

Konferenz-Software zur Präsentation von 3D-Modellen in VR

<Grafik>

Ein Projekt von:

Maarten Behn

Tim Jaeschke

Yesenia Möhring

24.01.2022

Kurzfassung

Inhaltsverzeichnis

Kurzfassung	2
Einleitung.....	3
Vorgehensweise, Materialien und Methode.....	4
Netzwerkverbindung	4
Network Package System.....	4
FileShare:	5
FBX Loader	5
VR	6
Ergebnisse	8
Ergebnisdiskussion.....	9
Quellen- und Literaturverzeichnis.....	10
Externe-Mittel	10
Abbildungsverzeichnis	10
Glossar	10
Literaturverzeichnis	10

Einleitung

Die Idee von unserem Projekt hatte Maarten durch seinen Nebenjob bei der Firma Arvico. Arvico erstellt unter anderem aus Bauplänen von geplanten Schiffen und ~~Dort werden 3D Modelle von geplanten Bau- und Schiffs- projekten~~ Bauprojekten 3D Modelle. Diese werden für Werbevideos und Kunden Präsentationen erstellt. Die Modelle werden in Werbevideos und Kundenpräsentationen visualisiert.

~~Dabei war immer eine offene Frage, wie man die 3D Modelle einfach einem Bei der Kundenvorstellung gab es allerdings das Problem das auf Bildern und potenziellen Kunden vorstellt. Bei der Vorstellung bieten sich VR Brillen an, da man Videos der 3D Modelle die Größe von zB einem Boot nicht richtig vermittelt werden mit einer VR Brille ein viel besseres Gefühl für Größen bekommt. Kann Eine VR Brille zu benutzen bietet sich daher an da mit ihr gerade Wegen der Corona Pandemie kann man sich oft nicht persönlich treffen können. großen Verhältnisse besser übermittelt werden können.~~

Deswegen haben ~~Wir uns~~ entschieden ein Konferenzprogramm zu entwickeln, in dem man 3D-Modelle online Vorstellen kann. Dabei loggen sich alle Nutzer in eine virtuelle Umgebung ein in der man in FirstPerson oder VR seinen Charakter bewegen kann. ~~mit einem Bildschirm~~

Es gibt ein Voicechat System mit dem man sich mit anderen Nutzern verständigen kann. Dieser Voicechat ist in Form von 3D-Audio realisiert. Das heißt, das man die Stimme einer Person aus der Richtung ~~der Person~~ hört. Außerdem werden Personen leiser, wenn ~~sie sich entfernen~~ ^{in der der Avatar der Person steht}. So kann man sich einfach in dynamische klein Gruppen aufteilen. ^{Ihr Avatar}

^{in der Konferenz}
~~Die dieser virtuelle Umgebung~~ ist es möglich 3D Dateien hochzuladen. Diese können ~~Um die Software zum gedachten Zweck verwenden zu können~~ verkleinert auf einem Tisch oder in Originalgröße zu Begehung angezeigt werden.

*~~Da so auch nicht das Problem entsteht das ein Kunde in einem anderen Land lebt und nicht zur Besichtigung vorbei kommen kann.~~

Vorgehensweise, Materialien und Methode

Unsere Software wurde mit der Unity Game Engine entwickelt. Dies hat den Grund, dass viele von uns genutzte Techniken wie z.B. 3D Audio, schon lange in Computerspielen verwendet werden.

Netzwerkverbindung

Als erstes haben wir das Netzwerksystem entwickelt.

Unser Netzwerk ~~System~~ basiert auf einem Host/Client ~~System~~ ^{Prinzip}. Die Person die die Konferenz startet ist der Host. Jede andere Person die der Konferenz beitritt verbindet sich daher als Client mit dem Host.
Damit wird sein Computer als Serverbenutzt auf dem sich als Client verbindet

Jeder Benutzer der Konferenz bekommt eine User-Id in Form eines bytes zugeordnet.

Der Host hat immer die Id 0.

Der Host verfügt über eine Implementation eines TCP sowie eines UDP Servers. Die Clients verbinden sich mit dem Host über TCP und wenn es möglich ist auch über UDP.

Hierfür haben wir die C# nativen TCPListener und UDPListener libaries verwendet. Diese erlauben es einen bytes zu einem anderen Nutzer zu schicken.

Network Package System

Für unser Programm haben wir Netzwerk Netzwerkkommunikationssystem, welches die rudimentären Funktionen der C# TCP und UDP libaries in ein einfaches nutzbares System verpackt.

Netzwerkprotokoll

Dafür haben ein Package-System entwickelt mit welchem wir Netzwerk Packages versenden können. Jedes Package ist mit einem Id als byte definiert. Diese Id gibt an welche Art von Package es ist. Jede verschiedene Netzwerkmessage, die wir senden hat eine eigene Individuelle Id. Z.B. hat so ein neue Postion eines Benutzers in der virtuellen Welt die Id 20 und die Nachricht das ein Benutzer die Konferenz verlassen hat die Id 4.

Jedes Package hat folgende Signatur:

Header:

- 1 byte (1 bool) ~~der~~ ^{das} Angibt, ob das Package asynchron behandelt werden soll.
- 4 bytes ~~(1 Int)~~ ^{die,} die die Länge des Inhalts angibt.
- 1 byte ~~der~~ ^{des} die Packet-Id angibt.
- 1 byte ~~der~~ ^{der} die User-Id des ~~Erstellers~~ ^{Packet} des Packets angibt.

Inhalt:

~~Packet~~ ^{ist von Packet}

Nun folgt eine Abfolge von Daten, die für jede ~~Packet Id~~ unterschiedlich sind. Die Menge ~~dieser~~ folgenden Bytes müssen mit der ~~Inhaltslänge~~ im Header übereinstimmen.

Das ~~Packet~~ System bietet eine Abstraktionsebene, die dafür sorgt, dass es für die anderen Bereiche des Programms egal ist ob der Benutzer ein Host oder Client ist.

Es gibt keinen zentralen Server in der Daten-Struktur aller anderen Systeme

FileShare:

Um 3D Modelle anzeigen zu können benötigt jeder Nutzer in der Konferenz eine Kopie der Ursprungs Datei. Um diese automatisch zu verteilen haben wir ein System namens FileShare gebaut. Dieses System funktioniert wie folgt:

Jeder Nutzer besitzt eine Liste aller Dateieninformationen, wie Name, Pfad und ob die Datei lokal vorhanden ist. Der Nutzer kann Dateien von seinem Gerät zu dieser Liste hinzufügen. Die Nutzer tauschen konstant ihre Listen untereinander aus, so weiß jeder Nutzer welche Dateien es alles im System gibt. Beim Austauschen wird auch vermerkt welcher Nutzer eine lokale Kopie der Datei besitzt.

Benötigt nun ein ~~Nutzer~~ ^{Teilnehmer} den Inhalt einer bestimmten Datei aus der Liste, stellt dieser eine Anfrage an einen der Nutzer, der die Datei lokal ~~vorhält~~ ^{gespeichert hat}. Nun starten die beiden Nutzer einen Datenaustausch ~~gestartet~~ ^{aufgeteilt und} ~~wird die Datei in kleinere Pakete~~ ^{Teilnehmer} zu dem zum anfragenden ~~Nutzer~~ gesendet.

FBX Loader

Um in ~~unserem~~ ^{unserem} Projekt 3D Modelle darstellen zu können, mussten wir uns erstmal auf ein Datei Format einigen. Wir sind zu dem Schluss gekommen das, das FBX Format, welches von Autodesk entwickelt wurde, die beste Wahl ist, da vielen Programme in diesem Format exportieren und es dafür entwickelt wurde mit vielen Programmen zu funktionieren [1]. ^{und}

Der erste Ansatz, den wir hatten, ist mit der internen Unity-Erweiterung „FBX-Export“ zu arbeiten, da Unity ~~FBX importen~~ bei Laufzeit nicht ~~direkt~~ unterstützt. Durch diese Erweiterung kann man FBX Dateien lesen und dann in für Unity verständliche ~~Daten~~ umwandeln. Dies hat auch ~~zu Anfang~~ gut funktionieren und lief im Unity Editor einwandfrei. Bis wir ~~dann~~ das Unity Project erstellen wollte zu einer fertigen Applikation, wo dann die Unity Erweiterung nicht mehr funktioniert hat, da diese nur für den Editor gedacht ist. Deswegen mussten wir uns nach neuen Wegen umschauen und haben ~~uns~~ dann im „Asset Store“ nach einer Lösung ~~umgeschaut~~. Da fast alle brauchbaren Lösungen Geld kosten haben wir ~~nach weiteren Lösung gesucht~~. Schlussendlich haben wir herausgefunden das man in Unity ein „Nativ Plugin“ schreiben kann. In diesem „Nativ Plugin“ kann man in der Programmiersprache C++ ~~einen~~ ~~die~~ Programmieren, wo es mehr „libraries“ gibt als in der Programmiersprache C# die von ~~nutzen~~ ~~für die es mehr~~ ~~für~~ Autodesk stellt so eine „library“ für das FBX Format breit [2]. Das war allerdings schwer mit dieser „library“ zu arbeiten, da ~~es~~ sehr umfangreich ist und ~~nicht sehr gut dokumentiert~~ ist. Zum Beispiel wollten wir texturen laden, aber ~~die gute Dokumentation bereitstellt~~ haben wir nicht richtig hinbekommen, da es kaum Beschreibung gab, wie ~~man~~ ~~Das laden von Texturen haben wir z.B. nicht hinbekommen da es kaum~~ ~~Texturen auslesen kann.~~ ~~Ressourcen zum auslesen dieser gibt.~~

Dadurch können ~~wir~~ über einem Umweg zu einer anderen Programmiersprache die FBX Dateien einlesen und an Unity weitergeben. Dies funktioniert einwandfrei hat, aber auch seine Nachteile, da das C++ Programm für verschiedene Plattform neu erstellt werden muss und man so ~~noch~~ mehr Aufwand hat, wenn man das Projekt für verschiedene Plattform ~~er~~ erstellen möchte.

VR

Damit man sich die 3D Model gut vorstellen kann, haben wir uns überlegt das man ~~um sich vor allem die Größe der 3D Modelle besser vorstellen kann~~ ~~haben sich die Modelle in VR angucken kann. Dadurch kann man die Größe und das Aussehen viel besser vorstellen.~~

Damit man VR in Unity umsetzen kann haben wir mit OpenXR [3] gearbeitet. Das ist ~~um VR in Unity einzusetzen~~ eine Schnittstelle, die von der „Khronos group“[4] entwickelt wurde und als einheitliche Schnittstelle dient ~~und es einfacher macht~~ eine Applikation für viel verschiedene VR-~~zwischen einem Programm und der VR Brille~~ Headsets zu erstellen. Des Weiteren haben wir mit dem SteamVR Plugin von Valve ~~Da kann das Programm nicht einzeln anpassen muss~~ gearbeitet, da wir zum Testen der VR Anwendung eine Valve Index VR Brille genutzt haben ~~ebenso konnten wir durch das Plugin noch weiter Funktionen der VR Brille nutzen wie zum Beispiel Finger Bewegungen erkennen. Mit diesem Plugin kann man~~ ~~das erkennen von~~ ~~Das~~ ~~macht es~~

auch leicht ~~ein~~ ^{zu} ein VR Spieler erstellen, da es schon sehr viele Beispiele gibt und vorgefertigter Code vorhanden sind.

Ergebnisse

Unserer Ergebniss ist eine Applikation, die mit der Unity Engine erstellt wurde, um eine Konferenz online stattfinden zu lassen, wo man auch 3D Modelle laden kann und mit einer VR Brille breitreten kann.

Es ist möglich einer Konferenz in VR beizutreten und 3D Modelle anzusehen und zu teilen

Um dies Realisieren zu können haben wir ein Network System aufgebaut, welches die Daten zwischen Clients austauscht wie zum Beispiel Position von Clients.

eine Position

Desweitern haben wir ein Voipchat System genutzt, damit man sich bei einer Konferenz unterhalten kann.

Damit man 3D Modelle laden kann haben wir ein „Nativ Plugin“ geschrieben welches mit dem FBX SDK 3D Modelle des Formates FBX laden kann.

?

Damit man in einer Konferenz beitreten kann haben wir auch ein UI erstellt, welches Einstellmöglichkeiten zur Netzwerk Verbindung und für Audio hat.

Ergebnisdiskussion

Schlussendlich haben wir es hinbekommen ein laufendes Programm zu schreiben. Wir haben ein gutes Server System gebaut, welches mit mehreren Clients funktioniert und übers Internet läuft.

Das FileShare System, um Datei austauschen zu können funktioniert auch gut, solange es im gleichen Netzwerk läuft. Sobald man Dateien übers Internet verschicken möchte, funktioniert ist nicht mehr so gut, da geschwindelt drastisch abnimmt.

Das Laden von FBX-Dateien hat auch nach ein paar Komplikationen funktioniert. Durch den Kompromiss mit der Nutzung eines „Nativ Plugins“ haben wir es hinbekommen. Jedoch erzeugen 3D-Modelle mit vielen Vertices lange Ladezeiten.

Den Voicechat haben wir auch schnell hinbekommen und hatten ein passables Ergebnis erreicht. Nur übers Internet konnten wir es nicht richtig testen, da sich die Server für Voicechat nicht verbinden konnten, aber im lokalen Netzwerk funktioniert alles einwandfrei.

Desweitern haben wir es geschafft das Laden von FBX Dateien ins Spiel auf den Clients zu synchronisieren. Dabei werden die 3D-Dateien automatisch vom bereitstellenden Nutzer heruntergeladen.

Das UI haben wir, für den Prototypen sehr simpel gehalten. In einer finalen Version des Programmes müssten wir noch mehr Zeit in das UI investieren.

Aus Zeitlichen Gründen konnten wir ein Feature, welches wir optional einbauen wollten, nicht mehr umsetzen. Die Applikation auf verschiedenen Betriebssystem laufen zu lassen wie zum Beispiel Android, IOS oder Linus. Zurzeit läuft das Programm nur auf Windows.

Quellen- und Literaturverzeichnis

Externe-Mittel

- Unity Game Engine 2021.1.13
- Unity Shader Graph
- Unity Cinemachine
- OpenXR
- SteamVR
- FBX SDK
- UniVoice (<https://github.com/adrenak/univoice>)
- Runtime File Browser (<https://assetstore.unity.com/packages/tools/gui/runtime-file-browser-113006>)
- HQ Acoustic system (<https://assetstore.unity.com/packages/3d/props/electronics/hq-acoustic-system-41886>)
- Gridbox Prototype Materials (<https://assetstore.unity.com/packages/p/gridbox-prototype-materials-129127>)
-

Abbildungsverzeichnis

Glossar

Begriff	Erklärung
<i>Nativ Plugin</i>	

Zu einem Buchnachweis gehören immer folgende Angaben:

- Name des Autors oder Herausgebers eines Werkes
- Titel des Werkes
- Erscheinungsort (=Verlagsort)
- Erscheinungsjahr!

Literaturverzeichnis

Buchnachweis

Webnachweis

Zu einem Webnachweis gehören immer folgende Angaben:

- Name und Vorname des Autors
- Titel und Untertitel des Dokuments
- Internet-Adresse (URL) und Zeitpunkt der Nutzung der URL !!!