HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

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 eingeschhränkten 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 "managedßich 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 betrachetet 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 selbständig 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 10. 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

Arbeitsprozesse in heutigen Game Engines verlangen von Entwicklern meist das erlernen neuer Toolsets und dies während eines meist sehr eingeschhränkten Projekt Zeitraums. Es wäre für Entwickler einfacher sich mit den bereits bekannten Tools zu beschäftigen und mit diesen großartige Ergebnisse zu erreichen.

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 Maxon[1]) das erstellen einer "fertigen"[2] Szene für die 3D Engine Fusee ermöglicht. Hierbei wird nach der Konzeption versucht die Basis Funktionalität in Visual Studio mit Hilfe von C# Code und der nach C# gewrappten 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 grundsätzlich Plugins für Cinema 4D in der Programmiersprache C# zu schreiben. Von Haus aus ermöglicht Maxon das schreiben 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 als eine Art Baum gespeichert und zur weiteren Verarbeitung im Speicher gehalten. Die hier konzipierte Software möchte diese Tatsache nutzen um eine Cinema 4D Szene in eine Szene des Fusee Szenen Formats (.fus) umzuwandeln. Eine ".fus" Datei ist ebenfalls in

[1] MAXON Computer GmbH. (2014-2015). Cinema 4D R16. http://www.maxon.net/de/products/cinema-4d-studio.html.
[2] Build fähige Version einer im Fusee Szenenformat abgespeicherten 3D Szene.

einer Baumartigen Struktur gespeichert. Dieses Prinzip der Szenendarstellung ist bereits aus verschiedenen Frameworks und Softwareprojekten für 2D Darstellung bekannt. So benutzten

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 welche es ermöglichen sollen das Projekt in Zukungt 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, sondern versucht eine art Grundstein für weitere Forschung und Entwicklung in den Bereich des Game Authoring Toolkit Developments 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 Developement betrieben wird, innerhalb eines großen Entwicklerteams 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 Projektmangament Methoden geben, sondern nur solche Ansprechen die sich direkt oder indirekt stark auf das Tool Development auswirken.

[1] Allgemein: 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.

Scrum

Der Scrum Prozess tauchte das erste mal in der Veröffentlichung "The New 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 Softareprojekten 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 größe 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 Developement 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.

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 Game Development and Production davon, dass Michael Abrash [2] ihm einst mitteilte, "dass 50% der Entwickler Arbeit bei idSoftware in das Tool Development fliesse."[3]. Nun ist das bereits eine Weile her, allerdings hat sich an der Relevanz des Themas kaum etwas getan. Sony hat für den Release der Playstation 4 ein Development Kit[4] 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 erstellen [5]. Sony hat im Herbst 2014 den für Playstation 3 Spiele eigens entwickelten Welt Editor "Level Editor"[6] als Open Source Software veröffentlicht und für Jedermann auf GitHub verfügbar gemacht. Der Editor kommt ohne Enginezug 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 erstellt um Spiele wie Naughty Dogs Uncharted[7], Guerilla Games' Killzone Serie[8] oder Quantic Dreams Beyond:Two Souls[9] 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] Ehemals idSoftware, ehemals Valve VR, aktuell Oculus VR Chief Scientist

[3] Bethke, E. (2003). *Game Development and Production*. Wordware Publishing Inc.; Auflage: Pap/Cdr (Februar 2003).

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

[5] SNSystems http://www.snsystems.com/

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

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

[8] 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.

[9] Quantic Dream. (2013). Beyond: Two Souls. http://www.quanticdream.com/en/. Sony Computer Entertainment.

2.2 Mitglieder eines Entwicklerteams

Hier gibt diese Arbeit einen kurzen Überblick über die gängigsten Mitglieder eines Entwicklerteams. Grob können Mitglieder in die folgenden drei Gruppen aufgeteilt werden - Artists, 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, 2006) 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 kreiiren 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.

Modeling/Animation Artist

Ein Animation Artist verbringt die meiste Zeit damit Animationen und Modelle (3D, 2D) für die Verwendung im Spiel vorzubereiten - kurz: Assets [10]. Programme wie Cinema 4D[11], 3DS Max[12], oder Modo[13] sind Beispiele für Kernsoftware dieser Entwickler. Der vollständigkeit halber sei hier noch das Open Source Projekt Blender[14] erwähnt.

[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.

Environment Artist

2.2.2 Designer

Designer (Gamedesigner) arbeiten eng mit Artists und Engineers zusammen. Meist Entwicklen 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 Atmosphähre des Spiels repräsentiert. Meistens werden diese Welten in einem extra dafür geschaffenen Editor angefertigt und können nich in einem Modeling Tool wie Cinema 4D entwickelt werden. Ein Beispiel für solche Level Editoren ist der GTKRadiant[15] Editor für Spiele basierend auf der idTech3 und idTech4 Engine [16], beide als Open Source auf der Platform 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 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 nurn 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 Projektivität erreichen.

[15] Open Source Projekt GTKRadiant http://icculus.org/gtkradiant/

[16] Beide Engines und weiterer Source Code von idSoftare herunterzuladen auf dem Account des Unternehmens auf GitHub unter https://github.com/id-Software - geprüft am 08.04.2015

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 Middelware wie Scaleform[17] ein Cross Plattform UI Solution Tool[18] 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 schrieben den Source Code für die Anwendung, Engine, Netzwerkfunktionen, KI, und Tools. Diese Entwickler arbeiten hauptsächlich in einer IDE [19] wie Visual Studio (auf welches sich das zu dieser Arbeit konzeptionierte Tool bezieht) oder 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 Gamelogic [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 Textureditoren bis hin zu kompletten Welteditoren

[17] Autodesk Inc., 2014-2015.
[18] Ermöglicht das erstellen von 2D, 2.5D und 3D Ui Elementen. Wird z.B. von der Unreal Engine 4 verwendet.
[19] Integrated Developement Environment
[20] X-Code ist nur für MacOSX erhältlich
[21] Logik des Spiels, ermöglicht das interagieren etc. mit und in der Software

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.

2.2.4 Weitere für diese Arbeit nicht relevante

Computer-Grahics Engineer

Computer Graphics Engineers beschäftigen sich mit dem erstellen des Codes für die grafische Repräsentation der Engine.

Network Engineer

Artificial Intelligence Engineer

Sound Engineer

Fusee bietet während der erstellung dieser Arbeit nur ein rudimentäres Soundsystem an. Aus diesen Gründen geht diese Arbeit nicht weiter auf das Sound Engineering ein.

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 betroffenen internen Stakeholder am Arbeitsplatz und das beobachten der jeweils verrichteten Aufgaben sehr aufschlussreich. Durch diese Erkenntnisse könnten Probleme im Arbeitsablauf frühzeitig identifiziert und behoben und besonders wichtige Features rechtzeitig vor Beginn der Implementierung geplant werden.
Eine Bedürfnisabalyse und eine allgemeine Analyse der Aufgabengebiete der jeweiligen Entwickler sollte in das System Design bzw. das Requirements Engineering ebenfalls mit einfließen.

2.4 Ein Arbeitsprozess wird entwickelt

2.4.1 Game Authoring / Game Development

Was ist das, was beschreibt es, wieso ist es hier relevant? 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. 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.

2.4.2 Tool Development

Nach Wihlidals Auffasung (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. Es wird zuerst eine Planung des Tools durchgeführt. Hier wird der Umfang des Tools festgelegt und es werden Bedingungen zum Funktionsumfang formuliert.
Die zweite Phase beschreibt eine Bedarfsanalyse der von der neuen Software betroffenen Teammitglieder. 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.
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.

2.4.3 Planung

Es folgt die Planungsphase des Tools. Hier wird zuerst eine Requirementsanalyse [22] 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

[22] dt. Anforderungsmanagement

Zeitdruck für den Produktivbetrieb entwickelt wird, ein wichtiger Bestandteil der Projektplanung. Ein Tool, welches nicht den Anforderungen der Teammitglieder entpsricht kann nicht im Betrieb eingesetzt werden und verzögert im schlimmsten Fall die weitere Entwicklung des gesamten großen Softwareprojektes.

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

- Ässetsßollen 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

2.4.4 System Design / System Modeling

Anschließend an die Analyse folgt das System Modeling in welchem die Anforderungen des Programs zu einem Softwareprodukt modeliert werden. Oft bedient sich das Entwicklerteam hierbei Notationen wie UML [23] 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

[23] Unified Modeling Language

2.4.5 Abgleich des System Designs mit den Anforderungen

2.4.6 Implementierung

Die Implementierung ist der letzte Schritt

- Die Software wird auf Basis der FUSEE Engine und Cinema4D R16 erstellt.

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 clonen 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 "geparsedöder "konvertiert"werden
- Dateien sollen Version Control kompatibel bleiben (wenig bis keine Binarys)
- Das Projekt muss Strukturiert sein
- Eine Fusee Solution soll gehandelt werden können

3.4 Die Implementierung

3.4.1 Cinema 4D Plugin API und SDK

Uniplug

3.4.2 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

Welche Programme und Systeme sind beteiligt?

3.5.2 Generieren eines Fusee Projektes

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

3.5.5 Partial Classes in .NET

4 Ergebnisse und Erkentnisse

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

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.

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.quanticdream.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