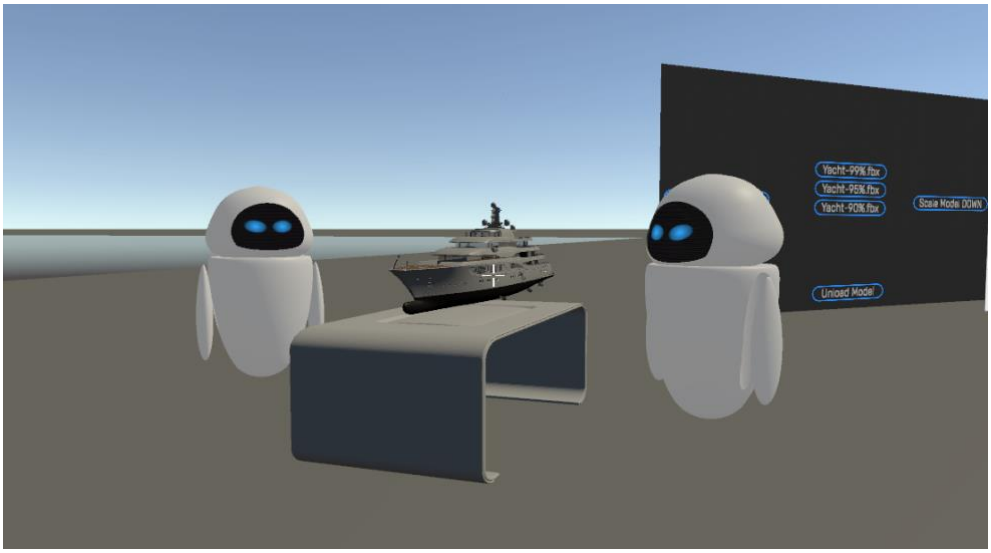
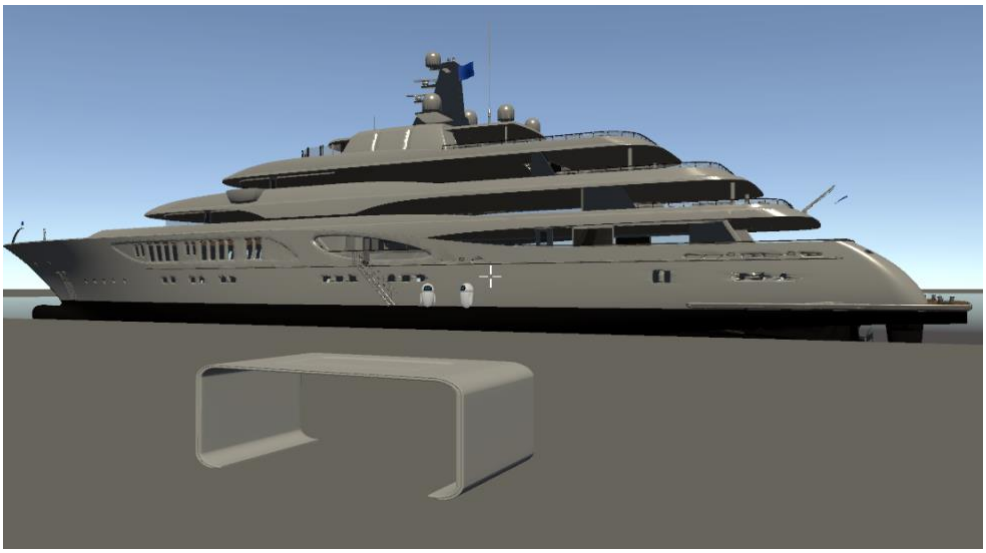
Konferenz-Software zur Präsentation

von 3D-Modellen in VR

VR CONFERENCE





Ein Projekt von:

Maarten Behn

Tim Jaeschke

Yesenia Möhring

27.01.2022

Kurzfassung

Dieses Projekt umfasst die Entwicklung einer Konferenz-Software zur Präsentation und visuellen Darstellung von 3D-Modellen.

Das Programm bietet die Möglichkeit, dass sich Nutzer in einer virtuellen Umgebung treffen. Sie können sich dort gegenseitig hören und sehen. Hochgeladene 3D-Modelle können hier präsentiert werden.

Man kann sowohl über VR sowie über einen PC mit einem normalen Bildschirm einer Konferenz beitreten. Der Ersteller der Konferenz startet einen Server und tritt daher als Host bei. Alle anderen Nutzer verbinden sich direkt mit dem Host.

Es ist möglich die ganze Konferenz ohne Verbindung zum Internet in einem lokalen Netzwerk durchzuführen. Es wird kein weiteres Gerät, auf dem ein Server läuft, benötigt.

Inhaltsverzeichnis

Einleitung

Die Idee von unserem Projekt hatte Maarten durch seinen Nebenjob bei der Firma Arvico. Dort werden 3D-Modelle von geplanten Bau- und Schiffsprojekten erstellt. Die Modelle werden in Werbevideos und Kundenpräsentationen genutzt.

Bei der Kundenvorstellung gab es allerdings das Problem, das auf Bildern und Videos der 3D-Modelle die Größe von z.B. einem Schiff nicht richtig vermittelt werden kann.

VR-Brillen bieten sich bei Präsentationen an, da mit solchen Größenverhältnisse besser verdeutlicht werden können.

Daher haben wir uns dazu entschieden, ein Konferenzprogramm zu entwickeln, in dem man 3D-Modelle online Vorstellen kann. Da so auch nicht das Problem entsteht, dass ein Kunde in einem anderen Land lebt und nicht zur Besichtigung vorbeikommen kann. Dabei loggen sich alle Nutzer in eine virtuelle Umgebung ein. In dieser kann man seinen Charakter in FirstPerson auf einem Bildschirm oder in VR bewegen.

Es gibt ein Voicechat in 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. So kann man sich einfach in dynamische kleine Gruppen aufteilen.

Während der Konferenz ist es möglich, 3D-Dateien hochzuladen. Diese können verkleinert auf einem Tisch oder in Orignalgröße zu Begehung angezeigt werden.

Vorgehensweise, Materialien und Methode

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

Netzwerkverbindung

Als erstes haben wir das Netzwerksystem entwickelt. Dies stellte sich als der komplizierteste Teil des Projekts heraus. Wir hatten noch nie mit TCP und UDP gearbeitet und uns so vorgenommen, dieses im Rahmen des Projektes zu lernen.

Unser Netzwerk basiert auf einem Host/Client Prinzip. Die Person die die Konferenz startet ist der Host. Daher wird sein Computer als Server benutzt auf dem sich jede andere Person die beitritt 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 bytes zu einem anderen Nutzer zu schicken.

Im laufe der Arbeit müssten wir den Netzwerkcode oft ergänzen, weil wir z.B. für das FileShare System eine schnellere Datenrate benötigten.

Network Package System

Für unser Programm haben wir ein Netzwerkprotokoll entwickelt, welches die rudimentären Funktionen der C# TCP und UDP libaries in ein einfach nutzbares Interface verpackt.

Dafür haben ein System entwickelt mit welchem wir Netzwerk-Pakete versenden können. Jedes Packet ist mit einem byte als Id definiert. Diese Id gibt an welche Art von Packet es ist. Jede art Netzwerknachricht, die wir senden hat eine eigene Packet 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) das angibt, ob das Package asynchrone behandelt werden soll.
- 4 bytes (1 Int) die die Länge des Inhalts angeben.
- 1 byte das die Packet-Id angibt.
- 1 byte das die User-Id des Packeterstellers angibt.

Nun folgt eine Abfolge von Daten, die für jede Art von Paket 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 Programmes egal ist ob der Benutzer ein Host oder ein Client ist. Es gibt keinen zentralen Server in der Daten-Struktur aller anderen Systeme. Daher kann die Konferenz auch ohne Aufwand in einem lokalen Netzwerk betreiben.

3D-Voicechat

Um sich während der Präsentation verständigen zu können, haben wir einen Voicechat eingebaut. Es wurde uns sehr für klar, dass es viel zu kompliziert wäre, diesen selbst zu programmieren. Deswegen haben wir eine Unity library namens Univoice genutzt. Univoice tauscht die Audio daten über Peer to Peer Verbindung aus. Jedoch ist ein zentraler signaling Server zu verbinden nötig. [6]

Univoice bietet dazu die Möglichkeit das ein- und ausgehenden Audiosignal zu jedem anderen Nutzer zu muten.

Die Audiosignale der anderen Nutzer werden an der Position von ihrem Avatar abgespielt. So hört man z.B. die Stimme eines anderen Nutzers, der links neben einem steht, auch auf dem linken Ohr.

Um das 3D Audio zu testen, haben wir die Funktion eingebaut das man Musik von Lautsprechern in der virtuellen Umgebung abspielen kann.

FileShare:

Um 3D-Modelle anzeigen zu können, benötigt jeder Nutzer in der Konferenz eine Kopie der Ursprungsdatei. Um diese automatisch zu verteilen, haben wir ein System namens FileShare gebaut.

Dieses System funktioniert wie folgt:

Jeder Nutzer besitzt eine Liste aller Dateien und der zugehörigen Informationen, 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.[7]

Benötigt nun ein Teilnehmer den Inhalt einer bestimmten Datei aus der Liste, stellt dieser eine Anfrage an einen der Nutzer, der die Datei lokal gespeichert hat. Nun starten die beiden Nutzer einen Datenaustausch. Dabei wird die Datei in kleine Pakete aufgeteilt und zum anfragenden Teilnehmer gesendet.

FBX Loader

Um in 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 es dafür entwickelt wurde mit vielen Programmen zu funktionieren und viele Programme in diesem Format exportieren können [1].

Der erste Ansatz, den wir hatten, ist mit der internen Unity-Erweiterung „FBX-Export“ zu arbeiten, da Unity das Importieren von FBX-Dateien bei Laufzeit nicht unterstützt. Durch diese Erweiterung kann man FBX-Dateien lesen und dann in für Unity verständliche daten umwandeln. Dies hat auch gut funktioniert und lief im Unity Editor einwandfrei. Bis wir das Unity Projekt zu einer fertigen Applikation umgewandelt haben, 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 dann im „Asset Store“ nach einer Lösung gesucht.

Da fast alle brauchbaren Lösungen Geld kosten, haben wir uns gegen diesen Weg entschieden. Schlussendlich haben wir herausgefunden, das man in Unity ein „Nativ Plugin“ schreiben kann. In einem „Nativ Plugin“ kann man die Programmiersprache C++ nutzen, für die es mehr „libraries“ gibt als für C#, die von Unity genutzte Programmiersprache. Autodesk stellt so eine „library“ für das FBX Format breit [2]. Es war allerdings schwer, mit dieser „library“ zu arbeiten, da sie sehr umfangreich ist und keine gute Dokumentation breitstellt. Das laden von Texturen haben wir zum Beispiel nicht hinbekommen, da es kaum Ressourcen zum Auslesen dieser gibt.

Wir können über einen 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 verschieden Plattform neu erstellt werden muss und man so mehr Aufwand hat, wenn man das Projekt für verschieden Plattformen erstellen möchte.

Zur Vorstellung haben wir ein 3D-Modell einer echten Yacht von Arvico bereitgestellt bekommen. Dieses 3D-Modell hatte in seiner Originalform über 10 Millionen Polygone. Daher mussten wir das 3D-Modell Signifikat unterkonvertieren, damit Unity das 3D-Modell händeln kann. Dabei sind leider ein paar Fehler, wie Kaputte Kanten und falsche Schatten im Modell entstanden. In unserem Projekt haben wir das Modell mit 90% -

95% und 99% Polygon-Reduktion genutzt. Bei weniger als 90% werden die FBX-Dateien zu groß und das Programm hat lags und lange Ladezeiten.

Abbildung 1 Das Modell der Yacht in Cinema 4D

VR

Um sich vor allem die Größe der 3D-Modelle besser vorstellen zu können, haben wir uns überlegt das man einer Konferenz auch in VR beitreten kann.

Um VR in Unity umzusetzen, haben wir mit OpenXR [3] gearbeitet. Das ist eine Schnittstelle, die von der „Khronos group“[4] entwickelt wurde und dient als einheitliche Schnittstelle zwischen einem Programm und einer VR-Brille. Sie macht es einfacher eine Applikation für viel verschiedene VR-Brillen zu erstellen. Desweitern haben wir mit dem SteamVR-Plugin von Valve gearbeitet, da wir zum Testen der VR Anwendung eine Valve Index VR-Brille genutzt haben. Es hat uns auch ermöglicht noch weitere Funktionen der VR Brille zu nutzen wie zum Beispiel das Erkennen von Finger-Bewegungen. Das Plugin macht es auch leichter ein VR-Spieler zu erstellen, da es schon sehr viele Beispiele gibt und vorgefertigter Code vorhanden sind.

~~UI~~ User Interface (UI)

Bei dem UI haben wir lange mit den Standard UI-Elementen von Unity gearbeitet. Als unser Projekt sich zum Ende bewegte, haben wir die Standardelemente durch simple selbsterstellte UI-Elemente ausgetauscht. Hierbei haben wir alle Grafiken in weiß gelassen. So können wir in Unity diese dynamisch einfärben und ~~switchen~~ zwischen Farb-~~Fab~~ Themes wechseln.

Zur Darstellung der Charaktere haben wir ein Modell des Disneyfilm-Charakters EVE aus Wall-E genutzt. Dieser Charakter bot sich an, da er keine Beine hat. So ist es möglich den Körper nur mit der Position des Kopfes realistisch darzustellen.

Dies war nötig da wir verschiedene Positionsdaten von First Person und VR-Nutzern bekommen. [5]

Abbildung 2 Erste Version des Startbildschirms

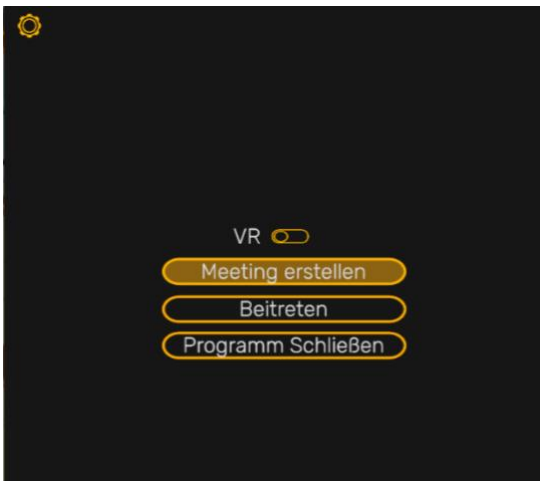


Abbildung 3 Zweite Version des Startbildschirms

Abbildung 4 Fertige Version des Startbildschirms

Ergebnisse

Unser Ergebnis ist eine Applikation, die mit der Unity Engine erstellt und es ermöglicht online eine Konferenz stattfinden zu lassen. Es ist möglich, einer Konferenz in VR beizutreten und 3D-Modelle zu präsentieren.

Um dies Realisieren zu könne, haben wir ein Netzwerk-System aufgebaut, welches die Daten zwischen Clients austauscht wie zum Beispiel eine Position.

Desweitern haben wir ein Voicechat genutzt, damit man sich untereinander 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.

Wir haben auch UI erstellt, welche 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 die 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 verschieden Betriebssystem laufen zu lassen wie zum Beispiel Android, IOS oder Linus. Zurzeit läuft das Programm nur auf Windows.

Quellen- und Literaturverzeichnis

Glossar

Begriff	Erklärung
library	*Eine Sammlung an externen Code*
Nativ Plugin	*Eine Möglichkeit eine library in der Sprache C++ einzubinden.*
TCP	*(Transmission Control Protocol) ein Netzwerkprotokoll, das definiert, auf welche Art und Weise Daten zwischen Netzwerkkomponenten ausgetauscht werden sollen.*
UDP	*(User Datagramm Protokoll) ist ein minimales, verbindungsloses Netzwerkprotokoll. Es ist in manchen Fällen schneller als TCP*
Voicechat	*Ist eine Telekommunikation über das Internet. Es wird hauptsächlich von Spielern in Mehrspielerspielen benutzt.*

Literaturverzeichnis

Webnachweis
[1] https://en.wikipedia.org/wiki/FBX (27.01.2022)
[2] https://www.autodesk.com/developer-network/platform-technologies/fbx-sdk-2020-0 (27.01.2022)
[3] https://www.khronos.org/openxr/ (27.01.2022)
[4] https://www.khronos.org/ (27.01.2022)
[5] https://de.wikipedia.org/wiki/WALL%C2%B7E_%E2%80%93_Der_Letzte_r%C3%A4umt_die_Erde_auf (27.01.2022)
[6] https://github.com/adrenak/univoice (27.01.2022)

Externe-Mittel

- Unity Game Engine 2021.1.13
- Unity Shader Graph
- Unity Cinemachine
- OpenXR
- SteamVR
- FBX SDK
- 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)

Unterstützungsleistungen

Das Projekt wurde Größtenteils bei uns Zuhause entwickelt. Teilweise haben die Betreuer uns bei Fachfragen weitergeholfen.

Anhang

Herunterladbar von:

https://drive.google.com/file/d/13_a_CHBbX0PSiRrGoN17VYdfUyIrp9B1/view?usp=sharing

- Programm (executable)
- Beispiel Code
- Screenshots des Programms
- Präsentation der Programm-Funktionen.mkv