Digital Humanities im Museum 4.0

Julia Albers, Jannick Bock, Felix Harms, Luca Junge, Pepijn Kampschöer,
Leon Leymann, Christian Lüth, Björn Schultz, Fabian Sehm, André Stadtelmeyer

Hochschule Bremerhaven

Interaktion mit den Modellen

Damit die BesucherInnen eines Museums auch den Vorteil eines 3D-Modells gegenüber eines Fotos ausnutzen können, müssen dafür Möglichkeiten der Interaktion geschaffen werden. Hierfür wurden im Projekt eine Touch-Steuerung über ein Web-Frontend, eine VR-Steuerung und eine Controller-Steuerung durch ein Gyroskop entwickelt. Neben der direkten Steuerung wurde eine grundlegende Infrastruktur zur Kommunikation der einzelnen Steuerungen mit dem Web-Frontend, welches das Modell darstellt, entwickelt.

Infrastruktur

Um mit den Modellen zu interagieren, benötigt es eine dazu passende Infrastruktur. Im Projekt selbst wurde dazu ein Linux-Computer genutzt. Dieser kann z.B. ein eigenes WLAN aufbauen und wird dann im Bridge-Modus betrieben. Dadurch ist er sowohl im LAN verfügbar, sodass die von ihm gehostete Modell-Webseite auch von anderen Computern im LAN erreichbar ist. Die Steuerungs-Elemente melden sich am WLAN des Linux-Computers an. Als zweite Schnittstelle bietet der Linux-Computer einen WebSocket-Server. Dort meldet sich sowohl die Modell-Webseite an, als auch die Steuerungs-Elemente. Dadurch kann das genutzte Steuerungs-Element seine Eingaben an die Webseite weiterleiten.

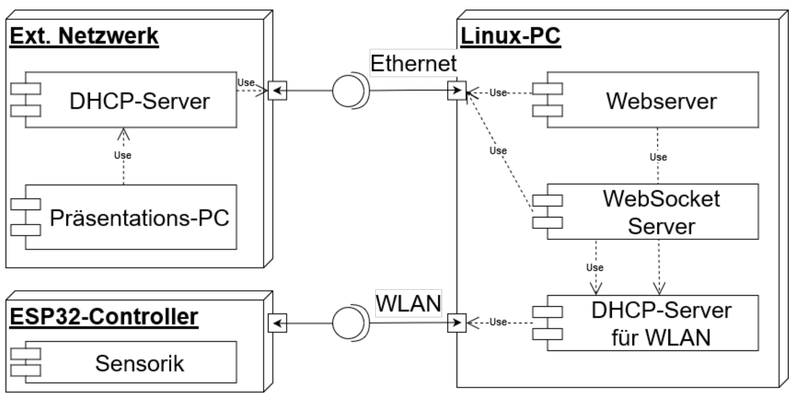

Abbildung 1: Netzwerkdiagramm am Beispiel der Controller-Steuerung

Die Modell-Webseite besteht aus einer three.js Umgebung und wird über den Linux-Computer dem restlichen Netzwerk zur Verfügung gestellt. Dadurch, dass die Modelle zur Zeit direkt auf dem Linux-Computer liegen, wird die Zugriffszeit verringert, was bei Modellgrößen von bis zu 200 MB bemerkbar ist.

Controller-Steuerung der 3D-Modelle

Aufbau des Controllers

Der Controller besteht aus zwei wesentlichen Komponenten: Einem ESP32-basierendem Mikrocontroller mit WLAN-Chip und einem Gyroskop-Sensor. Der Sensor ermöglicht es, die Neigung zu messen, sodass diese über den ESP32-Controller an die Modellseite gesendet werden kann. Zusätzlich wurden im Laufe des Projekts noch vereinzelte Taster und damit zusätzliche Funktionen hinzugefügt. Zu den Knopf-Funktionen gehört ein Modellwechsel und der Zoom des Modells.

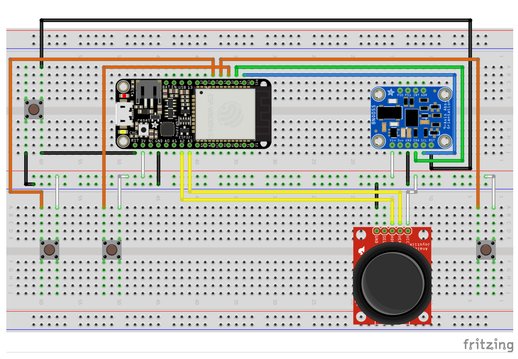

Abbildung 2: Schaltbild des Controllers

Der Controller befindet sich im Schaltbild in der oberen linken Hälfte. Rechts daneben befindet sich das Gyroskop. Der Taster oben links dient zum Reset des Controllers und des Sensors, die Taster unten links werden zum Zoomen genutzt. Der auf dem Schaltbild unten rechts zu sehende Joystick dient zur Verschiebung des Modells. Dadurch ist es möglich, einen bestimmten Bereich des Modells in den Fokus zu stellen und zu vergrößern. Durch den Taster rechts vom Joystick lässt sich ein Modellwechsel auslösen.

Der Controller verbindet sich bei Start mit dem WLAN des Linux-Computers und daraufhin mit seinem WebSocket-Server. Sobald dies geschehen ist, sendet der Controller bei Statusänderungen der Taster und des Gyroskops die aktuellen Zustände. Diese werden über den WebSocket-Server an das Frontend mit dem 3D-Modell übertragen.

Das 3D-Modell kann auf der Steuerungs-Webseite mithilfe des Bedienelements um die eigene Achse rotiert werden.

Touch-Steuerung der 3D-Modelle

Entwickelt wurde eine weitere Steuerung für touchfähige Endgeräte, wie zum Beispiel einem Smartphone oder einem Tablet. Dabei werden, ähnlich wie bei der Controller-Steuerung, die 3D-Objekte auf einer Modell-Webseite mit three.js dargestellt. Die Modelle werden vorgeladen um Ladezeiten beim Modellwechsel zu minimieren. Auf der Steuerungs-Webseite (QR-Code) sind Bedienelemente, mit denen die Rotation des 3D-Modells bestimmt werden kann. Die jeweiligen Rotationsdaten werden über eine Web-Socket-Verbindung an die Modell-Webseite geschickt, ausgewertet und schließlich in eine Rotation des Modells umgesetzt.

WebVR / WebXR

WebVR ist eine offene Spezifikation, die es Software-EntwicklerInnen ermöglicht, VR-Headsets sowie deren Controller in gängigen Browsern wie Chrome, Firefox und Safari nutzbar zu machen. So können mithilfe dieser Schnittstelle VR-Inhalte erstellt und NutzerInnen somit eine weitere Möglichkeit gegeben werden, ihr VR-Headset zu nutzen. [1]

Gestartet wurde das Projekt von Mozilla im Jahr 2014, als Virtual Reality und entsprechende Headsets einen immer größeren Nutzen fanden und der Ruf nach Unterstützung im Browser lauter wurde. Seitdem entwickeln mehrere Teams an dieser Spezifikation und den Schnittstellen. Heutzutage werden viele der bekanntesten VR-Headsets unterstützt, darunter Oculus Rift, Oculus Go, HTC VIVE und auch preigünstige Projekte wie Google Cardboard.

Da seit mehreren Jahren auch Augmented Reality (AR) und andere Formen der technikgestützten Erweiterung der Realität immer mehr Anklang finden, wurden die Bestreben unter der WebXR Device API zusammengefasst. So wurde ein einheitlicher Standard für diese Erweiterungen geschaffen. Das Ziel der Spezifikation ist es, VR- und AR-Geräte zu erkennen, die Möglichkeiten im Bezug auf Bewegung, Drehung und Neigung des Headsets sowie der Controller zu erfassen und mittels weiteren 3D-Bibliotheken wie three.js visuell zu nutzen.

Nutzung und Anwendungsfälle

Neben der wahrscheinlich größten Nutzung von VR und AR im Bereich der Videospielentwicklung, haben sich weitere Bereiche entwickelt, in denen diese neuen Technologien zum Einsatz kommen. Eine Anwendungsmöglichkeit ist die Bereitstellung von Inhalten über 3D- oder 360-Grad-Videos, aufgenommen mit speziellen Kameras. So können z.B. Stadtführungen, Ballonfahrten, ein Gang über Ausgrabungsstätten und Konzerte realitätsnah über VR-Geräte zur Verfügung gestellt werden. Auch KünstlerInnen haben die Möglichkeit ihre Projekte und Kunstwerke in den Browser zu integrieren und so für ein größeres Publikum verfügbar zu machen.

Da im Museumskontext das Thema VR und auch AR immer wichtiger wird, nutzen wir WebVR, um mithilfe der Photogrammetrie digitalisierte Modelle von den Ausstellungsstücken des Schifffahrtsmuseums sowie anderen interessanten Objekten zu visualisieren und den AnwenderInnen Informationen zu den einzelnen Objekten zu vermitteln. Dabei greifen wir über WebVR auch auf den Controller des VR-Headsets Oculus Go zu, um mit den Modellen zu interagieren und sie drehen und verschieben zu können. Durch die Nutzung dieser Interaktionsmöglichkeit erweitern wir die pure Darstellung, wie sie z.B. in 360-Grad-Videos zu finden ist, die die Inhalte dadurch noch realistischer wirken lassen.

Vorteile und Nachteile

Durch die Verfügbarkeit von kompatiblen Browsern auf den meisten Geräten ist der Einstieg in die Welt der Virtual Reality auch für unerfahrene Nutzer sehr einfach. Es ist keine Installation und Entwicklung einer eigenständigen App notwendig, sondern nur der Aufruf einer Webseite, was den Aufwand zur Nutzung der Technologie so gering wie möglich hält. Für die Darstellung vieler Szenen reicht bereits ein Mittelklasse-Smartphone aus, sodass kein Geld für High-End-PCs oder Smartphones investiert werden muss.

Allerdings sind der Browser sowie die erforderlichen Bibliotheken in ihren Leistungsfähigkeiten limitiert. Für große, leistungshungrige Anwendungen, wie Videospiele oder die Visualisierung hochdetaillierter 3D-Modelle ist die Leistung des Browsers nicht mehr ausreichend, wodurch auf spezialisiertere Programme und Entwcklungsumgebungen zurückgegriffen werden muss. Dies erhöht die Qualität des Ergebnisses, jedoch auf Kosten des Entwicklungs- und Nutzungsaufwandes.

WebVR Einbindung in three.js

Für die Nutzung mit three.js existiert eine JavaScript-Bibliothek, die man wie three.js in die HTML-Seite einbinden kann. Three.js kümmert sich größtenteils schon um die korrekte Darstellung der VR-Inhalte, lediglich die Methode, die für die Erstellung der Bilder für die beiden Augen zuständig ist, muss angepasst werden. Code, der für three.js-Szenen ohne VR geschrieben wurden, läuft größtenteils auch im VR-Modus. Die Kopf-Drehungen werden ohne größere Anpassungen auf die bereits vorhandene Kamera angewendet. Lediglich die Bewegung und Interaktion mit dem Controller muss abgefragt und entsprechende Methoden zur Interaktion mit der Szene hinzugefügt werden. [2]

Querverweise

Github-Repository des Controllers
`https://github.com/knigh7m4r3/dh2018-controller`

Literatur

[1] Group, Immersive Web C.: *Welcome to the immersive web*. `https://webvr.info/`. Version: Mai 2018. – abgerufen am 22. Januar 2019

[2] three.js: *How to create VR content*. `https://threejs.org/docs/#manual/en/introduction/How-to-create-VR-content`. Version: 2018. – abgerufen am 22. Januar 2019