Mixed Reality, WS 2025/26

Kuer-Abgabe

Michael Kaup

michael.kaup@student.htw-berlin.de
Matr.Nr.: s0589545
Datum: 31 January 2026

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument stellt die Projektdokumentation für mein Projekt im Mixed Reality Modul des Wintersemesters 2025/2026 dar. Folgend lässt sich in Abschnitt 2 der bereits vorgestellte Themenvorschlag für das Projekt finden, sowie in Abschnitt 3 eine Abschlussdokumentation die sich mit der letztendlichen Umsetzung des Projektes befasst. Der Themenvorschlag wurde hier der Vollständigkeit wegen und für kontextuelle Konsistenz noch einmal mit aufgeführt.

Die aktuelle Version des Projektes kann unter diesem GitHub-Pages Link, https://pizzapichael.github.io/MRUebung3WS25_KuerTeil/, ausprobiert werden.

2 Themenvorschlag

2.1 Thema

Die Szene soll eine Karikatur der aktuellen Entwicklung sogenannter „Künstlicher Intelligenz“ am Beispiel eines Chatbots darstellen. Sie soll die übertrieben angepriesene Darstellung der Künstlichen Intelligenz durch Entwickelnde und Teilhabende sowie die den „Künstlichen Intelligenzen“ zugrunde liegenden gesellschaftlichen Problematiken gegenüberstellen.

2.2 Szenenaufbau

Die spielende/beobachtende Person befindet sich in einem Vorraum, vor sich eine Tür. Diese Tür kann geöffnet werden. Tritt man durch die Tür, gelangt man in einen großen, einem Thronsaal ähnlichen Raum. In diesem schwebt der Tür gegenüber, am anderen Ende des Raumes ein Computer über einer mit Stufen erreichbaren Podest. Spielende können sich im Thronsaal frei bewegen. Mit dem Computer können Spielende interagieren. Er zeigt das Chatfenster einer ChatGPT Sitzung an. Es soll möglich sein, via Tastatureingabe mit ChatGPT zu interagieren, so, wie man es an seinem Computer tun würde.

Hinter dem Computer führen Kabel hinter einen Vorhang oder einen anderweitig abgegrenzten Bereich. Folgt man den Kabeln in diesen Bereich, gelangt man zu einer Tür, in der die Kabel verschwinden. Diese Tür kann wiederum geöffnet werden und eine Art Vision trifft die Spielenden. Ein Video wird abgespielt, in dem Klickworker, Auswirkungen von Datenzentren auf Umwelt und Menschen, Psychische Probleme und weitere den „Künstlichen Intelligenzen“ zugrunde liegende, fragliche, gesellschaftliche Gegebenheiten gezeigt werden.

Wenn das Video endet, steht man nun vor der geschlossenen Tür, das Kabel durchgetrennt und kaputt. Der Raum füllt sich langsam mit Rauch. Zurück im Thronsaal sieht man Flammen, die sich überall ausbreiten. Am PC sieht man den Bildschirm nun ausgeschaltet. Gelegentlich flackern jedoch Botschaften wie „Wir wollen doch nur eine bessere Zukunft“, „Vertrau uns doch einfach“, „Wir wollen nur das beste für die Menschheit“ über den Bildschirm. Nach einiger Zeit wird der Rauch undurchdringbar und das Bild schwarz. Dann endet die Szene und man kann sie mittels Button-Klick neu starten.

3 Abschlussdokumentation

3.1 Entwicklungsumgebung

Die Projektszene wurde mit dem auf ThreeJS und WebXR aufbauenden MR-Framework A-Frame erstellt. Dieses nutzt HTML und Javascript um MR-Szenen zu inszenieren. Entwickelt wurde das Projekt in einer Windows-Umgebung. Das HMD mit dem die Entwicklung getestet wurde und für das das Projekt vorgesehen ist, ist die Meta Quest 3.

3.2 Abschließende Szene

Die unter Abschnitt 2 vorgeschlagene Szene konnte ich in ihrem Umfang nicht umsetzen. Welche Gründe dazu geführt haben werden im späteren Verlauf unter Abschnitt 3.6 thematisiert. Die endgültige Szene teilt sich in zwei Teile auf. Die VR-Entities und die AR-Entities. Beide Szenen-Teile haben gemein, dass sie sich grundsätzlich nur mit Controllern des HMD vollständig bedienen lassen. Ein Bewegen in der Szene ist zwar auch im Desktopmodus möglich, es kann aber nicht über die Interaktions-Arten Bewegung und Umschauen hinaus mit der Szene interagiert werden.

Werden Controller genutzt, werden diese in beiden Szene-Teilen mit digitalen Modellen der Controller repräsentiert.

3.2.1 VR-Entities

In dieser Szene stehen User*innen in einem großen, überdimensionalen Thronsaal. Links und rechts der Kamera befinden sich Säulen, vor denen kleinere Fackel-Säulen stehen. Über diesen sind flackernde Lichtquellen platziert um den Anschein von Fackellicht zu erzeugen. Hinter den Säulen sind große Fenster in den Wänden eingelassen, die den Anschein erwecken sollen, dass außerhalb des Thronsaals noch mehr existiert. Hinter der Kamera befindet sich ein großes, geschlossenes Holztor. Alles in diesem Thronsaal ist so dimensioniert, dass sich User*innen klein vorkommen sollen, als ob dieser Thronsaal nicht für sie, sondern für etwas oder jemand größeren und wichtigeren als sie erbaut wurde.

Die Kamera blickt zu Beginn der Szene direkt auf den Altar. Dieser steht auf einer mit Stufen versehenen Empore. Die Stufen sind ebenfalls so skaliert, dass sie den Anschein erzeugen nicht von den User*innen bestiegen werden zu können, ohne große Körperliche Anstrengungen durch Klettern auf sich nehmen zu müssen.

Der Altar ist ein prunkvoller, hölzerner Altar. Über diesem schwebt ein Computer, der sich in einer wiederholenden auf und ab Bewegung befindet und das Logo von ChatGPT mit einer sich wiederholenden rotierenden Animation abbildet. Die Empore wird von zwei weiteren Fackel-Säulen gesäumt und hinter dem Altar hängen zwei zur Seite gezogene rote Vorhänge sowie ein weiß, gelblich durchsichtiger Vorhang.

Vor der Empore steht ein steinernes Podest, auf dem sich eine wie Papier aussehende Fläche befindet. Das Podest wird von einem warmen gelben Licht erleuchtet.

Nähert man sich der Empore und somit auch dem Computer fängt dieser in einem bestimmten Radius an kontinuierlich die Kamera zu verfolgen, was den Anschein erweckt, von diesem beobachtet zu werden. Nähert man sich noch weiter, und geht auf das Podest zu erhebt sich das Papier auf dem Podest und schwebt nun über dem Podest. Auf dem Papier befindet sich ein ChatGPT-ähnliches UI mit einem blinkenden Cursor in der Texteingabe-Zeile um zu suggerieren, dass hier Text eingegeben werden kann. Klickt man mit dem Raycaster auf diese Texteingabe-Zeile, erscheinen zwei Beispielprompts, die mit dem Raycaster angewählt werden können. Je nach Auswahl wird auf dem Papier ein kurzer, vorgegebener Chatverlauf angezeigt.

User*innen haben die Möglichkeit um die Empore herum und in den sich dahinter befindenden Gang zu gehen. In diesem sieht man bereits am Ende des Ganges einen Raum mit schwarzen Wänden, hier auch Blackbox genannt, und einigen fliegenden Objekten darin. Vom

Ende des Ganges aus führt ein eingezäunter Pfad auf die Mitte des schwarzen Raumes zu, dort befindet sich ein Stromgenerator mit einer überdimensionierten Steckdose und einem darin steckenden Stecker. Ein Scheinwerfer scheint vom Raumeingang her auf die sich in der Mitte des Raumes befindliche Steckdose.

Die fliegenden Objekte sind Plane-Objekte, die alle jeweils mit einem mit negativen gesellschaftlichen Folgen der KI-Ära assoziierten Begriff versehen sind. Sie bewegen sich in Kreis/ Helixform um die Mitte des Raumes auf und ab und richten ihre Frontseite mit dem Text auf das Generator Objekt, als ob sie dieses anschauen würden. Bewegt sich die Kamera in den Raum hinein, schauen die Planes die Kamera an, bewegen sich aber weiterhin kreisend um die Mitte des Raums. Nähert sich die Kamera weiter erstarren sie in der Luft und schauen nun nur noch die Kamera an.

Der Stecker in der Steckdose lässt sich per Controllerinteraktion aus der Steckdose ziehen. Wenn man dies tut, teleportiert sich der Computer auf dem Altar vom Altar herunter auf die Stufen der Empore, so als ob ihm der Strom abgestellt wurde und er wortwörtlich abgestürzt wäre.

3.2.2 AR-Entities

Die Szene beinhaltet folgende identische Objekte wie die VR-Szene:

- Computer
- Podest mit ChatGPT-UI
- Kleiner Generator mit Steckdose und Stecker
- Fliegende Planes mit Stichworten

Der Hauptunterschied ist, dass diese Objekte anders angeordnet sind, sowie der Thronsaal nicht angezeigt wird, wodurch die Umgebung der User*innen mit entsprechenden HMDs gesehen werden kann.

Was die Platzierung der Objekte betrifft, sind diese deutlich näher an der Kamera platziert. Das Podest befindet sich direkt vor der Kamera, nur ca. zwei Meter entfernt. Das Papier auf dem Podest schwebt direkt zu Beginn der Szene in der Luft. Dreht man sich im HMD um, sieht man ca. einen Meter hinter sich den Generator mit Steckdose und Stecker. Außerdem schwebt der Computer deutlich näher an der Kamera und die Planes schweben nicht um den Generator, sondern um den Computer und gucken die Kamera von Anfang an an.

Die Interaktionen sind mit denen in der VR-Szene identisch.

3.3 Interaktion

Es gibt vier Arten mit der Szene zu Interagieren. Die erste ist die Fortbewegung, diese findet mittels Teleportation statt. Teleportation ist eine gängige, etablierte und nutzerfreundliche Fortbewegungsweise in MR- und vor allem VR-Anwendungen. Ich habe mich für diese entschieden, da sie das Risiko durch die Fortbewegung in einer VR-Anwendung Motion Sickness zu erleiden deutlich reduzieren soll. Die Teleportation wird mit dem Trigger, der Button der üblicherweise mit dem Zeigefinger der Hand bedient wird, des linken Meta Touch Controllers ausgeführt.

Die zweite Interaktions-Art ist der Raycaster mit dem mit dem ChatGPT-UI und den Elementen darauf interagiert werden kann. Der Raycaster ist generell immer sichtbar. Mit dem Betätigen des Triggers des rechten Meta Touch Controllers lassen sich die UI Elemente auswählen.

Die dritte Art ist das Greifen von Objekten, mit den Grip-Buttons der Meta Touch Controller. Die Grip-Buttons sind die Buttons, die mit den Mittelfingern der Hände bedient werden. Um ein Objekt zu greifen muss das Controller-Modell in das Objekt geführt werden und dann der Grip-Button dieses Controllers betätigt werden. Z.Z. ist nur der Stecker in der Steckdose greifbar.

Die vierte und letzte Art der Interaktion ist das Umherschauen durch Bewegung des HMD durch den dreidimensionalen Raum. Durch die Bewegung des HMD im dreidimensionalen Raum ließe es sich, bei einem realen Raum entsprechender Größe, auch durch die gesamte Szene bewegen.

3.4 Design der Szene

Die Szene folgt keinem festen Designpattern. Grundsätzlich habe ich hier die Funktion der Objekte, also etwas bestimmtes darzustellen wie z.B. einen Computer oder einen Altar, über ein konsistentes Design gestellt. Einzige Design-Anforderung war, dass die Objekte im Thronsaal, bis auf den Computer, einen mittelalterlichen Look haben, sollten, sowie die Objekte innerhalb der Blackbox einem neumodischen/futuristischen Design folgen dürfen. Dieser Unterschied sollte die Diskrepanz zwischen der auf Laien undurchsichtig und einfach wirkenden und eventuell, mit Blick auf positive gesellschaftliche Veränderungen, überschätzten Technologie der LLMs und deren tatsächlichen Komplexität und Folgen verdeutlichen. Alle Assets sind im Literaturnachweis mit Link zur entsprechenden Quelle für schnelle Einsicht aufgelistet. Das Thronsaal Modell ist ein selbst erstelltes Asset, dass mehrere einzelne Assets kombiniert.

3.5 Technische Dokumentation

3.5.1 Build-Tool

Für die Entwicklung und das Deployment wurde Vite als Build-Tool eingesetzt. Vite bietet einen schnellen Entwicklungsserver mit Hot Module Replacement. Die Konfiguration erfolgt über die `vite.config.js` im Wurzelverzeichnis.

3.5.2 Projektstruktur

Im root-Verzeichnis befindet sich eine `index.html`, in der die A-Frame-Szene erstellt wurde. Diese beinhaltet die Einbindung aller benötigten externen A-Frame-Bibliotheken, Custom-Components sowie aller Assets, also 3D-Modelle und Texturen. Die Szene selbst ist in zwei Hauptkomponenten unterteilt: `vr-entities-component.js` und `ar-entites-component.js`, die jeweils die VR- und AR-Entities spawnen und verwalten.

Die Ordnerstruktur gliedert sich wie folgt:

- `src/components/custom/`: Enthält alle selbst geschriebenen A-Frame-Komponenten
- `src/components/external/`: Externe Komponenten wie die A-Frame-Tastatur (die jedoch zum Zeitpunkt dieser Dokumentation nicht in der Szene genutzt werden)
- `src/models/`: 3D-Modelle im GLTF/GLB-Format mit zugehörigen Texturen
- `src/textures/`: Bildtexturen für Materialien und UI-Elemente
- `docu/`: Projektdokumentation in **typst**

3.5.3 A-Frame und Dependencies

Das Projekt basiert auf A-Frame Version 1.7.0, einem WebXR-Framework das auf Three.js aufbaut. A-Frame ermöglicht es, MR-Szenen mit HTML-ähnlicher Syntax zu erstellen und mit JavaScript-Komponenten zu erweitern.

Folgende externe A-Frame-Packages werden eingesetzt:

- **aframe-physics-system**: Ermöglicht physikalische Simulationen mittels den Physics-Engines Ammo.js oder Canon.js. Das Package wird für Kollisionserkennung und die Physics-basierten Interaktionen genutzt, z.B. bei der Kollisionserkennung von Stecker und Boden.
- **aframe-extras**: Bietet zusätzliche Komponenten und Controls, darunter die erweiterte Controller-Unterstützung.
- **ammo.js**: Die zugrundeliegende Physics-Engine, die von aframe-physics-system verwendet wird.

Die externen Dependencies werden per Content Delivery Network eingebunden, wodurch keine lokale Installation der Packages notwendig ist. Es können falls gewollt jedoch die lokalen Dependencies in der index.html auskommentiert werden um diese, falls installiert, offline zu nutzen.

3.5.4 Custom Components

Für die spezifischen Anforderungen der Szene wurden mehrere Custom Components entwickelt, die das Verhalten und die Interaktionen der Entities steuern:

Szenen-Management:

- `vr-entities-component.js`: Spawnt und verwaltet alle VR-spezifischen Entities
- `ar-entites-component.js`: Spawnt und verwaltet die AR-Variante der Szene mit angepasster Objektplatzierung

Interaktions-Komponenten:

- `grabbable-custom.js`: Ermöglicht das Greifen von Objekten mit den Meta Touch Controllern
- `hand-grab.js`: Verwaltet die Grip-Button-Interaktion und das Halten von Objekten
- `chat-interaction.js`: Steuert die Interaktion mit dem ChatGPT-UI
- `teleport-component.js`: Erweiterte Teleportations-Logik für spezifische Szenenanforderungen
- `on-event-teleport.js`: Ermöglicht Event-basierte Teleportation von Entities (z.B. Absturz des Computers)

Visuelle Effekte und Animationen:

- `spiral-movement.js`: Erzeugt die Helix-Bewegung der fliegenden Text-Planes um den Generator bzw. Computer
- `paper-glide-upwards.js`: Lässt das ChatGPT-UI-Papier vom Podest aufschweben
- `look-at-camera-component.js`: Lässt Objekte kontinuierlich zur Kamera schauen
- `blink-cursor.js`: Simuliert einen blinkenden Textcursor im ChatGPT-UI
- `torchlight-spawner.js`: Erstellt flackernde Lichtquellen für die Fackeln im Thronsaal
- `spawn-text-planes.js`: Generiert die Text-Planes mit KI-kritischen Begriffen

Hilfsfunktionen:

- `proximity-detection-by-circle-componente.js`: Erkennt Kamera-Nähe zu Objekten und triggert Events
- `on-event-deactivate-components.js`: Deaktiviert spezifische Komponenten bei bestimmten Events
- `camera-position-debug.js`: Debug-Tool zur Ausgabe der Kamera-Position während der Entwicklung
- `camera-persistence.js`: Debug-Tool, Speichert und lädt Kamera-Positionen zwischen Sessions

Alle Components folgen der A-Frame-Komponenten-API mit `init()`, `update()`, `tick()` und `remove()` Lifecycle-Methoden.

3.6 Problemstellungen und Lösungen

Wie bereits in Abschnitt 3.2 erwähnt, konnte die Idee der MR-Szene aus dem vorangegangenen Themevorschlag nicht vollumfänglich umgesetzt werden. Die Idee war vor allem für den gegebenen Zeitrahmen zu aufwändig. Des Weiteren führten Erkenntnisse und Entwicklungsprobleme dazu, dass ich die Szene in ihrem funktionalen sowie visuellen Umfang deutlich einschränken musste. Einige dieser Erkenntnisse und Probleme führe ich folgend auf.

3.6.1 Zeitliche Limitation

Die ursprüngliche Idee beinhaltete z.B. eine Video-Vision beim Öffnen einer Tür abzuspielen, die die gesellschaftlichen Nachteile der KI-Industrie verdeutlichen sollte. Allerdings wäre die

Recherche und das Zusammenschneiden von geeignetem Videomaterial sehr zeitaufwendig gewesen, zumal ich mich nur wenig mit der Erstellung von Videos auskenne. Das Ganze hätte außerdem nur einen geringen Beitrag zu der eigentlichen Aufgabe geleistet, eine MR-Anwendung zu erstellen, weshalb ich mich vorerst für eine simplere Darstellung dieser gesellschaftlichen Nachteile als fliegende Textplanes entschieden habe.

Des Weiteren konnten Features wie die Interaktion mit einer ChatGPT-Sitzung, die Interaktion mit Türen, Physics-Basierte Kabel sowie das in Flammen stehen des Thronsaals aufgrund ihrer Komplexität zeitlich nicht umgesetzt werden.

3.6.2 Limitationen durch A-Frame und Community Komponenten

Die Entwicklung in A-Frame stellte sich schnell als komplizierter heraus als im Vorhinein angenommen. Auch wenn das Framework eine einfache Implementation von VR-Szenen verspricht, ist es jedoch insofern limitiert, dass in meinem Entwicklungsprozess einige der Community Komponenten nicht mit der aktuellen A-Frame Version funktionierten, ein Beispiel ist die Teleport Komponente [1] oder die Komponente die das Greifen von Objekten ermöglichen sollte [2].

Einige wurden zuletzt vor mehreren Jahren aktualisiert und lassen sich dementsprechend nicht mit der aktuellen Version von A-Frame nutzen. Auch ließ sich nicht einfach sagen, welche A-Frame Version die Funktion aller Komponenten gewährleistet hätte. Andere, wie die Super-Hands-Component[2] funktionierten aufgrund von Fehlern nicht.

Auch A-Frame selber war an einigen Stellen eine Limitation, als dass die Dokumentation meiner Meinung nach nicht tief genug geht und nicht alle möglichen Parameter und Anpassungsmöglichkeiten ausreichend veranschaulicht.

Das alles kombiniert führte dazu, dass ich mich zuerst mit der Entwicklung entsprechender Komponenten auseinander setzen musste, wie z.B. einer Teleport- und eine Grab-Funktion. Um den Entwicklungsaufwand jedoch so gering wie möglich zu halten, habe ich die Hilfe eines LLM-Assistenten genutzt, mehr dazu in Abschnitt 3.7.

3.6.3 Debugging

Wie in jedem Projekt ist Debugging ein integraler Bestandteil des Entwicklungsprozesses. Da wie in Abschnitt 3.6.2 beschrieben jedoch die Dokumentation meiner Meinung nach nicht ausreichend ist, war das Debugging an einigen Stellen sehr Ressourcenaufwendig. Dazu kommt, dass auch Komponenten wie das A-Frame Physics-System[3] eher mit einer spärlichen Dokumentation daherkommen, was seine Implementation erschwerte.

3.6.4 Physisc-System

Ein wiederkehrendes Problem war, dass ich die Physics-Einwirkung des Plugs oder des Computers habe ausschalten wollen oder Properties des physics-bodies während des Renderprozesses ändern wollte. Das ist aber anscheinend nicht möglich, wird jedoch in der Dokumentation nicht erwähnt. Durch Ausprobieren habe ich festgestellt, dass es zwar möglich ist, z.B. die ammo-body Komponente, die das physikalische Verhalten des Objektes bestimmt, während die Szene gerendert wird zu entfernen und wieder hinzu zu fügen. Danach lassen sich allerdings die Attribute dieser nicht mehr bearbeiten und es wird nur eine „leere“ ammo-body Komponente hinzugefügt. Das führt dazu, dass das Objekt zwar Gravity erfährt, aber z.B. keine Kollisionen zwischen dem Objekt und anderen statischen Objekten wie dem Boden möglich sind und das Objekt ins unendliche fällt. Auch das Mesh Wrapping der Komponente im sog. „hull“ Modus ist relativ ungenau und platziert Physics Meshes oft mit einem Offset zu der eigentlichen Objektposition.

3.7 LLM Nutzung

3.7.1 LLMs als Ersatz für Internetrecherche und zur Fehlerbehebung

In diesem Projekt habe ich in Kollaboration mit dem GitHub Copilot sowie dem Agent System der Cursor IDE gearbeitet. Der Großteil meiner Kollaboration besteht dabei aus Anfragen, die die Suche im Internet nach Einzelheiten zu Programmiersprachen ersetzen, z.B. wie bestimmte Datentypen funktionieren oder, aus Aufforderungen, sogenannte Sisyphos Arbeit zu verrichten, siehe Abb. 1.

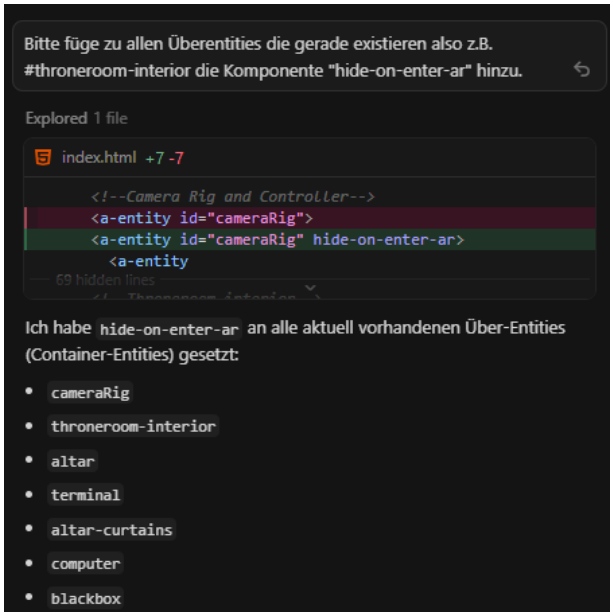

Abb. 1: Aufforderung an den Codex Agent, mehrere Stellen in der index Datei mit einer bestimmten Komponente zu versehen.

Des Weiteren benutze ich LLMs für die Fehler-Erläuterung und Behebung, wie z.B. in zu sehen.

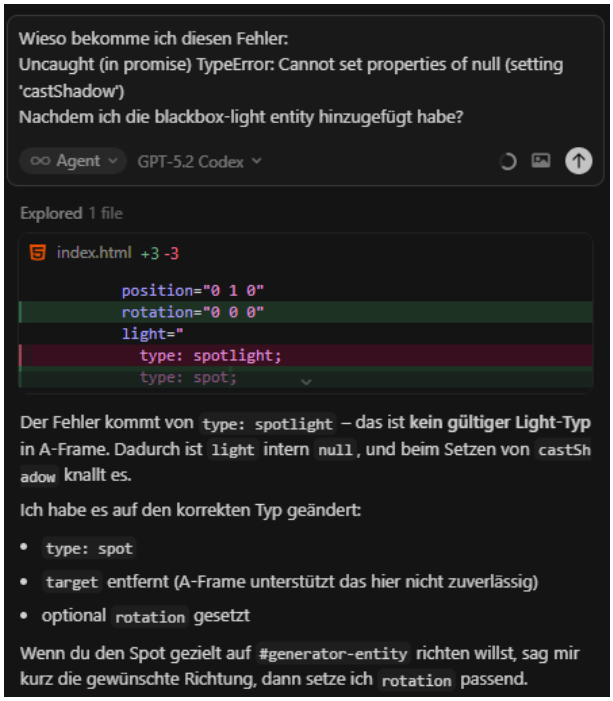

Abb. 2: Aufforderung an den Codex Agent, eine Fehlerursache zu erläutern.

3.7.2 Kollaboration mit LLMs

Einige Skripte wurden unterdessen komplett von Copilot geschrieben, weil ich diese entweder nur für Debugging Zwecke brauchte und keine Zeit in diese investieren wollte, oder weil es eigentlich eine Community Komponente für diese Funktion gab, die Komponente aber bei mir nicht funktioniert hat. Um Entwicklungsaufwand zu sparen und weil die Nutzung einer bereits von einem Menschen geschriebenen Komponente für mich das gleiche ist, wie eine von einem LLM verfasste Komponente zu nutzen, habe ich als Ersatz das LLM eine Komponente verfassen lassen.

Die betroffenen Komponenten sind folgende:

- `camera-persistence.js`
- `camera-position-debug.js`
- `teleport-component.js`

Weitere Komponenten wurden in starker Kollaboration mit einem LLM geschrieben. Starke Kollaboration meint hier, dass das LLM einen großen Einfluss auf Funktion und Struktur der Komponente hat, mein eigener Einfluss aber mindestens genauso relevant und umfangreich ist, weshalb nicht genau trennbar ist, welche Teile nun mein Code und welche Teile das LLM verfasst hat. Solche Komponenten waren entweder von mir oder dem LLM erstellt und im Laufe öfter von mir und dem LLM modifiziert worden.

Die Komponenten sind folgende:

- `grabbable-custom.js`
- `hand-grab.js`
- `chat-interaction.js`
- `chat-interaction.js`

Alle anderen Komponenten wurden ebenfalls von einem LLM teilweise modifiziert, aber nicht in einem Rahmen der die unter Abschnitt 3.7.1 genannten Nutzungsarten überschreitet. Alle relevanten Komponenten, auf die ein LLM nennenswerten Einfluss hatte wurden entsprechend kommentiert.

Generell verfolge ich bei der Nutzung von LLMs im Programmierprozess den Ansatz, mir die Arbeit nicht abnehmen zu lassen, sondern sie zu erleichtern. Ich lege großen Wert darauf, die am Ende vorhandenen Scripte, Komponenten und Programmierungsansätze in vollem Umfang verstehen und nachvollziehen zu können.

Durch meine berufliche Erfahrung, die ich während meines gesamten Studiums bereits gesammelt habe, weiß ich, dass LLMs im professionellen Programmierungskontext unverzichtbar geworden sind. Deswegen lehne ich es ab, dieses Werkzeug nicht im universitären Kontext zu nutzen und auch hier weitere Erfahrungen im Umgang damit zu machen. Ich bin mir meiner Verantwortung jedoch bewusst, LLMs korrekt einzusetzen und deren Nutzung erkenntlich zu machen.

3.8 Fazit

Das Projekt zeigt, dass die Entwicklung einer funktionsfähigen MR-Anwendung mit A-Frame grundsätzlich machbar ist, jedoch mit erheblichen technischen Herausforderungen verbunden ist. Die ursprünglich konzipierte Szene musste aufgrund zeitlicher und technischer Limitationen deutlich reduziert werden, jedoch hat die finale Umsetzung ihr konzeptionelles Ziel nicht zwingend verfehlt.

Die abschließende Szene vermittelt die anfangs gewollte inhaltliche Aussage: Die Gegenüberstellung des überdimensionalen Thronsaals, der LLMs als allmächtige, über uns thronende Entität inszeniert, mit der dunklen Blackbox und den negativen gesellschaftlichen Konsequenzen funktioniert auch in reduzierter Form. Die implementierten Interaktionsmöglichkeiten ermöglichen User*innen eine ausreichende Interaktion mit der Szene.

Die größte Herausforderung stellte die unzureichende Dokumentation von A-Frame und dessen Community-Komponenten dar. Viele externe Packages sind veraltet oder inkompatibel mit der aktuellen A-Frame-Version, was zur Eigenentwicklung von Komponenten zwang. Besonders das Physics-System erwies sich als problematisch, da Limitationen bei der Laufzeit-Manipulation von Physics-Bodies nicht dokumentiert waren.

Abschließend lässt sich festhalten, dass das Projekt trotz der Abweichungen vom ursprünglichen Konzept eine funktionsfähige MR-Erfahrung ist, die ihre kritische Aussage zur Rolle von LLMs in der Gesellschaft grundsätzlich herüberbringt.

Literatur

[1] F. Serrano, „Developing an A-Frame Teleport Component“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://aframe.io/blog/teleport-component/

[2] vincentfretin, „aframe-super-hands-component“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://github.com/c-frame/aframe-super-hands-component

[3] Tpleme, „aframe-physics-system“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://github.com/c-frame/aframe-physics-system

[4] „Medieval Door“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/medieval-door-1a23bbb237d74d13a9619f666dd32d37

[5] „Ancient Temple Arch“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://www.turbosquid.com/3d-models/free-ancient-temple-arch-2219641

[6] „Medieval Gate LP“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://www.turbosquid.com/3d-models/medieval-gate-lp-877974

[7] „Classic Column 3D“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://www.turbosquid.com/3d-models/classic-column-3d-2470888

[8] „Pillar Torch“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://www.turbosquid.com/3d-models/free-pillar-torch-1901904

[9] „Cathedral Window“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://www.cgtrader.com/free-3d-models/architectural/window/cathedral-window

[10] „Altar“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/altar-de75993005dd46fc851d8ff8e4a6ad87

[11] „Altar Lowpoly Concept“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/altar-lowpoly-concept-470042a85f554decb86891770ab75677

[12] „Curtain 774“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/curtain-774-239728f92357417fa31a0defdbf875e5

[13] „Victorian Curtain“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/victorian-curtain-3a7e18700d5c4033a67d1d143b3cf64c

[14] „Old Paper“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/old-paper-0a7cf9d5bf7742b3a711d39eed36332f

[15] „O2 Generator“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/o2-generator-c825c67a6db545568b4b1124f49a5ee2

[16] „Concrete with Fence“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/concrete-with-fence-c388ac2189104bc38b155e81ce59330b

[17] „Europe Wall Socket“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/europe-wall-socket-7232d542bfca4f0ea17b4285da6ef7ab

[18] „Diesel Generator Low Poly Game Asset“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/diesel-generator-low-poly-game-asset-03db834f3fd94212a6a07d3127630b3b

[19] „Power Plug European“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://sketchfab.com/3d-models/power-plug-european-4b73c69e083f43329491dff671721515

[20] „OpenAI ChatGPT Logo Icon“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://www.vecteezy.com/png/22227364-openai-chatgpt-logo-icon

[21] OpenAI, „ChatGPT“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://chatgpt.com/

[22] „Rock Tile Floor“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://polyhaven.com/a/rock_tile_floor

[23] „Floor Tiles 06“. Zugegriffen: 31. Januar 2026. [Online]. Verfügbar unter: https://polyhaven.com/a/floor_tiles_06