# Disputationsfragen Prof. C. Müller

## Frage 1

* 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?

Aufgeschlüsselt:

Möglichkeiten, 3D Objekte, automatisiert, programmiersprachliche Konstrukte, Zugriff für Programmierer

## Frage 2

* 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?

Aufgeschlüsselt:

Wie funktioniert das obere, wenn:

Jeder nur sein eigenes Tool hat?

Wenn beide abwechselnd, bzw. gleichzeitig am gleichen Projekt arbeiten?

## Antworten:

### Frage 1

Ansatz 1:

Von einem Hintergrundprozess des Frameworks / Tools (beim Export oder Speichern des Projektes) als ein der GameEngine bekanntes binäres oder lesbares Format (z.B. Fusee .fus oder .fbx, oder per protocol buffers) exportieren. Hierbei eine Klasse / ein Konstrukt (Struct) mit einem spezifischen Namen / ID anlegen. Dieses Konstrukt enthält eine Verbindung zu den Daten (Pfade) des exportierten „Deliverable“ Objektes (Component basiertes System). Es enthält außerdem wenn nötig Prüfsummen (für Prüfung durch das Framework) zur Integrität des Objektes.

Der Basistyp der Konstrukt-Klasse könnte von einer Engine eigenen „Object“ (generischer Assettyp für Engine Objekte / 3D Objekte etc.) Klasse abgeleitet sein.

Das Objekt könnte als „Partial Class“ (mehrfach aufgeteilte Klassen zur Repräsentation eines Objekts, physikalisch separierte Dateien, nicht vom Compiler beachtet sondern eher für den Nutzer gedacht) repräsentiert werden. Somit würde der Entwickler ein „sauberes“ Objekt enthalten, während die nötigen Daten im zweiten Teil der „Partial Class“ stecken. Dieser wird jedoch nur vom Framework kontrolliert und nicht durch den Programmierer bearbeitet. Natürlich müsste das Framework prüfen, ob das Objekt so schon vorhanden ist. Hierzu gleich noch eine Anmerkung.

Für das Toolkit / Framework an sich benötigt es noch ein Informationsfile welches alle nötigen Pfade und wichtige Parameter sammelt, damit das Projekt im 3D Editor wieder rekonstruiert werden kann. Hierzu ist im System Design von FuseeAT das Konstrukt „struct ACRelationData“ vorgesehen. Es speichert die erwähnten Parameter über eine Asset Code Beziehung und wird anschließend als XML File serialisiert. Es wird benötigt, sollte ein Projekt im 3D Editor geöffnet werden um im Speichert die Beziehungen zuzuordnen.

Der Programmierer kann nun im Code Objekte des gewünschten Typs instanziieren bzw. sie in ein Szene Node System integrieren.

Der Vorteil ist, Informationen die von einem Code Generator stammen (in diesem Fall das Toolkit / Framework) verändern nicht den Code der während der Entwicklung vom Programmierer geschrieben wurde.

### Frage 2

#### A

Das Tool / Framework übernimmt das eintragen der Code Dateien in Visual Studio Projekt Struktur beim ersten generieren der Dateien. Somit kann auch der Programmierer direkt ohne Probleme auf die Dateien zugreifen. Hierzu gilt natürlich die Bedingung, dass das Tool / Framework auch die Pfade der Objekte abfragen kann (was aus Cinema 4D heraus laut API funktioniert). Eigentlich ist das Framework Design bereits unabhängig von der Art des Benutzers.

#### B

Durch die Verwendung von „Partial Classes“ ergibt sich kein Problem durch überschneidende Funktionalität. Der Programmierer kann seinen Teil der „Partial Class“ mit jeglichem gewünschten Inhalt füllen. Die einzige „Problematik“ ergibt sich in der Handhabung der „Projektdateien“ (XML Files mit Pfaden etc.). Sobald der Designer seine Arbeit das erste mal zur Verfügung gestellt hat (bzw. der automatische Exporter hat dies), gibt es hier keine Überschneidungen mehr. Sollten sich in einem weiteren Schritt Pfade oder Checksummen ändern, so kann das Toolkit / Framework diese während der Arbeit des Designers updaten und mit dem nächsten Übertragen über das Versionskontrollsystem würden die Informationen aktualisiert ohne den Code des Programmierers zu beeinflussen. Durch die Lesbarkeit der XML Pfad Dateien wird Flexibilität in der Projektverwaltung erreicht. Programmierer können im Falle eines   
„Merge-Conflicts“ eingreifen und Entscheidungen treffen. Hierzu sollte es allerdings nicht kommen.