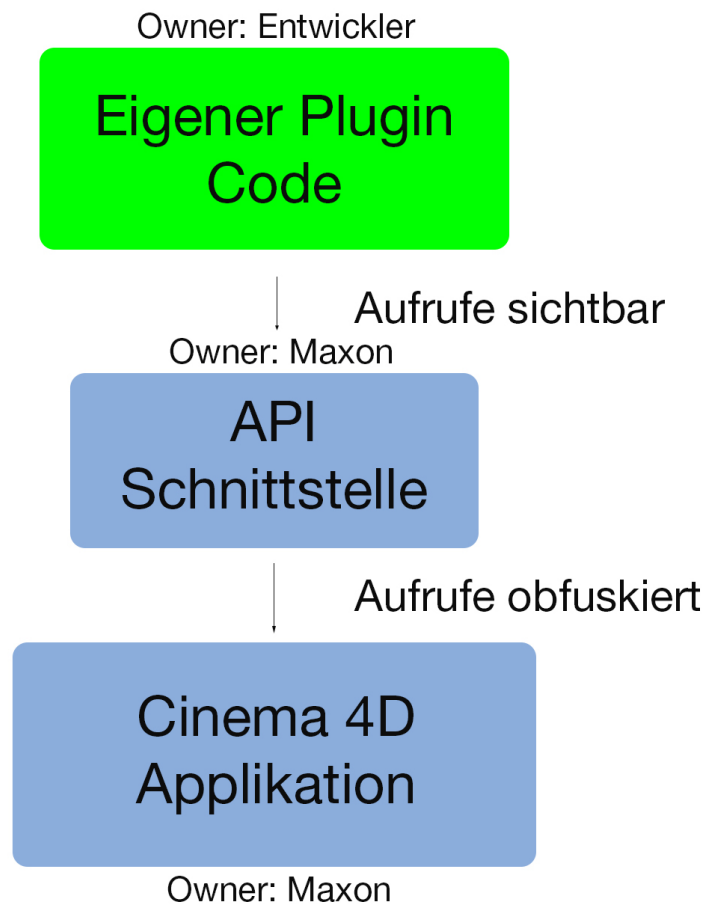


Abbildung 3.1: Schaubild des Cinema 4D API Systems

Das hier dargestellte Minimalbeispiel in C++ Code zeigt ein einfaches minimalistisches Plugin, welches beim Ausführen nur eine Ausgabe auf der Konsole von Cinema 4D erzeugt.

Code/minimalbeispielPlugin.h

```
1 #define ID_PLUGINID 1234567 // Dies ist eine einzigartige
   Plugin ID.
2
3 #include "c4d.h"
4
```

```

5  class MinimalesPlugin : public CommandData // Das Plugin ist
    vom Plugin Typ "Command Plugin"
6  {
7  public:
8      virtual Bool Execute(BaseDocument *doc)
9      {
10         GePrint("Eine Ausgabe zum Cinema 4D Konsolenfenster.");
11         return TRUE;
12     }
13 };
14
15 // Mit dieser funktion steigt das Plugin ein.
16 Bool PluginStart(void)
17 {
18     return RegisterCommandPlugin(ID_PLUGINID, "MinimalesPlugin"
        , 0, NULL, String("MinimalesPlugin"), gNew MinimalesPlugin)
        ;
19 }
20
21 // Wird beim beenden des Plugin aufgerufen.
22 void PluginEnd(void)
23 {
24 }
25
26 // Ist teil des Message Systems von C4d. Dient der
    Kommunikation.
27 Bool PluginMessage(LONG id, void *data)
28 {
29     return TRUE;
30 }

```

3.4.2 Uniplug

Wrapping von C++ Code nach C# mit SWIG

Erweiterung des Uniplug Codes für Plugins vom Typ TagPlugin

Probleme

Erst entwicklung von Uniplug erweitern bevor mit dem Plugin gearbeitet werden kann. Probleme durch das Wrappen und die fehlende Möglichkeit zu debuggen.

3.4.3 Fusee

Der Fusee Szenengraph

Das Fusee Level, Welt wie auch immer es hier bezeichnet werden sollte. Eine Basis wird gebraucht. Hierzu eine Zentrale anlaufstelle, ein Spiele “Kernel”? Irgend etwas dass im Zentrum steht. Alles andere muss auf Level etc aufgeteilt werden und bis zum einzelnen Asset heruntergebrochen werden.

3.5 Das eigentliche Plugin

3.5.1 Visualisierung der Systemarchitektur

Schaubilder und Grafiken sind hier nützlich. Abhängigkeiten und Aufteilung sinnvoll gestalten.

Welche Programme und Systeme sind beteiligt?

Kette aufzeigen von Cinema4D nach Fusee Authoring Toolkit.

3.5.2 Generieren eines Fusee Projektes

Ablauf erklären.

3.5.3 Code Generation und die Vermeidung von Roundtrips (nicht so ganz roundtrips, generierung um generierung etc.)

3.5.4 XPresso Schaltungen - Programmieren ohne Programmieren

Ein Ausblick auf die Zukunft. Eventuell hier den Artikel anführen welcher über Visuelle Programmierung berichtet hat.

3.5.5 Partial Classes in .NET

Sind bis jetzt noch nicht verwendet worden.

4 Ergebnisse und Erkenntnisse

4.1 Game Authoring Entwicklungsprozesse jetzt und in Zukunft

4.2 Wie weit ist die Implementierung fortgeschritten?

4.3 Welcher Mehrwert wurde erreicht?

4.4 Integration des Systems in den weiteren Projektverlauf von FUSEE

Anhang

Verzeichnis der Sourcecode Beispiele

Tabellenverzeichnis

Abbildungsverzeichnis

3.1	Schaubild des Cinema 4D API Systems	20
-----	---	----

Literatur

- Abrash, M. (1997). *Graphics Programming Black Book*. The Coriolis Group Inc.
- Autodesk Inc. (2014-2015). Autodesk Scaleform. <http://gameware.autodesk.com/scaleform>.
- Autodesk, Inc. (2015). 3DS Max. <http://www.autodesk.de/products/3ds-max/overview>.
- Bethke, E. (2003). *Game Development and Production*. Wordware Publishing Inc.; Auflage: Pap/Cdr (Februar 2003).
- Blender Foundation. (2014-2015). Blender 2.74. <http://www.blender.org/>.
- Carter, B. (2004). *The Game Asset Pipeline*. Delmar Cengage Learning.
- CCP Games. (2009). Eve online fanfest 2009 - scrum and agile development processes.
- Chandler, H. (2006). *The Game Production Handbook*. Thomson Delmar Learning.
- Cifaldi, F. (2005). E3 report: the path to creating aaa games.
- Florian, Mehm, S. R., Christian, Reuter. (2014). Projektmanagement mit scrum. S. 536-541.
- Freeman, W. (2014). Kingdom come: deliverance video update 9.
- Gamma, E., Johnson, R., Helm, R. & Vlissides, J. (2014). *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Deutschland.
- Gergory, J. (2009). *Game Engine Architecture*. Taylor & Francis Ltd.
- Guerrilla Games. (2004, 2007, 2011, 2013, 2013). Killzone 1, Killzone 2, Killzone 3, Killzone: Mercenary, Killzone: Shadow Fall. <http://www.guerrilla-games.com/>. Sony Computer Entertainment.
- Lengyel, E. (2012). *Mathematics for 3D Game Programming and Computer Graphics*. Course Technology, Cengage Learning.
- Martin Klekner, D. V., Votja Nedved. (2014). Kingdom come: deliverance video update 9.
- MAXON Computer GmbH. (2014-2015). Cinema 4D R16. <http://www.maxon.net/de/products/cinema-4d-studio.html>.

- NaughtyDog. (2007, 2009, 2011). Uncharted. http://www.naughtydog.com/games/uncharted_drakes_fortune/. Sony Computer Entertainment.
- Nilsson, T. (2014). Sony releases free, open source game development toolset.
- Quantic Dream. (2013). Beyond: Two Souls. <http://www.quanticroam.com/en/>. Sony Computer Entertainment.
- Schmitz, C. (2013). The art of waiting.
- Schmitz, C. (2014). Projektmanagement mit scrum. Available online - last checked 07.04.2015 <http://www.makinggames.biz/features/projektmanagement-mit-scrum,2534.html>.
- The Foundry Visionmongers Ltd. (2015). Modo. <https://www.thefoundry.co.uk/products/modo/>.
- Wawro, A. (2014). Sony releases level editor that's open source and engine-agnostic.
- Wihlidal, G. (2006). *Game Engine Toolset Development*. Thomson Course Technologies.

UML Diagramme
