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 7. April 2015

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

- TBD.

Kontakt:
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 Anforderung

1.2 Ziele

1.3 Fragestellung

1.4 Entwicklungsprozesse in Interaktiver 3D Software und Games

1.4.1 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.

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 Proble-

men.

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.

1.4.2 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 ihm einst mitteilte, “dass 50% der Entwickler Arbeit bei idSoftware in das Tool Development fliesse.” Vgl.

Bethke (2003, Seite 44). 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 [1] 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 [2]. Sony hat im Herbst 2014 den für Playstation 3 Spiele eigens entwickelten Welt Editor "Level Editor" als Open Source Software veröffentlicht [3]

1.5 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 jedoch 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.

1.5.1 Artists

Artists sind in einem Games Projekt für jegliche repräsentation der Spiellogik nach Außen zuständig. Hierbei handelt es sich um Modelle, Texturen, User Interfaces und weitere Oberflächen.

1.5.2 Designer

Designer (Gamedesigner) arbeiten eng mit Artists und Engineers zusammen. Sie schreiben oft Skripte und kleine Implementierungen oder verbessern Grafiken oder Spielfunktionen. Sie verwenden Assets aus der Designabteilung und fügen diese mit Skripten zusammen.

1.5.3 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 [4] wie Visual

[1] Interview mit dem SNSystems Team Freeman (2014)
[2] SNSystems http://www.snsystems.com/
[3] Wawro (2014)
[4] Integrated Developement Environment

Studio (auf welches sich das zu dieser Arbeit konzeptionierte Tool bezieht) oder XCode [5]. 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 [6]. 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 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.

1.5.4 Weitere für diese Arbeit nicht relevante

1.6 Stakeholderanalyse intern

Das bedeutet, diese Gruppen müssen bei der entwicklung eines Tools beachtet werden. Am besten wäre eine Rücksprache mit diesen Teammitgliedern und anschliessend eine analyse der Bedürfnisse für die jeweiligen Aufgabengebiete. Hierzu würde es sich empfehlen ein Gespräch zu führen und weiterhin einen praktischen Besuch am Arbeitsplatz zu vereinbaren.

1.7 Ein Arbeitsprozess wird entwickelt

1.7.1 Game Authoring / Game Development

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

1.7.2 Tool Development

1.7.3 Planung

- Eine Software soll es ermöglichen, dass Artists, Designer und Developer an ein und dem selben Projekt arbeiten können ohne die gewohnte Ar-

[5] X-Code ist nur für MacOSX erhältlich
[6] Logik des Spiels, ermöglicht das interagieren etc. mit und in der Software

beitsumgebung (3D-Modellierungssoftware, IDE) zu verlassen und etwas komplett neues (Level-Editor) zu erlernen.

- Die komplette Entwicklung und der Entwurf bedienen sich der FUSEE Engine.
- Ziel ist es in Cinema 4D ein FUSEE Projekt anzulegen, zu speichern und es zu öffnen (In Form einer Szene, Level, Welt.). Weiterhin müssen Ässetsïns Spiel integriert werden können die von Artists, Designern und Entwicklern bearbeitet werden können.
- Das Projekt sollte aus C4D heraus gebaut werden können.

1.7.4 Analyse

- Eine Stakeholderanalyse schafft klarheit, welche Parteien mit dem zu erstellenden Tool arbeiten müssen.
- Die Game Engines sollten Kategorisiert werden. Hierzu ist ein "Gitternetzäufzustellen.
- Verschiedene Game Engines müssen darauf untersucht werden, wie die Arbeitsschritte in diesen zu erledigen sind. Jede Untersuchung sollte dokumentiert werden.
- Es ist zuerst zu analysieren, welche Schritte für welche Art der Arbeit notwendig sind. Hierzu werden Usecases der verschiedenen Rollen und Aufgaben erstellt.

1.7.5 Design

- Aufgrund der erstellten Use Cases wird ein System Design erstellt.
- Das Systemdesing sollte Roundtrips vermeiden
- Hierzu können Partial Classes untersucht werden.
- 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

1.7.6 Implementierung

- 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.

Die Struktur von Game Assets

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

1.7.7 Warum eine Trennung von Code und Content?

2 Entwicklung eines Konzeptes

2.1 Use Cases der verschiedenen Entwickler

2.1.1 Was möchten Artists?

2.1.2 Was möchten Designer?

2.1.3 Was möchten Entwickler?

2.1.4 Projekt bezogen

Projekt anlegen

Projekt öffnen

Projekt speichern

Projekt bauen

In Projekt einsteigen

Projekt clonen etc.

2.1.5 Prozess bezogen

Gleichzeitig an Projekt arbeiten

Model Datei importieren

Gleichzeitig an einem Objekt arbeiten

2.2 Aktuelle Engines und deren Arbeitsprozesse

2.2.1 Prozesse in Game Engines und einem Framework

2.2.2 Unreal Engine 4

2.2.3 Unity 3D

2.2.4 idTech X

2.2.5 Weitere

2.3 Konzeptentwurf

2.3.1 Systemdesign für ein Plugin

2.3.2 Systemdesign für einen Project-Handler

Irgend etwas dass im Zentrum steht. Alles andere muss auf Level etc aufgeteilt werden und bis zum einzelnen Asset heruntergebrochen werden.

2.5 Das eigentliche Plugin

2.5.1 Visualisierung der Systemarchitektur

Welche Programme und Systeme sind beteiligt?

2.5.2 Generieren eines Fusee Projektes

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

2.5.4 XPresso Schaltungen - Programmieren ohne Programmieren

2.5.5 Partial Classes in .NET

3 Ergebnisse und Erkentnisse

3.1 Game Authoring Entwicklungsprozesse jetzt und in Zukunft

3.2 Wie weit ist die Implementierung fortgeschritten?

3.3 Welcher Mehrwert wurde erreicht?

3.4 Integration des Systems in den weiteren Projektverlauf von FUSEE

Anhang

Verzeichnis der Sourcecode Beispiele

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Abrash 1997] Abrash, Michael: *Graphics Programming Black Book*. The Coriolis Group Inc., 1997. – ISBN 978–1576101742

[Bethke 2003] Bethke, Erik: *Game Development and Production*. Wordware Publishing Inc.; Auflage: Pap/Cdr (Februar 2003), 2003. – ISBN 978–1556229510

[Carter 2004] Carter, Ben: *The Game Asset Pipeline*. Delmar Cengage Learning, 2004. – ISBN 1–58450–342–4

[CCP Games 2009] CCP Games: *Eve Online Fanfest 2009 - Scrum and Agile development processes*. 2009. – Available online - last checked 07.04.2015 https://www.youtube.com/watch?v=GqsReCZD4hc

[Chandler 2006] Chandler, Heather: *The Game Production Handbook*. Thomson Delmar Learning, 2006. – ISBN 1–58450–416–1

[Freeman 2014] Freeman, Will: *Kingdom Come: Deliverance Video Update 9*. 2014. – Available online - last checked 07.04.2015 http://www.develop-online.net/tools-and-tech/inside-the-system-sony-s-internal-console-tools-team/0191319

[Gamma u. a. 2014] Gamma, E ; Johnson, R ; Helm, R ; Vlissides, J: *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Deutschland, 2014. – ISBN 0–201–63361–2

[Gergory 2009] Gergory, Jason: *Game Engine Architecture*. Taylor & Francis Ltd., 2009. – ISBN 978–1568814131

[Lengyel 2012] Lengyel, Eric: *Mathematics for 3D Game Programming and Computer Graphics*. Course Technology, Cengage Learning, 2012. – ISBN 1–4354–5886–9

[Martin Klekner 2014] Martin Klekner, Daniel V. Votja Nedved N. Votja Nedved: *Kingdom Come: Deliverance Video Update 9*. 2014. – Available online - last checked 07.04.2015 https://www.youtube.com/watch?v=LfklQR-36-8

[Schmitz 2013] Schmitz, Christopher: *The art of waiting*. 2013. – Available online - last checked 07.04.2015 http://www.warhorsestudios.cz/index.php?page=blog&entry=blog_031

[Schmitz 2014] Schmitz, Christopher: Projektmanagement mit Scrum. (2014). – Available online - last checked 07.04.2015 http://www.makinggames.biz/features/projektmanagement-mit-scrum,2534.html

[Wawro 2014] Wawro, Alex: *Sony releases level editor that's open source and engine-agnostic*. 2014. – Available online - last checked 07.04.2015 http://www.gamasutra.com/view/news/224682/Sony_releases_level_editor_thats_open_source_and_engineagnostic.php

[Wihlidal 2006] Wihlidal, Graham: *Game Engine Toolset Development*. Thomson Course Technologies, 2006. – ISBN 1–59200–963–8

UML Diagramme