Master-Thesis Disputation Dominik Steffen

BETREUER:

PROF. C. MÜLLER UND PROF. DR. W. TAUBE

Gliederung

- Überblick
  - Vorstellung des Themas
  - Ziele der Arbeit
  - Herangehensweise und Methodik
  - Verwendete Software
  - Problematik und Ergebnisse
- Beantwortung der eingereichten Fragen
  - Block A (Prof. Dr. W. Taube)
  - Block B (Prof. C. Müller)
- Fragen und Diskussion aus dem Auditorium

Vorstellung des Themas

- **„Analyse des Game Authoring Tool Entwicklungsprozesses und Konzeption eines Authoring Tool Frameworks“.**
  - *„A tool is a software application used in either the construction or modification of game-related content, where the content can be virtually anything that makes up a game.“*
    - G. Wihlidal „Game Engine Toolset Development“ S. 3.

Ziele der Arbeit

- Was war das Ziel?
  - Das Untersuchen des Entwicklungsprozesses und bereits erhältlicher Tools.
  - Ist getrennte Software für Programmierer und Designer für Fusee umsetzbar?
  - Daraus folgend → Konzeption eines Editor Authoring Tools und Umsetzung eines Prototyps soweit es möglich ist.
  - Verwendet werden sollten die Fusee Engine und Cinema 4D.

Herangehensweise

- Betrachtung von Referenzmodellen im Entwicklungsprozess
- Untersuchung beliebter und erfolgreicher Editoren und Frameworks
- Konzeption eines Frameworks
  - Konzeption mit Hilfe von Methoden des Software Engineering im Bezug auf die Fusee Engine
- Implementierung der Konzeption
  - Als prototypischen Ansatz

Verwendete Tools und Software

- Cinema 4D
  - Entwickler: Maxon
  - Proprietäre 3D Modeling Software
  - Wird verwendet als: Welteditor und Benutzerschnittstelle
- Fusee
  - Furtwangen University Simulation and Entertainment Engine
  - Open Source 3D Engine
  - Entwickler:
    - Prof. C. Müller
    - Projektgruppen der Studierenden der HFU.
    - Weiterentwickelt in Abschlussarbeiten.
- Das Projekt Uniplug
  - Ermöglicht das Entwickeln von C# Plugins für Cinema 4D.
  - In die Codebase von Fusee integriert

Problematik

- Komplexität der proprietären Cinema 4D API
  - Erschwerte das Debuggen
  - Oft nötige Anpassungen durch API Updates
  - Uniplug Projekt musste verwaltet und umstrukturiert werden
- Probleme mit den managed Code Bestandteilen des Uniplug Projekts
  - GUI Problematik in Cinema 4D
  - Plugin Typen
  - Datentypen in der C++ API
    - Rückgabewerte

Ergebnisse der Prozessanalyse

- Tool Development ist ein wichtiger Produktionsschritt in der Spieleentwicklung.
  - Sony und das ATF Framework (Tool Studio).
  - Deck 13 (Making Games Magazin 03/15)
- Es ist sinnvoll den Prozess für das Produktionsteam transparent zu gestalten und das Team einzubinden.
  - Am Ende einfach zu nutzendes Produkt
- Geschickte Stakeholder Analysen und Managementprozesse:
  - Erleichtern die Einhaltung der Anforderungen.
  - Erhöhen die Wiederverwendbarkeit und Qualität des Tools.

Ergebnisse der Prozessanalyse

- *„It is very important to ask the right questions to your stakeholders [...] a lot of design and development time is wasted because of incorrect user requirements. Getting them right from the start will help alleviate this problem.“* [S. 28]
- *„They (Stakeholders, Anmerkung des Autors) are the users who are most affected by the introduction of a tool and they ultimatelycontribute to the design and goals. […] defined as anyone who stands to gain or lose from the succes or failure of an application [...].“* [S. 4-5]
- Wihlidal, “Game Engine Toolset Development”, 2006

Ergebnisse der Prozessanalyse

- Der Tool Entwicklungsprozess sollte mit der nötigen Sorgfalt in den Produktionsprozess eines Projektes integriert sein.
  - Produktion steht und fällt mit guten Tools.
  - Mitarbeiter als Nutzer entscheiden über gelingen des Projektes und die Nutzung der Tools.
- Managementprozesse spielen eine große Rolle.
  - Tool Entwickler stehen oft unter Zeitdruck.
  - Agile Modelle sind zu bevorzugen, da beim Tool Development oft schnelle Entscheidungen getroffen werden und Konzepte schnell umgesetzt werden müssen.

Ergebnisse der Prozessanalyse

- Getrennte Tools können funktionieren
  - Wenn das Team sich darauf einlässt
  - Das Projekt (der Zeitplan) es ermöglicht
- Erweiterung bestehender Tools ist sinnvoll
  - Es verkürzt den Entwicklungszeitraum
  - Es erleichtert den Entwicklern den Einstieg durch bereits bekannte Workflow Elemente.
- Verwendung von Best Practice Beispielen aus bereits erfolgreicher Software ist sinnvoll
  - Konzepte zu Szenengraphen werden für die Konzeption betrachtet.
  - Versionskontrollsysteme empfohlen.
  - → Erhöht Akzeptanz der Nutzer

Ergebnisse der Konzeption und Implementierung

- Konzeption der Architektur des Frameworks wurde erstellt
  - Setzt das Konzept der nach Entwicklern getrennten Authoring Tools um.
- Basisfunktonalität wurde konzipiert und implementiert.
- Nach Analyse der untersuchten Best Practice Software stellte sich heraus:
  - Funktionalität von Game Engine Editoren ist der Funktionalität von Modeling Editoren wie Cinema 4D oder IDEs wie Visual Studio sehr ähnlich.
  - Somit sind diese Tools meist nur Aggregatoren verschiedener Nutzerkonzepte.

Ergebnisse der Konzeption und Implementierung

- Möglichkeit zur Erweiterung über das Plugin System von Cinema 4D ist berücksichtigt → Projekt Uniplug.
- Modeling Tool unabhängiges System Design
  - Kann durch Implementierung des Plugin Interfaces angepasst werden.
  - Auch IDE unabhängig so lange die Software mit .sln Projekten umgehen kann.

Problematik in der Implementierung

- Probleme bei der Umsetzung des Prototypen zum Ende der Arbeit
  - Durch die Komplexität der Cinema 4D API
  - Die Updatezyklen der API
    - Sehr eng
    - Oft neue Funktionen oder Parameter
  - Uniplug Funktionalität erweitern
    - GUI etc.
  - Das komplexe „wrapping“ der API mit Swig.
    - Simplified Wrapper and Interface Generator
    - GUI Komponenten
    - Plugin Typen

Fragen Block A *(Prof. Dr. W. Taube)*

- **„Welche Objekte werden von welchen Benutzergruppen (während eines Projektes) wie bearbeitet und wie hängen diese Objekte untereinander zusammen? Und wie entwickeln sich die Abhängigkeiten im Zeitverlauf – also bei Änderungen von Objekten?“**

Fragen Block A *(Prof. Dr. W. Taube)*

- Alle Antworten auf Fusee bezogen
- Programmierer bearbeiten C# Code Dateien. (Evtl. JS und Projekt-Dateien)
  - Mit Visual Studio
  - Mono Develop / Xamarin Studio
- Designer und Artists erstellen Assets (Texturen, Modelle, Sound, etc.)
  - In Cinema 4D
  - In Photoshop oder ähnlicher Software

Fragen Block A *(Prof. Dr. W. Taube)*

- Der Zusammenhang:
  - Ergibt sich daraus, dass Programmierer Assets verwenden, um sie in die Spiellogik zu integrieren.
  - Ergibt sich daraus, dass Designer Code Dateien verwenden um sie in einer Szene zu platzieren, oder ihre Parameter verändern (Geschwindigkeiten, Positionen, etc.). Sie erstellen Welten aus einer Sammlung von Assets und setzen hierzu Skripte und Grafische Assets ein.

Fragen Block A *(Prof. Dr. W. Taube)*

- Während der Entwicklung wachsen diese Objekte meist näher zusammen.
- Framework kümmert sich um Verwaltung
- Theoretisch soll eine möglichst lose Kopplung der Elemente erreicht werden.
  - Dies unterstützt den Entwicklungsprozess und vereinfacht das Erstellen der einzelnen Objekte.

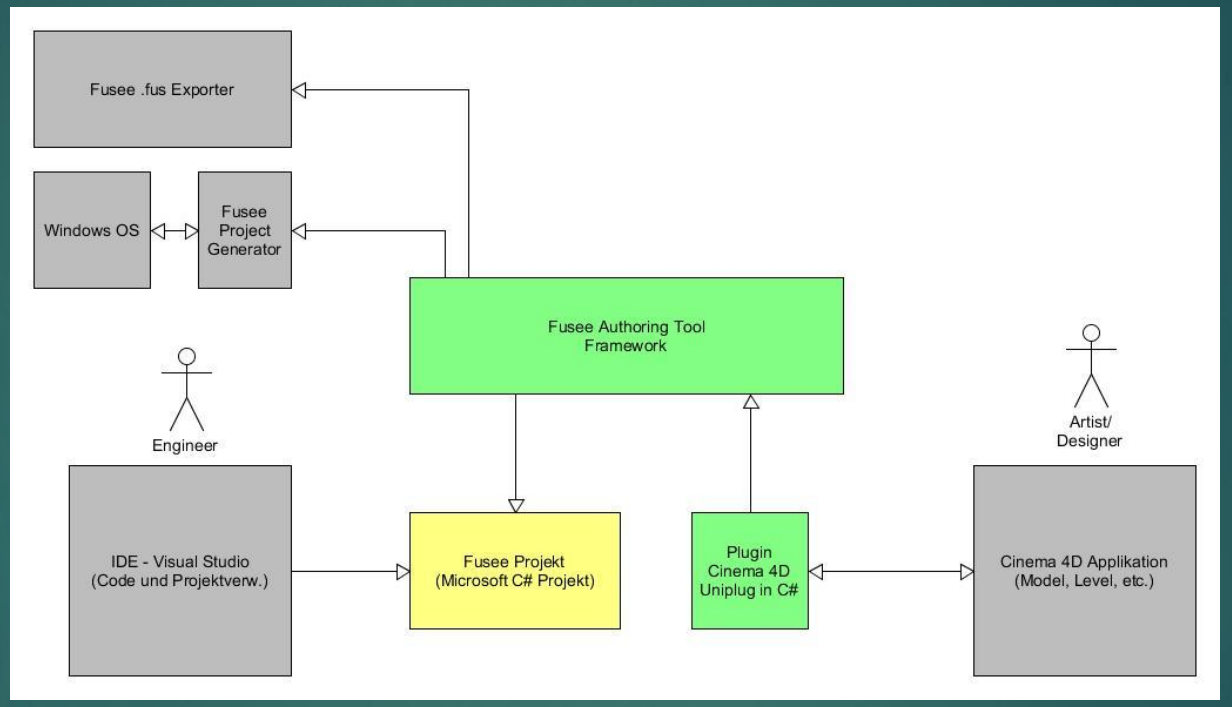
Fragen Block A *(Prof. Dr. W. Taube)*

- **F2 - „Wie wirken sich denn die unterschiedlichen Sichtweisen der verschiedenen Gruppen auf das konkrete Design des FuseeAT aus?“**
  - Das nach Entwicklergruppen getrennte Design zeigt sich hauptsächlich durch die Konzeption des FuseeAT als Framework welches als Vermittler in der Mitte zwischen den jeweiligen Tools sitzt.
  - Es übernimmt Verwaltungsaufgaben
    - Im Projektbereich
    - Im Dateibereich

Fragen Block A *(Prof. Dr. W. Taube)*

- Programmierer arbeiten weiterhin in Visual Studio
- Artists arbeiten in ihrer Modeling Software (Cinema 4D)
  - Das erlaubt vereinfachte Repräsentation von Projektoptionen wie:
    - Neues Level erstellen
    - Assets platzieren
    - Code Dateien anlegen
- Designer nutzen bevorzugt die grafische Oberfläche von Cinema 4D, können aber optional in den Code eintauchen.

Fragen Block A *(Prof. Dr. W. Taube)*



Fragen Block A *(Prof. Dr. W. Taube)*

- Gesteuert wird die Verwaltung der Projekte von FuseeAT aus.
- Die Plugins für Cinema 4D erleichtern und ermöglichen den Artists und Designern die Arbeit an Welten und Leveln.
  - Fusee verfügt aktuell über keinen Welt Editor
  - Level werden von Hand im Code gebaut oder als Geometrie importiert.
- Ein Teil des Designs ist auch der Anforderung an ein Editor unabhängiges Tool geschuldet
  - → der Extra „Layer“ „Plugin Cinema 4D Uniplug in C#“.
  - Dieser enthält speziellen Cinema 4D Code aber keine FuseeAT Funktionalität.
  - Er ruft diese nur auf. So kann das Tool an andere Editoren angepasst werden.

Fragen Block A *(Prof. Dr. W. Taube)*

- **F3 – „Existieren zentrale Probleme bei der Versionierung? Ist das bei der FuseeAT der Fall und wie werden die Probleme gelöst? Bei der Versionierung von Assets verweisen Sie auf Git (S. 71) – aber das bedeutet ja wieder eine völlig andere Oberfläche mit anderen, gar nicht so einfach zu verstehenden Konzepten.“**

Fragen Block A *(Prof. Dr. W. Taube)*

- Bei Fusee vorerst keine verlustbehaftete Konvertierung.
  - Game Asset Packaging oder Komprimierung ist aktuell nicht implementiert.
- Verlustloser Export wird durch Cinema 4D behandelt.
  - Export als .obj oder .fbx oder .fus
- Die Problematik der Versionierung ergibt sich also nur durch die allgemeine Entwicklung an sich.
  - Es werden Konzepte zur Lösung des Problems eingesetzt die erprobt sind
    - Versionskontrolle mit Hilfe eines erprobten Systems.

Fragen Block A *(Prof. Dr. W. Taube)*

- Wie werden die Assets verwaltet?
  - Sie werden im Nativen Format vorgehalten.
    - Für spätere Änderungen und Ergänzungen.
  - Für den Einsatz im Projekt exportiert und im Projektverzeichnis verortet.
- Für die Versionskontrolle (Arbeit Seite 71, Abschnitt 4.7.2) wird:
  - Git empfohlen.
  - Fusee Projekte werden aktuell mit GIT verwaltet.
  - Es macht wenig Sinn eine neue Versionskontrollsoftware zu entwickeln.
- Für nicht lesbare Dateiformate können andere Tools eingesetzt werden.
  - Alienbrain
  - Shotgun
  - Beide spezialisiert auf Asset Versionierung.
  - Aber losgelöst vom tatsächlichen Software Projekt.

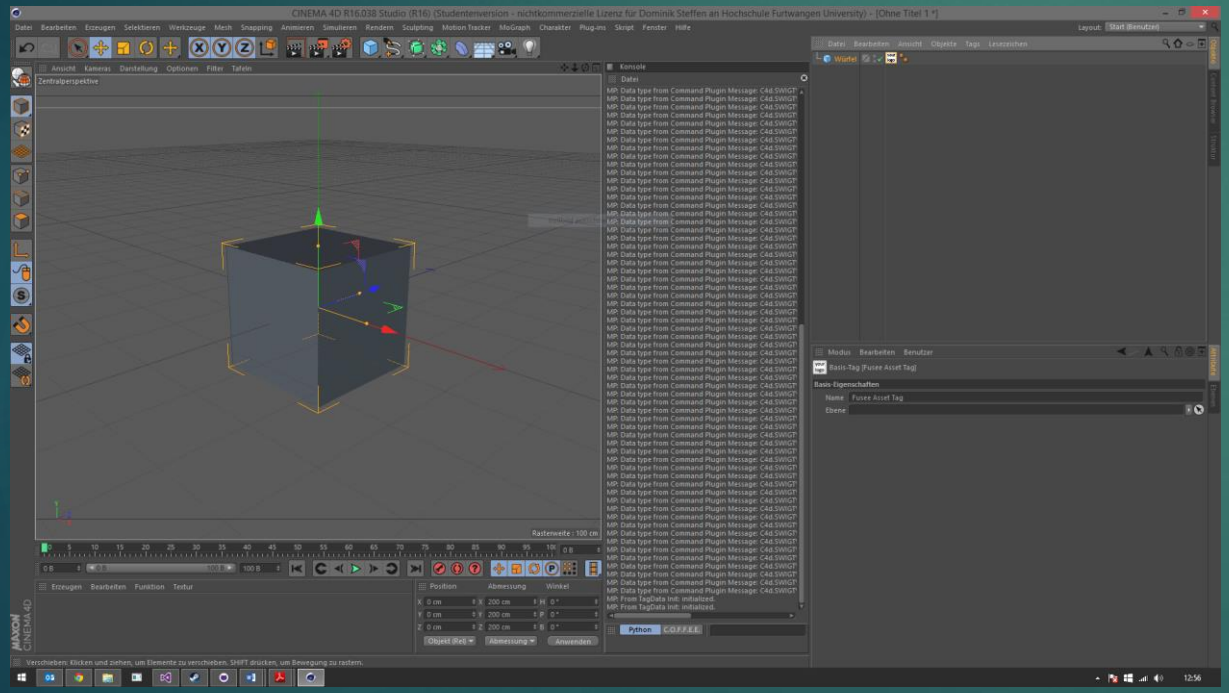
Fragen Block A *(Prof. Dr. W. Taube)*

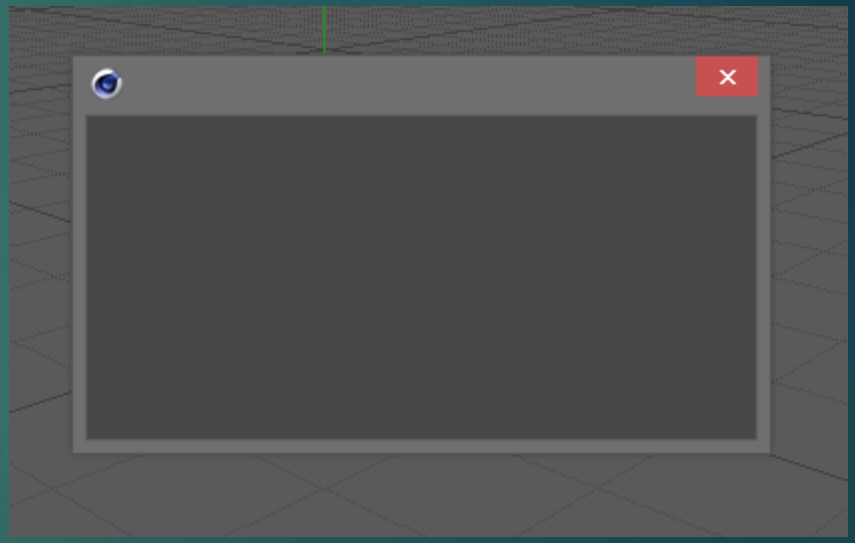
- Das Konzept Versionskontrolle
  - Sicherlich kein besonders einfach zu verstehendes Konzept …
  - … aber nötig um ein Projekt in der Games Branche zu stemmen.
- Die anderen analysierten Engines (Unity und Unreal)
  - Bieten Versionskontrolle über Schnittstellen
  - Bieten Versionskontrolle über:
    - Lizenzpflichtige Team Server Software (integriert, aber teuer)
    - Open Source Software (Git)
- Versionskontrolle muss vom Team verstanden werden.
  - Einsatz in vielen Projekten
  - Unreal Tournament als Open Source Community Entwicklung

Fragen Block A *(Prof. Dr. W. Taube)*

- **F4 – „Die Unterstützung von Arbeitsprozessen macht sich doch sicherlich auch in Screen-Designs für die unterschiedlichen beteiligten Entwicklergruppen bemerkbar?“**
  - „Jein“.
  - Cinema 4D Plugins ermöglichen kein freies Anpassen des Interfaces.
  - Hinzufügen von Caption Text und Buttons gestattet.
  - GUI Funktionalität Problematisch in Uniplug.
  - FuseeAT unterstützt aktuell nur Funktionalität im Hintergrund des Editors und bietet nur wenig GUI Elemente.

Fragen Block A *(Prof. Dr. W. Taube)*





Fragen Block A *(Prof. Dr. W. Taube)*

- **F5 – „Wo bleiben in Fusee (Umwandlung des C4D in den Fusee Graphen) die zusätzlichen C4D-Daten? Vermutlich werden sie als Components gespeichert, oder?**
- Die zusätzlichen Daten können/werden wie Sie angemerkt hatten als Components gespeichert falls sie für die Arbeit mit Fusee/FuseeAT benötigt werden.
  - Hierzu zählen folgende Daten:
    - Transformationen (Positionsdaten etc.)
    - Texturen / Materialien
    - Mesh Objekte (entspricht Geometriedaten)
  - Cinema 4D speichert aufgrund seiner Natur als Modeling- und Animations-Editor noch weitere Daten welche aktuell nicht in Fusee verwendet werden.

Fragen Block A *(Prof. Dr. W. Taube)*

- **F5.5 – „Was ist denn der große Vorteil vom Fusee-Szenegraphen? Es muss ja einen Sinn haben, die Nodes nur als Container zu fassen.“**
  - Der Fusee Szenengraph ist vom Fusee Team (in diesem Fall Herrn Prof. C. Müller) entworfen worden um die Traversierung von Szenenobjekten zu vereinfachen und ein auf Components basierendes System mit Hierarchien aufzubauen.
  - Die Daten als Components zu hinterlegen ergibt folgende Vorteile:
    - Verschiedene Components können zur Laufzeit unproblematisch ausgetauscht werden
      - Model wechseln
      - Material wechseln
      - Transformationskomponenten bearbeiten

Fragen Block A *(Prof. Dr. W. Taube)*

- Es muss hierzu kein neues Szenenobjekt instanziiert werden.
- Es genügt lediglich eine Referenz auf das Component-Objekt zu verändern um die neuen Eigenschaften an das Szenenobjekt zuzuweisen.
- Das System ist zukünftig erweiterbar für neue Component Typen
- Das System ist vielen Studierenden aus Unity etc. bekannt.

Fragen Block A *(Prof. Dr. W. Taube)*

- **F6 - "Der Szenegraph der UE4 gleicht dem Fusee und Unity 3D Szenengraphen stark. …. und darum kann das System von Cinema 4D übernommen werden ohne voraussichtlich dem Arbeitsfluß der Designer und Artists zu schaden." - Das ist ja die zentrale Fragestellung der Arbeit, die sie hier einfach als gesetzt voraussetzen und dies bedarf dann doch wohl einer Erläuterung.**

Fragen Block A *(Prof. Dr. W. Taube)*

- Der Abschnitt betrachtet die Verwendung des Szenengraphen bzw. der Repräsentation dieses Systems in der Fusee Engine.
- Orientiert sich an Best Practice Beispielen der analysierten Engines
  - Unity
  - Unreal Engine 4
- → Component basiertes Szenengraphen System in Fusee sollte den Workflow nicht beeinträchtigen.
  - Erfolgreiche Tools auf dem Markt setzen es so ein
  - Es ist ein bekanntes Konzept in der Spieleentwicklung
- Für Artists und Designer bezieht sich diese Aussage nur auf die Grafische Repräsentation des Konzeptes.

Fragen Block B *(Prof. C. Müller)*

- **F1: „Welche Möglichkeiten gibt es, von 3D-Modellierern erzeugte „Deliverables“ (3D-Objekte, 3D-Objekt-Bestandteile, -Eigenschaften) automatisiert als geeignete programmiersprachliche Konstrukte (Klassen/Objekte/Eigenschaften) zu repräsentieren, so dass Programmierer darauf Zugriff haben?“**

Fragen Block B *(Prof. C. Müller)*

- Trigger zu Beginn:
  - Von einem Hintergrundprozess oder GUI Aktion (Speichern eines Projektes) ausgehend.
- Endbedingungen: Die „Objektrelation“ muss für das FuseeAT identifizierbar bleiben.
  - Lesbares Format mit Informationen für das FuseeAT (XML in FuseeAT) enthält:
    - IDs
    - Pfade
    - Prüfsummen
- Wie kann also das Problem gelöst werden?

Fragen Block B *(Prof. C. Müller)*

Fragen Block B *(Prof. C. Müller)*

- „Deliverable“ Repräsentation als C# Klasse für Programmierer
  - Als Partial Class repräsentieren
    - Klasse aufgeteilt auf mehrere Files
    - Compiler baut daraus wieder eine Klasse
  - Das ermöglicht:
    - Sauberen Code.
    - Generierter Code getrennt von selbst erstelltem.
    - System des Frameworks greift nicht in Nutzercode ein.
    - Informationen zur Projektverwaltung stecken im automatisch generierten Teil. (bzw. Verweise auf XML Datencontainer)

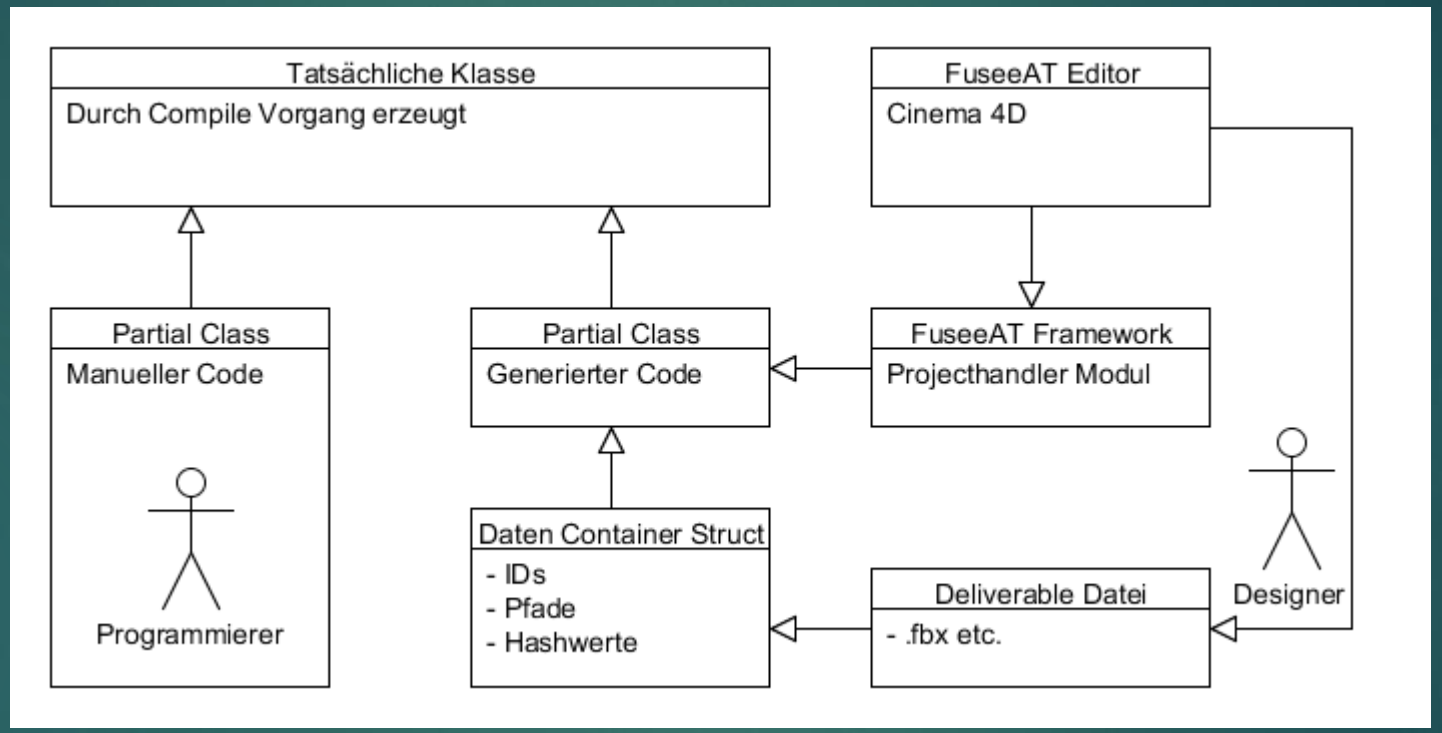
Fragen Block B *(Prof. C. Müller)*

- XML Datencontainer
  - Repräsentiert durch ein Struct
  - Serialisierbar über C# Mechanismen
  - Enthält Daten zur „Deliverable“ / „Asset“ Code Relation.
  - Wird benötigt wenn ein Projekt im Editor geöffnet wird um Relationen wieder herzustellen.

Fragen Block B *(Prof. C. Müller)*

- Welche Vorteile ergeben sich aus diesem Lösungsweg?
  - Programmierer kann Objekte im Code instanziieren.
  - Relationsdaten automatisch vom FuseeAT generiert und behandelt.
  - Partial Classes trennen generierten und manuellen Code.
- Wie sieht diese Repräsentation grafisch aus?
  - Das Schaubild verdeutlicht den Zusammenhang des Systems.

Fragen Block B *(Prof. C. Müller)*



Fragen Block B *(Prof. C. Müller)*

- **F2: - „Wie kann eine solche automatisiert generierte Brücke zwischen 3D-Modellierer und Programmierer funktionieren, wenn:“**
  - **A: „Programmierer und Modellierer nicht über das vollständige Toolset verfügen (Modellierer hat nur C4D, Programmierer hat nur Visual Studio)?“**
  - B: „Programmierer und Modellierer abwechselnd/gleichzeitig iterativ an ihren jeweiligen Bestandteilen arbeiten?“

Fragen Block B *(Prof. C. Müller)*

- Das Toolset übernimmt das Eintragen von Projektpfaden im Visual Studio .sln Projekt.
  - Natürlich müssen beide Parteien über das Projekt verfügen.
  - Das Projekt muss vom Editor „geöffnet“ und „im Speicher“ gehalten werden.
  - Beim ersten generieren der Dateien.
  - Speichert Deliverables im Projektverzeichnis (Vorraussetzung)
- Somit sind die Dateien im Visual Studio Projekt ansprechbar
  - Hierzu zählen:
    - Deliverable Dateien (.obj, .fbx, .fus, etc.)
    - Generierte Partial Classes
- Bedingungen hierfür:
  - Tool kann die Pfade der Objekte abfragen und hat Zugriff auf das Dateisystem (Cinema 4D API ermöglicht das).

Fragen Block B *(Prof. C. Müller)*

- **F2: - „Wie kann eine solche automatisiert generierte Brücke zwischen 3D-Modellierer und Programmierer funktionieren, wenn:“**
  - A: „Programmierer und Modellierer nicht über das vollständige Toolset verfügen (Modellierer hat nur C4D, Programmierer hat nur Visual Studio)?“
  - **B: „Programmierer und Modellierer abwechselnd/gleichzeitig iterativ an ihren jeweiligen Bestandteilen arbeiten?“**

Fragen Block B *(Prof. C. Müller)*

- Voraussetzung für die Funktionalität ist
  - ein funktionierendes Versionskontrollsystem und eine geordnete Projektstruktur
- Es wird GIT als Versionskontrollsystem angenommen weil:
  - Git weit verbreitet ist.
  - Git bereits in der Fusee Entwicklung verwendet wird.
  - Die Konzepte der Versionskontrollsysteme sich für einfache Anwender wenig unterscheiden.
- Der Prozess des abwechselnden Arbeitens inkludiert
  - Hinzufügen von Dateien zum Versionskontrollsystem.
  - Übertragen der Dateien zu anderen Mitarbeitern.

Fragen Block B *(Prof. C. Müller)*

- Partial Classes verhindern Probleme durch Konflikte
  - Der Programmierer bearbeitet nur die manuelle Datei.
  - Das Tool greift nur auf den generierten Teil der Klasse zu.
- Problematik ergibt sich nur in der Handhabung der Projektdateien
  - Designer muss seine Dateien erst zur Verfügung stellen.
  - Tool muss korrekte Pfade übergeben.

Fragen Block B *(Prof. C. Müller)*

- Sollten sich Daten ändern so
  - Kann das FuseeAT diese im generierten Teil updaten
  - Kann der nächste Commit diese Dateien dem Programmierer zur Verfügung stellen.
- XML Datencontainer unterstützen Programmierer
  - Werden von FuseeAT aktualisiert.
  - Können zur Not vom Programmierer angepasst werden.
  - Updaten gleichzeitig indirekt den generierten Code beim nächsten verwenden des Editors.
  - Eventuell auch ein Prozess des FuseeAT möglich welcher sich auch in VS um die aktualisierung kümmert.

Abschluss

- Fragen und Diskussion des Auditoriums.

Verabschiedung und Dank

- Vielen Dank an die Betreuer der Arbeit für ihre Unterstützung.
- Vielen Dank an die Fakultät Digitale Medien für die Unterstützung und den Rückhalt in meiner Tätigkeit als Mitarbeiter während des Bearbeitungszeitraums.