Membran WS16/17 - Ein Kooperations-/Medienprojekt zwischen der HBK und der Universität des Saarlandes

Ba Thinh Tran, Tim Düwel, Johannes Geiser, Felix Wilcken

Betreuer: Janosch Obenauer und Michael Schmitz

Abstract

Bei diesem Projekt implementieren wir eine Methode, um einen Mensch live als 3D-modell auf die Leinwand zu bringen. Dabei trägt unser Schauspieler eine VR-Brille, welche die Zuschauer in der virtuellen Welt sichtbar für ihn macht. Aufgrund fehlender Resourcen wurde dieser Teil durch eine aktive Skype Verbindung ersetzt. Dies ist der initiale Schritt, um zu zeigen wie interaktionsvielfältig die virtuelle Welt auf der Leinwand ist.

Keywords: Virtual Reality, Motion Capture, MAD, HBK, UdS, Medienprojekt, Unity, LiveCaptury, Speech2Text2Speech, 3D-Model

Contents

List of Figures

1. Idee

Im Prinzip wollen wir eine Interaktionsmöglichkeit zwischen dem Publikum und einer 3D-Figur aufbauen. Dabei soll der Zuschauer zunächst die Figur als AI wahrnehmen, welches sich dann langsam als eine echte Person entpuppt. Die Idee kann hierbei endlos erweitert werden, indem wir einfach mehr Interaktionen einbauen oder die 3D-Modelle sowie die Umgebung an sich interessanter gestalten. Außerdem könnten ganze Geschichten oder Filmszenen in der virtuellen Welt erzählt werden, was wiederum den vorhandenen Unterhaltungsspektrum ausbreitet.

2. Aufbau

Ort 1 Dachatelier der HBK - A

Ort 2 Tonstudio der UdS - B

Figur 1 illustriert unseren Aufbau.

2.1. Dachatelier der HBK

Im Dachatelier der HBK (A) befinden sich Leinwände, an die eine 3D-Umgebung mit einem humanoiden 3D-Modell projiziert wird, welches LIVE von einem Schauspieler aus dem Tonstudio der Universität (B) gesteuert wird. Außerdem befindet sich in A eine Kinect-Kamera, deren Tiefenbild als Pointcloud nach B gestreamt und dem Schauspieler dort in Form einer virtuellen Umgebung zu Verfügung gestellt wird. Wir bieten dem Zuschauer vor Ort auch die Möglichkeit mit unserem Schauspieler durch ein Mikrofon zu reden.

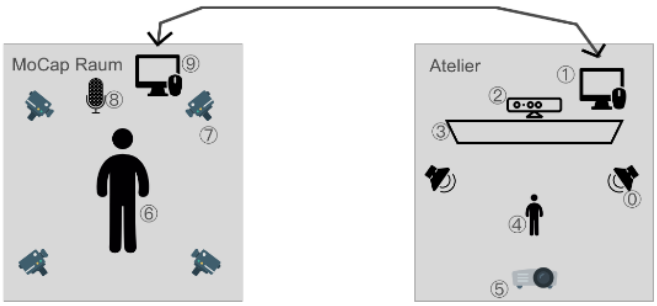

Figure 1: 1) Windows Rechner — 2) Kinect — 3) Leinwand — 4) Publikum — 5) Projektor — 6) Schauspieler — 7) Kameras — 8) Mikrofon — 9) Mocap Rechner

2.2. *Tonstudio der UdS*

Um die Bewegungen des Schauspielers zu tracken benutzen wir das MoCap-System *Captury Live*[1] von *Nils Hasler*. Das System steht im Tonstudio der UdS funktionsbereit. Hierbei arbeiten wir mit insgesamt 8 Kameras, welche teilweise an Stangenmasten und einem Gerüst befestigt sind. Mit der dortigen VR-Brille kann der Schauspieler außerdem das Publikum dank der Kinect-Aufnahme sehen.

3. Implementierung

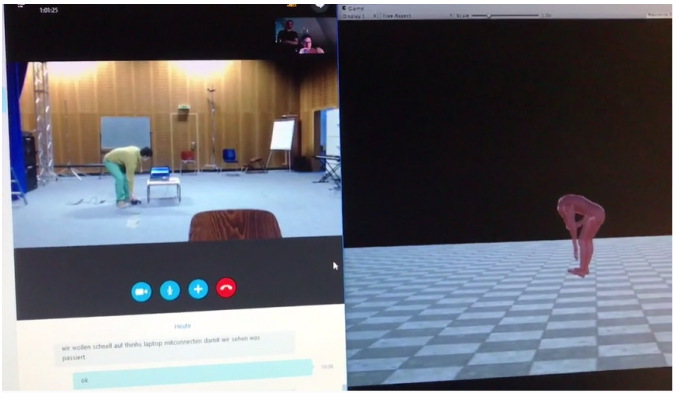

Figure 2: Ein Screenshot von der Generalprobe. Die Bewegungen des Schauspielers werden an einem 3D-Modell abgebildet.

Bei dieser Beschreibung handelt es sich um eine groben Überblick, wie wir durch das Projekt gegangen sind. Dabei lassen wir hier einige Details aus, die ohnehin schwer zu beschreiben sind. Ein Blick in den Projektordner und ein paar Einarbeitungsstunden genügen, um alles aufzunehmen. Hier ist unser git-repo: (https://github.com/PewhProgrammer/Membran).

3.1. *Konzeptentwicklung und 3D-Modell* - ***Felix***

Hierbei wurde in den ersten Anfangswochen gründlich über das zugrundeliegende Konzept geredet. Die 3D-Umgebung/Figur in Unity wurde von Felix aus der HBK modelliert und bearbeitet. Leider konnten wir das Modell nicht verwerden, da das Skinning nicht auf die Rotationsdaten der LiveCaptury passten.

3.2. *Verbindung zwischen zwei Unity Instanzen an verschiedenen Orten über das Internet* - ***Thinh/Tim***

Es gibt hierfür drei Möglichkeiten, die wir getestet haben.

- Peer-to-peer
- Dedicated Server
- PhotonEngine[2]

Das Problem bei peer-to-peer war die Verwendung des MasterServers von Unity. Dieser erlaubt es uns zwar die öffentlichen IP-Adressen von unseren Mitspielern zu finden, jedoch funktionierte das Verbinden aufgrund von NAT Problemen nicht (NAT punchthrough hat auch nichts gebracht).
Dann war die Arbeit an einem dedicated Server in der kurzen Zeit zu aufwändig , wäre jedoch wohl die ideale Lösung gewesen. Damit hätten wir unabhängig unsere Bandbreite bestimmen und das Datenvolum regeln können.
Letztendlich haben wir uns für PhotonEngine entschieden, eine unabhängige Netzwerk-Plattform, die uns ihren eigenen Server bereitstellen(natürlich mit begrenzter Bandbreite). Hierbei stellt uns Photon einen "Raum" zur Verfügung, welchen wir als Client betreten können. Nun können wir verschiedene Daten (Strings, floats, ...) mit begrenzter Größe mit Hilfe von "Remote-Procedure-Calls" (RPC) über den Server an Clients im selben Raum schicken. Die Daten werden von der entsprechenden Methode mit dem **[PunRPC]** Attribut empfangen. Dabei ist zu beachten, dass nur RPCs zwischen Objekten aufgerufen werden können, welche die gleiche PhotonView-ID haben, wobei eine Photon-View selbstverständlich als Unity-Komponente an die entsprechenden GameObjects angehängt werden muss.

3.3. *LiveCaptury: Übertragung und Verwertung der Rotationsdaten* - ***Thinh***

Dies lässt sich in drei Abschnitte gliedern: das Importieren der Daten in Unity, das Übertragen der Daten an den Unity Client der HBK und das Koppeln der Daten an das dort vorhandene Modell.

3.3.1. *LiveCaptury-Feed in Unity*

Hierbei hat uns *Nils Hasler* gut betreut und geholfen. Er hat uns ein Plug-in zur Verfügung gestellt mit dem wir die Rotationsinformationen aus LiveCaptury live in Unity importieren können. Das Plug-in wurde entsprechend angepasst, beispielsweise haben wir uns auf die Aufnahme von einer Person beschränkt.

[1] http://www.thecaptury.com/captury-live/ - Aufgerufen Februar 27, 2017

[2] https://www.photonengine.com/en/PUN - Aufgerufen Februar 27, 2017

3.3.2. *Rotationsdaten zwischen zwei Unity-instanzen verschicken*

Die Rotationsdaten bestehen aus ca. 180 floats, die wir per RPC übertragen haben.

3.3.3. *Rotationsdaten an Skelett anbinden*

Dies passiert ebenfalls im Plugin. Es muss darauf geachtet werden, dass die Knochen am 3D Modell die gleichen Namen tragen wie von LiveCaptury vorgegeben, da das Plugin anhand dieser Information entscheidet, welcher Knochen angesprochen wird. Außerdem hatten wir einige Probleme mit dem Skinning unseres 3D Modells. Das Template Modell von Nils Hasler hingegen funktioniert einwandfrei.

3.4. *Kinect: Aufnahme* ***Johannes***

3.4.1. *MimeSys*

Zum Aufnehmen von Videos mit der Kinect haben wir zuerst das Programm MimeSys verwendet. Dabei hatten wir Datenmengen von 700MB in 15s bei höchster Qualität. Hier wurde die Kinect V1 verwendet. Bei der Kinect V2 waren die Daten geringer, lagen aber auch da bei hohen Werten.

3.4.2. *Kinect Studio*

Das Verwenden der Kinect v2 funktionierte nicht an allen PCs, dabei hatte sie an einem PC einen Wackelkontakt und an dem anderen wurde sie nicht erkannt. Lediglich an einem PC funktionierte sie. Beide Programme konnte man nur zur Aufnahme nicht aber zum Streamen verwenden, wobei MimeSys eine Option zum Streamen in Zukunft liefern möchte.

3.5. *Streaming -* ***Tim***

Point Cloud, Lokal

Zum Streamen der Daten von der Kinect nach Unity verwenden wir das durch Microsoft bereitstehende Unity Plugin[3]. Damit können wir direkt auf die (Tiefen-)Bilddaten der Kinect zugreifen und diese entsprechend verwerten, sodass in Unity das korrekte Tiefenbild wiedergegeben wird.

Point Cloud, Internet

Die Generierung des Tiefenbilds haben wir komplett von der Kinect entkoppelt, bedeutet unser Skript benötigt nichts weiter als ein float-array, welches entsprechend verwertet wird. Dadurch mussten wir nur noch die vorhandenen Tiefeninformationen übertragen. Diese versuchten wir per RPC zu versenden, leider waren sie aber selbst nach Komprimierung zu groß, weshalb die Verbindung seitens Photons nach wenigen MS geschlossen wurde. Daher beschlossen wir lediglich das RGB-Video zu verwenden.

[3] https://developer.microsoft.com/de-de/windows/kinect/tools - Aufgerufen Februar 27, 2017

Figure 3: Die Point-Cloud

3.5.1. *RGB-Video, Internet -* ***Johannes***

Das Streamen von Point-Cloud Daten aufgrund hoher Datenmengen war auf die Kürze nicht mehr realisierbar, daher musste auf eine Alternative zurückgegriffen werden, nämlich den Gebrauch der Kinect als gewöhnliche Webcam. Da die Kinect standardmäßig nicht als Webcam funktioniert, benötigt man hierfür eine spezielle Software[4].

1. Man lädt die Software auf der Seite runter.
2. Entpackt sie nach ”C:/KinectCamV2”
3. Führt die install.bat als Admin aus.
4. Hat man Skype bereits installiert, muss man unter *%AppData%* das Verzeichnis ”Skype” löschen.

Anschließend sollte die Kamera als Webcam funktionieren.

3.6. *Interaktion -* ***Tim***

Um unseren Schauspieler an der Universität mit dem Publikum an der HBK kommunizieren zu lassen, realisierten wir eine Speech-to-text-to-speech Übertragung.
Dabei spricht der Schauspieler in ein Mikrofon. Diese Eingabe wird mit Hilfe der Unity DictationRecognizer Klasse[5] in Text umgewandelt, welcher per RPC an die Kunsthochschule übertragen wird. Dort wird ein Plugin[6] verwendet, welches die Windows-Sprachausgabe (verfügbar ab Vista) nutzt, um den Text wieder in Sprache zu verwandeln. Ein besonderes Feature hierbei ist, dass bei Sprachbefehl (”Stopp!”) die Übertragung aus- und eingeschaltet werden kann. Parallel dazu verschwindet das Modell oder taucht wieder auf. Zu beachten ist, dass der DictationRecognizer nur auf Windows 10 funktioniert. Außerdem muss unter ”Einstellungen/Datenschutz/Spracherkennung, Freihand und Eingabe” Cortana aktiviert sein.

References

[4] http://codingbytodesign.net/2014/07/20/kinectcamv2-for-kinect-v2/ - Aufgerufen Februar 27, 2017

[5] https://docs.unity3d.com/ScriptReference/Windows.Speech.DictationRecognizer.html - Aufgerufen Februar 27, 2017

[6] http://www.chadweisshaar.com/blog/2015/07/02/microsoft-speech-for-unity/ - Aufgerufen Februar 27, 2017