Bearbeitung Fragen Prof. W. Taube

# Abschnitt 1

Welche Objekte werden von welchen Benutzergruppen wie bearbeitet und wie hängen diese Objekte untereinander zusammen? Und wie entwickeln sich die Abhängigkeiten im Zeitverlauf – also bei Änderungen von Objekten?

## Antwort:

* (Alle Antworten auf Fusee bezogen)
* Programmierer bearbeiten C# Code Dateien. (Evtl. JS und Projekt-Dateien)
* Designer und Artists erstellen Assets (Texturen, Modelle, Sound, etc.)
* 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.
* Während der Entwicklung wachsen diese Objekte meist näher zusammen.
  + Durch eine Mechanik welche sich im Framework um die Verwaltung der Assets kümmert, muss der jeweilige Entwickler nicht bei jeder neuen Version eines Assets das Objekt neu einbinden, sondern durch eine Referenz geschieht dies „automatisch“. Code Dateien sind für das Verhalten von Assets im Produkt verantwortlich.
* Theoretisch soll eine möglichst lose Kopplung der Elemente erreicht werden.
  + Dies unterstützt den Entwicklungsprozess und vereinfacht das Erstellen der einzelnen Objekte.

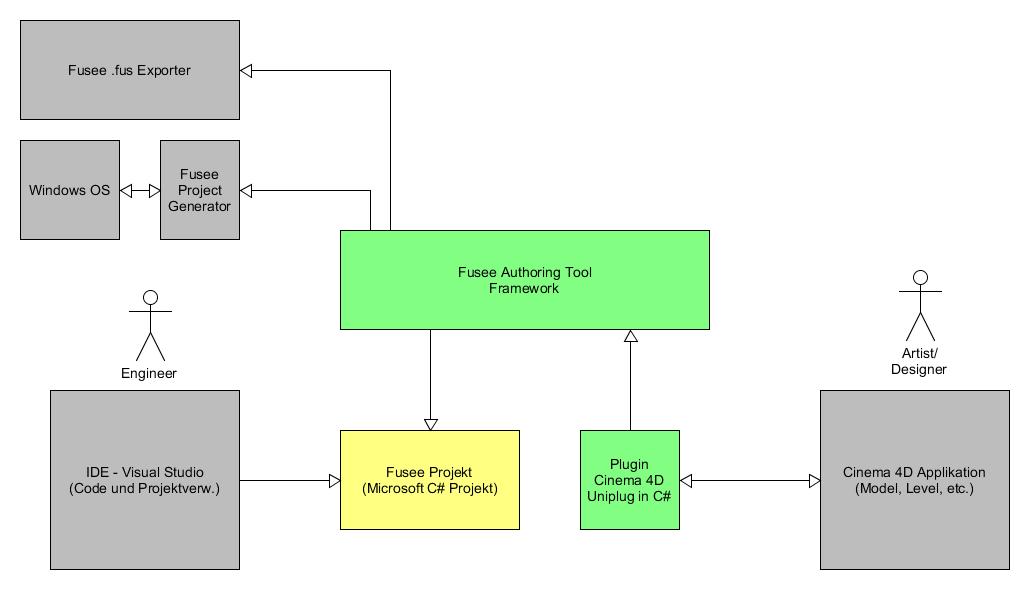
# Abschnitt 2

Wie wirken sich denn die unterschiedlichen Sichtweisen der verschiedenen Gruppen auf das konkrete Design des FuseeAT aus?

## Antwort:

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.

* Soll heißen:
  + Programmierer arbeiten weiterhin in Visual Studio
  + Artists arbeiten in ihrer Modeling Software (Cinema 4D)
    - 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.
* Gesteuert wird die Verwaltung der Projekte von FuseeAT
* Die Plugins für Cinema 4D erleichtern und ermöglichen den Artists und Designern die Arbeit an Welten und Leveln. (Fusee verfügt über keinen Welteditor. Level müssen von Hand im Code gebaut werden.)
* Ein Teil des Designs ist auch der Anforderung an ein Editor Unabhängiges Tool geschuldet, darum 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.



# Abschnitt 3

Zentrale Probleme bei Versionierung. Ist das auch 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.

## Antwort:

Bei Fusee (bzw. FuseeAT) findet vorerst keine verlustbehaftete Konvertierung von Objekten statt. Diese Art des „Packaging, Deployment, etc.“ ist noch nicht in Fusee integriert. Eine verlustlose Konvertierung findet im Fall des Exports von Cinema 4D Model Dateien nach .fus statt. Eine Problematik der Versionierung ergibt sich weniger durch FuseeAT im speziellen, als eher durch die Natur der Sache im Game Design an sich.

Unter „Wie werden Assets im Tool verwaltet? (Entscheidung für eine “Version

Control Software”)“ beschreibe ich, dass Assets während der Entwicklung in nativen Formaten vorgehalten werden, während sie für die Integration in das Softwareprojekt an sich in .fus oder ähnliches (je nach Wunsch der Entwickler auch nur .obj etc.) exportiert werden. Das ist eine gängige Praxis während der Entwicklung von Spielen.

Für die Versionsverwaltung sollten (laut Seite 71, 4.7.2) externe Tools eingesetzt werden (es würde wenig Sinn machen eine neue Funktion in FuseeAT zu integrieren, die ein bereits bekanntes Tool nachbildet). Für Code und lesbare Dateiformate kann Git eingesetzt werden (mit Grafischer Oberfläche oder als Bash, natürlich auch Subversion etc.) für Assets empfiehlt sich die Verwendung von Versionierungs-Software die mit Binären Asset Dateien umgehen kann wie Alienbrain oder Shotgun.

Die Konzepte sind sicherlich nicht die einfachsten im Game Design Prozess, aber Versionskontrolle muss für die Arbeit an der Fusee Engine Aktuell über die genannten Tools erfolgen. Unity bietet hier eine eigene Server Applikation (hohe Lizenzkosten durch Unity Team Lizenz) die sich nur mit der Versionskontrolle des Projektes beschäftigt. Eine solche Applikation könnte in Zukunft als erweitertes Feature (auf Basis von Open Source Systemen wie z.B. Git) für FuseeAT geplant und integriert werden.

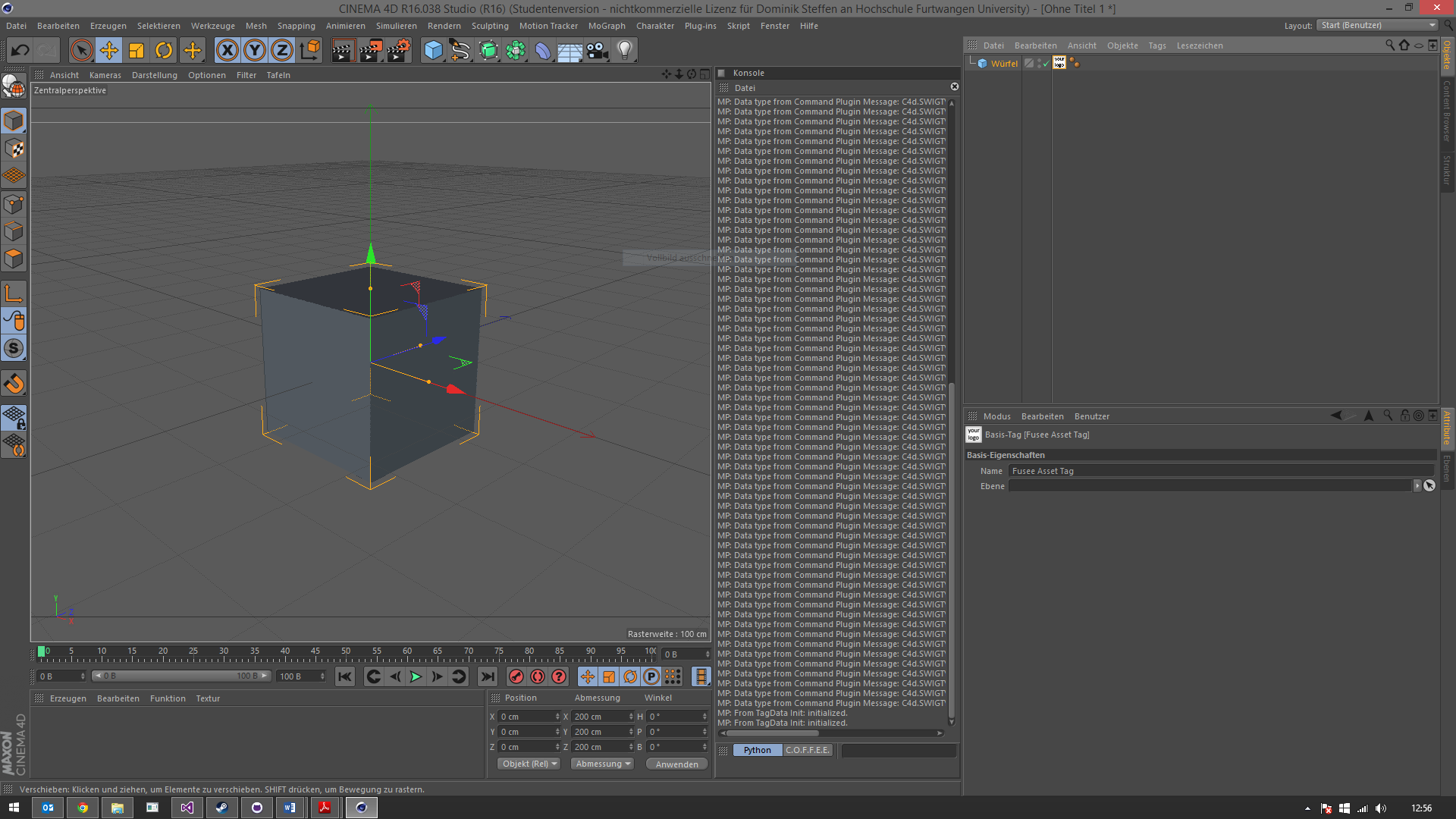
# Abschnitt 4

Die Unterstützung von Arbeitsprozessen macht sich doch sicherlich auch in Screen‐Designs für

die unterschiedlichen beteiligten Entwicklergruppen bemerkbar?

## Antwort:

„Leider“ nein. Cinema 4D Plugins ermöglichen aktuell (von der API her definiert) keine große Anpassung des Interfaces. Es gibt lediglich kleine Areale im Interface welche sich durch das Hinzufügen von Buttons und Caption Text auszeichnen. Es wird für die GUI Funktionalität von FuseeAT nur auf diese Möglichkeiten gesetzt. Leider besteht hier auch nicht die Möglichkeit diese Buttons und Menüs beliebig zu platzieren.

FuseeAT an sich unterstützt lediglich Funktionalität im Hintergrund welche sich um das Zusammenspiel der verschiedenen Tools mit dem einen Projekt kümmert.

# Abschnitt 5

Wo bleiben da (Umwandlung des C4D in den Fusee Graphen) die zusätzlichen C4D‐Daten? Vermutlich werden sie als Components gespeichert, oder?

Die eigentliche Frage ist aber: Was ist denn der große Vorteil vom Fusee‐Szenegraphen? Es muss ja einen Sinn haben, die Nodes nur als Container zu fassen.

## Antwort:

Die züstzlichen Daten können/werden wie Sie angemerkt hatten als Components gespeichert falls sie für die Arbeit mit Fusee/FuseeAT benötigt werden.

Hierzu zählen vorerst folgende Daten:

* Transformation (Positionsdaten)
* Texturen / Materialien
* Mesh Objekte

Cinema 4D speichert aufgrund seiner Natur als Modeling- und Animations-Editor noch weitere Daten welche aktuell nicht in Fusee verwendet werden.

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 aufzubauen.

Die verschiedenen Daten als Components zu fassen ergibt erst einmal folgenden Vorteil:

* Verschiedene Components können zur Laufzeit unproblematisch ausgetauscht werden
  + Model wechseln
  + Material wechseln
  + Transformationskomponenten bearbeiten
* 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.
  + Die Szene ist in einer Baumstruktur geordnet, das bedeutet Traversierungen fallen leichter.
* Das System ist erweiterbar für neue Component Typen.

# Abschnitt 6

Das ist ja die zentrale Fragestellung der Arbeit, die sie hier einfach als gesetzt voraussetzen und dies bedarf dann doch wohl einer Erläuterung.

*Zitat Arbeit (Seite 54, oben): „Der Szenengraph der UE4 gleicht dem Fusee und Unity 3D Szenengraphen stark. Es handelt sich bei dieser Ausgestaltung um eine Konvention im Sektor der 3D Engines und darum kann das System von Cinema 4D übernommen werden ohne voraussichtlich dem Arbeitsfluss der Designer und Artists zu schaden.“*

## Antwort:

Es geht hier um die Verwendung des Szenengraphen bzw. der Repräsentation dieses Systems in der Fusee Engine. Da in vielen andern Game Engines (inklusive Unity 3D und UE4 – zwei sehr beliebte Game Engines deren Workflows und System Strukturen sich oft als best practice Beispiele in der Spiele Entwicklung herausstellen) der Szenengraph auf genau diese Art und Weise implementiert ist bzw. verwendet wird, sollte hier davon ausgegangen werden können, dass ein Component basiertes System in Fusee den Workflow (mit dem Szenengraphen, auch mit seiner grafischen Repräsentation in der GUI) in keiner Weise beeinträchtigt. Für Artists und Designer bezieht sich diese Aussage nur auf die grafische Repräsentation des Szenengraphen in Form der Darstellung als aufklappbare Listenrepräsentation.



**