Projekt 2 Dokumentation

Marlon Lückert, Julius Neudecker, Vincent Schnoor

26. Oktober 2020

1 Projektbeschreibung

Das Projekt *ARExhibition*, welches im Rahmen des Projektes *XRchitecture* entstanden ist macht das Ausstellen digitaler Modelle und Medien einfacher. Mit nur wenigen Klicks können 3D-Modelle, Bilder und Videos aus einem hierfür entwickelten Content-Management-System (CMS) in AR platziert und gespeichert werden. Die so entstandenen AR-Szenen können von Besuchern über einen Marker geladen und betrachtet werden.
Diese Ausstellungen können bspw. von Hochschulen und anderen Bildungsinstituten genutzt werden um Projekte der Studierenden und Schüler auszustellen.

1.1 Projektziel

Ziel des Projektes ist es, die Darstellung medialer Inhalte der Hochschule und anderer Lehrinstitute einfacher zu gestalten und eine Oberfläche für die Studenten der HAW zu schaffen um ihre im Semester erstellten 3D-Modelle, Bilder und Videos an einem zentralen Ort zu speichern und mit anderen Studierenden zu teilen.

1.2 Zielgruppen

Für unser aus zwei Teilen bestehendes Projekt gibt es zwei Zielgruppen.
Die erste Zielgruppe besteht aus den Studenten der HAW welche das Content-Management-System zum einfachen Teilen ihrer Arbeiten nutzen können. Die Projekte anderer Studiengänge und Fakultäten können so leichter eingesehen werden. Außerdem können die Arbeiten anderer Studenten heruntergeladen und für eigene Studienprojekte verwendet werden.

Bildungsinstitute wie die HAW Hamburg und Bildungsstätten wie Museen bilden die zweite Zielgruppe. Für bspw. die HAW Hamburg wird die Ausrichtung von Ausstellungen der Studentenprojekte vereinfacht, da Räume interaktiver und sinnvoller mit digitalen Medien gefüllt werden können. Wo vorher 3D-Objekte auf PC Bildschirmen betrachtet werden mussten, können diese mithilfe der App als Teil des Raumes betrachtet und mit ihnen interagiert werden. Museen und Ausstellungen können durch die App ihr Repertoire an Kunst und Objekten erweitern, da virtuelle Bilder und Objekte einfach ausgestellt werden können. Die Inhalte können genau wie vom Kurator gewollt platziert werden und erscheinen dem Besucher in der gewollten Position, Rotation und Größe.

2 Technische Umsetzung

Das Projekt besteht aus zwei gleich großen Bestandteilen: einem Content-Management-System, welches für die Bereitstellung des Inhalts der App und als Speichermedium studentischer Pro-

jekte dient und der App, welche die Platzierung virtueller Inhalte in einem Raum in AR ermöglicht.
Im Folgenden wird die technische Umsetzung des CMS und der App erläutert. Dabei wird auf bereits existierende ähnliche Produkte eingegangen und wie unser Projekt sich von diesen abgrenzt. Verschiedene Ansätze der Umsetzung werden diskutiert und miteinander verglichen und die Schwierigkeiten der Umsetzung werden erläutert.

2.1 Stand der Forschung

2.2 Content-Management-System

2.2.1 Aufbau der Datenbank

2.3 AR App

2.3.1 Verwendung von 3D-Modellen

Da die Platzierung von 3D-Modellen ein Hauptbestandteil der App ist, wurde ein großer Fokus auf das Dateiformat der 3D-Modelle gelegt. Die dynamische Verwendung der Modelle in der App sollte möglichst einfach und problemlos sein.
Aus diesem Grund wurden die verschiedenen von Unity unterstützten Dateiformate miteinander verglichen, um die im Dateiformat unterstützten Funktionalitäten und die Qualität der Darstellung in der App zu vergleichen.
Unity unterstützt nativ die folgenden Dateiformate:

- FBX
- DAE (Collada)
- DXF
- OBJ

Da die meisten Studenten Blender als 3D-Modelling Tool ihrer Wahl nutzen wurde ein Vergleich der in Blender möglichen Export-Dateiformate und der in Unity unterstützten Formate gemacht. Blender exportiert 3D-Modelle unter anderem als FBX und OBJ, welche ohne großen Aufwand in Unity importiert werden können.
Der Nachteil dieser Formate ist, dass die Materialien und Texturen als externe Dateien in einem getrennten Ordner exportiert werden. Für die direkte Verwendung in Unity kein Problem erschwert dies jedoch das Zwischenspeichern in unserem Content-Management-System. Weiterhin werden „Faces“, Flächen eines 3D-Modells, mit mehr als 5 „Vertices“, Eckpunkten, nicht unterstützt und somit in Unity nicht dargestellt. Bei nicht komplett sauber erstellten 3D-Modellen können so unschöne Lücken im Modell entstehen die unerwünscht sind.
Aus diesem Grund wurde das von der Khronos Group entwickelte und in Blender integrierte GLTF 2.0 Format auf seine Verwendbarkeit in Unity untersucht. GLTF speichert sämtliche Daten eines 3D-Modells, darunter auch Materialien und Animationen, in einer einzigen, auf dem JSON-Dateiformat basierenden Datei ab. Das Dateiformat ist extrem robust und speichert jegliche Geometrie eines 3D-Modells ab. Der Nachteil des GLTF-Formates ist allerdings, dass dieses erst ab der Blender Version 2.81 unterstützt und erst ab 2.83 richtig implementiert ist. Alte Blender Modelle müssen demnach auf eine höhere Blender-Version gebracht und dort exportiert werden. Weiterhin unterstützt Unity, wie in der Auflistung oben zu sehen, das Dateiformat GLTF 2.0 nicht nativ, weshalb zusätzliche Bibliotheken benötigt werden um die Dateien nutzen zu können.

Eine dieser und die in unserem Projekt verwendete Bibliothek ist die *GLTFUtility* Bibliothek des GitHub-Nutzers *Siccity*. *GLTFUtility* unterstützt den Import und Export von GLTF Dateien in Unity während der Laufzeit, was für unser Projekt von elementarer Bedeutung ist, da die 3D-Modelle beim Starten der App nicht bereits vorliegen, sondern während der Laufzeit der App dynamisch geladen und verwendet werden.

Der Vergleich der drei untersuchten Dateiformate hinsichtlich Qualität zeigte, dass die Darstellung von GLTF-Modellen gegenüber FBX- und OBJ-Modellen besser ist (siehe Abbildungen x, y und z). Die von Blender exportierten Materialien werden im GLTF-Format, durch die bessere Unterstützung diverser Material-Eigenschaften, besser dargestellt.

Aus diesen Gründen, der besseren Darstellung der 3D-Modelle in Unity, die robustere Geometrie-Darstellung und der Export in einer einzigen Datei, entschieden wir uns dazu GLTF als einziges Dateiformat für die Verwendung innerhalb der App zu verwenden.

2.3.2 Export-Anforderungen an 3D-Modelle

Die im Laufe des Projektes erstellten Anforderungen an den Export der 3D-Modelle hinsichtlich ihrer Material-Eigenschaften und Animationen beziehen sich ausschließlich auf den Export in der 3D-Modelling-Software Blender.

Die in Blender erstellten Materialien weisen mit zunehmender realitätsnähe eine steigende Komplexität auf. Aus diesem Grund mussten bestimmte Anforderungen an den Export von 3D-Modellen aus Blender gestellt werden, um die korrekte Darstellung in der von uns entwickelten App zu gewährleisten. Die folgenden Anforderungen gelten für den Export von 3D-Modellen aus Blender um die Darstellung in unserer App zu gewährleisten.

Meshes:
GLTF unterstützt den Export von jeglicher Geometrie des Meshes. Dabei ist die Anzahl der Vertices eines Faces irrelevant, da Quads und N-Gons beim Export automatisch in Triangles umgewandelt werden.
Kurven und andere „Nicht-Mesh“Daten werden nicht übernommen uns müssen vor dem Export in Meshes umgewandelt werden.

Materialien:
GLTF unterstützt die folgenden Material-Eigenschaften beim Export:

- Base Color
- Metallic
- Roughness
- Baked Ambient Occlusion
- Normal Map
- Emissive

Texturen werden als Base Color problemlos unterstützt. Bei Roughness und Metallic Texture-Maps müssen einige Einstellungen vor dem Export getroffen werden. Bei einer Textur erwartet GLTF die Metallic-Werte kodiert im B-Farbchannel, während die Roughness-Werte im G-Farbchannel kodiert sind. Das Node-Setup in Blender sollte demnach folgendermaßen aussehen:

Wurde das Node-Setup nicht angepasst wird versucht beim Export die relevanten Daten auszulesen, was mitunter zu längeren Exportzeiten führen kann.
Baked Ambient Occluion, Normal Maps und Emissive Materials werden problemlos unterstützt und müssen nicht weiter angepasst werden.

Animationen
Animationen im GLTF-Format zu exportieren ist nicht kompliziert. Folgende Animationstypen werden nativ beim Export unterstützt:

- Keyframes (Translation, Rotation, Scale)
- Shape Keys
- Armatures/Skinning

Animationen anderer Eigenschaften wie Licht oder Materialien werden ignoriert.
Wenn das 3D-Modell nur eine Animation hat gibt es bei der Verwendung in Unity keine Probleme. Probleme entstehen wenn das 3D-Modell aus mehreren Einzelteilen besteht die jeweils eigene Animationen haben, da die Animationen in Unity im Legacy Animation-System abgespielt werden müssen, welches nur eine Animation zur Zeit unterstützt. Aus diesem Grund müssen mehrere Animationen einem NLA Track hinzugefügt werden. Dieser dient quasi als Animations-Controller, welcher die einzelnen Animationen zu einer einzigen Animation zusammenfasst und die verschiedenen Objektteile bewegt.
Objekt-Constraints, wie „Copy Location“können ebenfalls exportiert werden, wenn diese vorher in Keyframes umgewandelt wurden.
Die letzte Anforderung beim Export von 3D-Modellen mit mehreren Einzelteilen und Animationen ist, dass es nur **ein** Parent-Objekt geben darf. Dies kann ein leeres Objekt sein, da dies für die Hierarchy in Unity und die Verwendung der Animationen relevant ist.

Wenn diese Anforderungen eingehalten werden können Blender-Modelle exportiert und in unserer App verwendet werden.

3 Wissenschaftliche Umsetzung

3.1 Forschungsfrage

„LidAR besser? “

3.2 User Research

3.3 Analyse der Ergebnisse

4 Schlussfolgerungen

5 Ausblick