Realitätsnahe Nachbildung von Umgebungen für interaktive Echtzeit-Simulationen

Martin Weber[1], Hans Standfest[1], Niklas Räder[1], Lukas Iwai[1], Benjamin Erl[1], Cheyenne Emmerlich[1], Jannik Büge[1], Melissa Weidlich-Rau[1], Manuel Heinzig[1], Alexander Kühn[1], Christian Roschke[1], Matthias Vodel[1], Mike Espig[2], Michaela Banert[2], Marc Ritter[1]

1 Hochschule Mittweida, Fakultät Angewandte Computer- & Biowiss.,Technikumplatz 17, 09648 Mittweida
2 Westsächsische Hochschule Zwickau, Data Science Research Group, Kornmarkt 1, 08056 Zwickau

Abstract

Renderingmethoden für die Darstellung von virtuellen Umgebungen werden immer realistischer (Tatarchuk, 2009) und finden häufiger Anwendung außerhalb der Spieleindustrie (Wu und Fu, 2019).

Eine Studentengruppe der Hochschule Mittweida (HSMW) beschäftigte sich im Verlauf von 3 Semestern, im Rahmen des Lehrkonzeptes *Digital Skills and Products* (Ritter et al., 2019) unter der Leitung von Prof Ritter, mit der Erstellung einer Echtzeitsimulation des Roboters *Double 3* von *Double Robotics* (Double Robotics, 2022).

Als Teil des Projekts entstand eine virtuelle Nachbildung der Büroräume der Data Science Research Group (DSRG) der Westsächsischen Hochschule Zwickau (WHZ).

Außerdem beinhaltet die Simulation sowohl eine Programmierschnittstelle, als auch eine Webapplikation, über welche der virtuelle Roboter wahlweise angesteuert werden kann. Ein speziell dafür erstellter Prototyp wurde zum Ende des zweiten Projektsemesters evaluiert, um die Benutzerfreundlichkeit der Anwendung und den visuellen Realismus der Umgebung von Personen bewerten zu lassen, die mit den Büroräumen der DSRG vertraut sind.

Die Evaluation zeigt, dass die Echtzeitsimulation für digitales Lernen im Zusammenhang mit Künstlicher Intelligenz (KI), sowie für das Training eines maschinellen Lernmodells geeignet ist. Künftig soll der Prototyp an der WHZ für Lehrzwecke adaptiert und ein Algorithmus für maschinelles Lernen angebunden werden, welcher gleichzeitig mit dem realen Double Roboter kompatibel ist.

1. Einführung

Die Leistung von Computern sowie von Echtzeit-Rendering-Technologien steigt stetig und Werkzeuge zur Erstellung von virtuellen Umgebungen werden immer effizienter und nutzerfreundlicher (Nordhaus, 2007). Virtuelle Echtzeitumgebungen spielen dabei auch außerhalb der Spieleindustrie eine wichtige Rolle, wie z. B. bei der Produkt- und Architekturvisualisierung oder digitalen Führungen von Gebäuden. Eine weitere Anwendung ist das Training von künstlichen Intelligenzen (KI) in einer virtuellen Umgebung, um diese später auf die echte Welt zu übertragen. Im Rahmen des Lehrprojekts "Wissenschaft und Wirtschaft" nach (Ritter et al., 2019) entstand ein Studentenprojekt an der HSMW, welches sich drei Semester (25ECTS) lang mit der Erstellung einer solchen Simulationsumgebung beschäftigt hat. Dieses Projekt war eine Zusammenarbeit zwischen der

HSMW und der WHZ unterstützt vom Forschungsprojekt Saxony5 (Saxony5, 2022). Innerhalb des Lehrmodules hat eine Gruppe von sieben Studenten die Büroräume der DSRG der WHZ und das Modell des *Double* Telepräsenzroboters (Double Robotics, 2022), mit hohem Detailgrad und Originaltreue nachgebildet. Dieser ist ein 2-rädriger, selbstfahrender Roboter, der über Kameras sowie weitere Sensoren verfügt und von der Ferne über eine Webanwendung gesteuert werden kann. Die Simulation stellt eine Schnittstelle bereit, über die ein virtuelles Abbild des Roboters komplett gesteuert und dessen Sensorik abgefragt werden kann. Das ermöglicht es bspw. ein maschinelles Lernmodell virtuell zu trainieren und dieses Modell später auf einen echten Roboter zu übertragen. Am Ende Projekts erfolgte die Durchführung einer Evaluation des Prototypen. Diese untersuchte ob sich die Echtzeitsimulation für digitales Lernen im Zusammenhang mit Künstlicher Intelligenz (KI), sowie für das Training eines maschinellen Lernmodells eignet. Im Folgenden wird zusätzlich der Entstehungsprozess der Simulationsumgebung beschrieben.

2. Grundlagen

Maschinelle Lernmodelle sind nur so gut, wie die Daten, welche sie für das Training benutzt haben (Rudraraju und Boyannapally, 2019). Für KIs, die in einer virtuellen Umgebung trainiert und anschließend auf die echte Welt übertragen werden, bedeutet dies, dass die Lernumgebung so realitätsnah wie möglich an der Umgebung sein muss, in der das Modell letztendlich angewendet wird. Folglich muss die Umgebung, in der ein Modell für einen Double Roboter im Einsatz in den Büroräumen der DSRG trainiert wird, so originalgetreu wie möglich zu diesen Räumen sein.

In anderen Bereichen finden virtuelle Nachbildungen von Gebäuden ebenfalls Anwendung. Modelle können zur Beschleunigung von Bauprojekten beitragen (Arayici und Hamilton, 2005) oder im Immobillien-Service-Sektor eingesetzt werden (Mahdjoubi, 2013). In den meisten Fällen kommen dafür 3D-Laserscanning-Techniken und automatisierte Prozesse für die Modellerstellung zum Einsatz (Mahdjoubi, 2013). Damit ist es möglich, millimetergenaue Modelle herzustellen. Jedoch erfordern derartige Techniken spezielle Vorraussetzungen. Solch eine Vorgehensweise war für das gegebene Projekt nicht möglich, da die hohen Voraussetzungen nicht erfüllt werden konnten und weil die Technik des Laserscannings verschiedene Nachteile hat. Die Verwendung von 3D Scans liefert ein schnelles und akkurates Ergebnis, jedoch sind diese nicht für die Anwendung in Echtzeit optimiert und somit nicht für diesen Anwendungsfall geeignet. Ein weiteres Argument gegen 3D Scans ist die fehlende Modularität. Objekte können nicht individuell nachbearbeitet werden, sondern es entstehen große unhandliche Modelle, welche viel anspruchsvolle Optimierung benötigen. Aus diesen Gründen wurden die Räumlichkeiten und darin enthaltene Objekte manuell erstellt.

Da das Training eines maschinellen Lernmodells für einen Roboter in Echtzeit ablaufen muss, wurde die Anwendung mithilfe der 3D-Spielengine *Unity (Unity Technologies, 2022)* erstellt. Diese beinhaltet ein grundlegendes Framework, in dem Spiele und ähnliche Echtzeitanwendungen in 3D ablaufen können. Außerdem stellt der Unity-Editor einen großen Teil der benötigten Werkzeuge zur Verfügung, um die gewünschte Anwendungen zu erstellen. Die Engine bietet letztlich noch eine wertvolle Programmierschnittstelle, um die Programmierung in C\# (.Net) zu ermöglichen.

3. Systemarchitektur

Die Systemarchitektur ist aus den früh im Projektverlauf ermittelten Anforderungen an die Anwendung hervorgegangen. Über ein Webinterface soll die Simulation beobachtet, sowie mit dieser interagiert werden können. Zudem soll die Simulation unabhängig davon sein, ob gerade ein Nutzer das Interface nutzt oder nicht. Im Einsatz muss der virtuelle Roboter extern über eine Programmierschnittstelle gesteuert werden können.

Die folgende Grafik zeigt schematisch den Aufbau des Gesamtsystems:

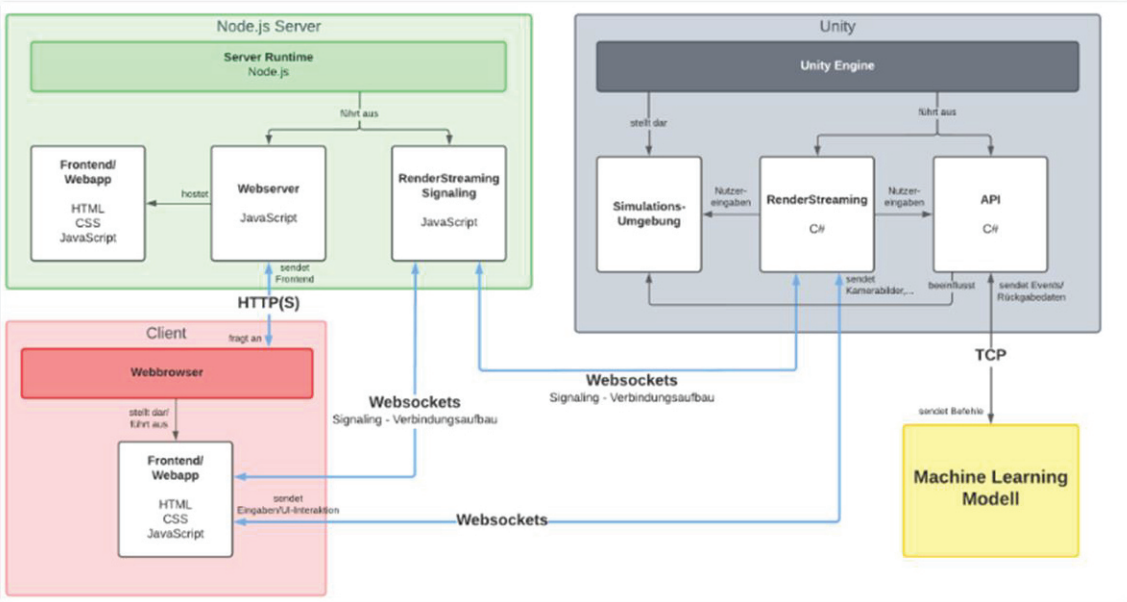

Abbildung 1: Ein Node.js Webserver (Grün) hostet das Webinterface und ist Verbindungsvermittler zwischen Unity-Simulation (Grau) und Client/Webbrowser (Rot). Eine peer-to-peer-Verbindung zwischen Client und Simulation ermöglicht einen Videostream an die Webapplikation.Die Programmierschnittstelle/API erlaubt einem Machine Learning Modell (Gelb) die Steuerung des Roboters

Die *Unity Game Engine* bietet verschiedene Pakete bzw. Rendertechnologien zur Darstellung von Grafik mit verschiedenen Realismusgraden und Leistungsanforderungen, die sog. "Render Pipelines". Für die Umsetzung der Simulation wurde sich für die "High Definition Render Pipeline" (HDRP) entschieden. Diese bietet das Potential für den höchstmöglichen grafischen Realismus in der Gameengine. Eine realitätsnahe Optik war im Projekt wichtig, da die Simulation im Zusammenhang mit einem maschinellen Lernmodell eingesetzt werden soll, sodass es möglich wäre, das Modell in der Simulation zu trainieren und dann auf den echten Roboter zu übertragen. Aus diesen Gründen wurde sich für diese Rendertechnik entschieden und die virtuelle Umgebung mithilfe dieser umgesetzt.

Unity bietet die Möglichkeit Projekte als Webanwendungen zu exportieren, welche dann WebGL für das 3D-Rendering nutzen. Hierbei wird jedoch die HDRP nicht unterstützt, welche für das Projekt benötigt wurde. Stattdessen wird die Simulation auf einem Server ausgeführt und das dabei dargestellte Bild an ein einfaches Frontend in Echtzeit übertragen. Zudem sollte die Simulation, z.B. für das Training eines maschinellen Lernmodells, unabhängig von der Nutzung der Weboberfläche auf einem Server ausführbar sein. Aus diesen Gründen war eine komplexere Architektur für das System erforderlich.

Das Webinterface bzw. Frontend der Simulation macht es möglich, die Simulation zu beobachten und mit dieser zu interagieren. Der Webserver ist nun nicht nur für die Bereitstellung der Informationen zuständig, sondern dient auch als Vermittler für den Verbindungsaufbau zwischen Website und Simulation. Statt die Simulation direkt im Browser des Clients auszuführen, verbindet sich dieser mit der Simulation auf dem Server. Die Simulation sendet über die Verbindung einen Bild-Stream, der im Browser angezeigt wird und so die Simulation für den Nutzer darstellt. In der entgegengesetzten Richtung sendet das Frontend Eingaben des Nutzers an die Simulation, welche dann für die Interaktion weiter verarbeitet werden. Für den Verbindungsaufbau sowie die spätere Datenübertragung werden Websockets als Kommunikationmittel verwendet.

4. Programmierschnittstelle (API)

Damit ein maschinelles Lernmodell in der Simulationsumgebung trainiert und anschließend reibungslos auf den realen *Double 3* übertragen werden kann, ist die Programmierschnittstelle eine möglichst genaue Nachbildung des d3-sdk (Double3-sdk, 2022). Das macht es Programmierern möglich, Anwendungen für den Roboter zu entwickeln, welche die Bewegungs- und Navigationsmöglichkeiten des *Double 3* nutzen. Der Roboter kann anschließend also komplett über diese Schnittstelle gesteuert werden.

Wie auch das d3-sdk, nutzt die replizierte Schnittstelle JSON (JavaScript Object Notation) als Nachrichtenformat für jegliche Kommunikation. Der Unterschied zur originalen API (Application Programming Interface) liegt darin, dass keine Unix Domain Sockets (auch IPC Sockets genannt) genutzt werden können. Diese bilden unter Unix-Systemen eine Schnittstelle für die lokale Kommunikation zwischen Programmen auf einem Computersystem. Stattdessen wird das Netzwerkprotokoll TCP (Transmission Control Protocol) zur Kommunikation verwendet, welches die Datenübertragung nicht nur lokal sondern auch zwischen mehreren Computern ermöglicht. Grund dafür ist, dass die Entwicklung auf Windows-Systemen stattfand und dort die Benutzung von rohem IPC nur schwer durchführbar ist. Durch TCP wird zudem ermöglicht, dass die Simulation nicht auf demselben Computer ablaufen muss, auf dem das Lernmodell ausgeführt wird. Da der IPC Socket beim Double vom Typ Stream ist unterscheidet sich die Handhabung im Vergleich zu TCP nur im Verbindungsaufbau, nicht aber bei der Übertragung von Daten. Somit ist die Anpassung der KI, bei der Übertragung auf den realen Roboter, mit nur geringem Aufwand verbunden. Die Unity-Anwendung startet einen TCP-Server, welcher nun auf Verbindungen von Clients wartet, die Befehle an die Simulation senden möchten.

Die verwendeten JSON-Befehle bestehen aus zwei Teilen. Der erste Teil ist das Kommando, welches ausgeführt werden soll (Beispiel: "base.pole.setTarget"). Der zweite Teil enthält eine Liste aller Parameter welche an die Methode übergeben werden (Beispiel: "percent": 0.3). In Unity werden empfangene Befehle dann interpretiert und die entsprechende Funktion des Roboters aufgerufen. Hierbei wird die interne Klassenstruktur anhand des Kommandonamens nach der angefragten Methode durchsucht und diese dann aufgerufen. Dadurch ist das System einfach um neue Funktionen erweiterbar, weil keine manuelle Zuordnung zwischen Anfrageformat und Aufruf benötigt wird, sondern die internen Methoden direkt auf dieses Format abgebildet werden.

5. Erstellungsprozess der Umgebung

Um die Räumlichkeiten der DSRG nachzubilden, fand eine Exkursion zu den Büroräumen der WHZ statt. Anhand der vor Ort genommenen Maße konnte ein Raumplan, sowie Vorlagen für die spätere Nachbildung der Einrichtungsgegenstände erstellt werden. Um eine möglichst genaue Nachbildung zu erhalten, erfolgte die Vermessung der Räumlichkeiten sowie Möbel mit Zollstöcken und Lasermessgeräten. Die Durchführung einer detaillierten Rekonstruktion erforderte die Vermessung aller Baugruppen, die größer als ein Dezimeter sind. Zusätzlich erfolgte die Aufnahme von Referenzbildern und Maßen der Möbelstücke. Auf Grundlage eines von der WHZ bereitgestellten Grundrisses entstand ein Modell in *Autodesk CAD* (Autodesk, 2022). Als Ergebnis der Exkursion ergab sich ein digitaler Raumplan, 354 Referenzbilder, sechs erfasste Räume inklusive Blick auf den Außenbereich, sowie 45 verschiedene vermessene Objekte im Raum.

5.1. Modellierung

Die Referenzen der Exkursion bildeten eine Grundlage für die Modellierung der Modelle. Zunächst ist die Struktur der Wände basierend auf dem Grundriss entstanden. Als nächstes erhielt diese Struktur Ausschnitte für Fenster und Türen und es erfolgte die Platzierung von Säulen. Die aufgenommenen Bilder dienten außerdem als Vorlage zur Modellierung der Möbel im 3D-Modellierungsprogramm *Blender* (Blender, 2022). Die Modelle beinhalten so wenig Geometrie wie möglich, ohne dabei relevante Abstriche im Detailgrad zu machen. Abgeschrägte Kanten sorgen für eine realistischere Lichtberechnung, was dem Modell zusätzlich mehr Detailgrad verleiht. Aus diesen Modellen entstand ein 2D Netz (UV-Map) für die Texturierung. Die Erstellung eines identischen, höher aufgelösten Modells für die Generierung von Oberflächendetails auf der Textur ermöglichte einen höheren Realismus ohne Leistungseinbußen.

Die Texturierung, also das Einfärben der Modelloberflächen, erfolgte mit *Substance Painter* (Substance, 2022). Um den hohen Realismusgrad zu wahren, erhielt jedes Objekt dort eine individuell erstellte Textur. Danach begann der Import und die Platzierung der fertig texturierten Modelle in Unity. Mit Hilfe des Raumplans und der Bilder konnten die Objekte mit hoher Genauigkeit platziert werden.

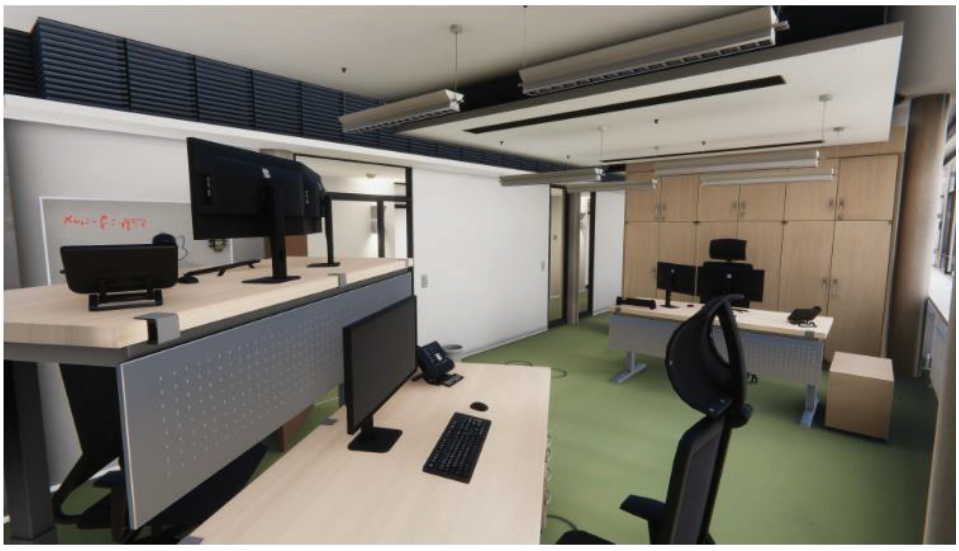

Abbildung 2: Virtuelle Nachbildung der Büroräume der DSRG

5.2. Lighting

Anschließend erfolgte die Beleuchtung der Szene. Lichtquellen, die sich an den Stellen befinden, wo es im Büro ebenfalls Lampen gibt, sorgen für eine realitätsnahe Helligkeitsstimmung. Zusätzlich simulieren Lichtflächen an Türen und Fenstern, die nach Außen führen, einfallendes Sonnenlicht. Die Beleuchtung aller unbeweglichen Objekte der Umgebung ist bereits vor der Ausführung des Programms berechnet und gespeichert (Light Baking (Unity Documentation: Baked Light, 2022). Dies ermöglicht annähernd realistische Schattierungen, die in Echtzeit dargestellt werden können (siehe Abb. 2).

Reflexionen werden mittels Screen-Space-Reflections (SSR) dargestellt. Diese Technik verwendet bereits vorhandene Bilddaten im Grafikspeicher, um Reflektionen nachträglich zu berechnen.

6. Evaluation

Um die Benutzerfreundlichkeit der Anwendung und den visuellen Realismus der Umgebung von Personen bewerten zu lassen, die mit der Umgebung vertraut sind, fand eine Evaluation über Zoom statt. Dabei war das Ziel auch die Untersuchung, ob sich die Echtzeitsimulation für digitales Lernen im Zusammenhang mit KI, sowie für das Training eines maschinellen Lernmodells eignet. Außerdem sollte die generelle Qualität des Prototypen, also die der Modelle und Texturen, sowie die Handhabung der Anwendung, eingeschätzt und mögliche Fehler gefunden werden.

Für diese Zwecke entstanden zwei Fragebögen für die Beurteilung der Anwendung durch die Probanden, wie auch ein speziell für diese Evaluation entwickelter Prototyp mit zusätzlichen Funktionen. Ein eigenständig programmiertes Autopilotsystem diente dabei als Platzhalter für ein maschinelles Lernmodell, wodurch eine eigenständige Navigation des Roboters durch die Simulationsumgebung ermöglicht wird. Dieses nutzt das von Unity bereitgestellte *NavMesh*-System, welches automatisch eine Karte der Szene generiert. Sie gibt Auskunft darüber, welche Areale betretbar sind und welche nicht. Mittels NavMesh-Agent wird so ein Pfad gefunden, über den sich der Roboter von einem Punkt zum nächsten bewegen kann. Der Autopilot lies sich dabei über ein einfaches Menü konfigurieren. Hier kann der Nutzer neue Zielorte hinzufügen oder bereits bestehende umbenennen.

Um die Handlungen der Probanden während der Evaluation zu verfolgen und eine Überprüfung verschiedener Funktionen bereitzustellen, kam eine Liste mit Aufgaben direkt in der Anwendung zum Einsatz. Diese befand sich am linken Bildschirmrand. Dort waren bestimmte Punkte der Räumlichkeiten aufgelistet, zu welchen die Probanden navigieren sollten. Begibt sich ein Proband zum jeweiligen Punkt, wird die Aufgabe automatisch in der Liste als erledigt markiert und der Proband bekommt Feedback, dass er diese absolviert hat.

Zusätzlich entstand ein Logging-System für die Analyse zur Nutzung des Programms, welches die Position des Double im Programm und das Erledigen der Aufgaben im Verlauf des Testzeitraums aufzeichnete. Ebenso erfolgte die Aufnahme der Bildrate, wie auch der Hardware des Probanden, für die Behebung von möglichen Fehlern.

Die Probanden für die Evaluation wurden von der DSRG gestellt und sind alle mit der den realweltlichen Räumlichkeiten vertraut. Zuvor erhielt jeder den Prototyp als Windows-Anwendung, um mögliche Probleme bei einer Verteilung während der Evaluation und Fehler beim Starten zu verhindern. Das bedeutet aber auch, dass ein möglicher Ersteindruck verloren gehen könnte, da die Probanden den Prototypen bereits vor der Evaluation ausprobieren hätten können.

Es kam schließlich zur erfolgreichen Durchführung der Evaluation mit sechs Probanden. Diese führten selbständig alle Aufgaben durch und beantworteten die beiden gestellten Fragebögen mit dem Web-Tool *AttrakDiff* (AttrakDiff, 2023) und *Google Forms*.

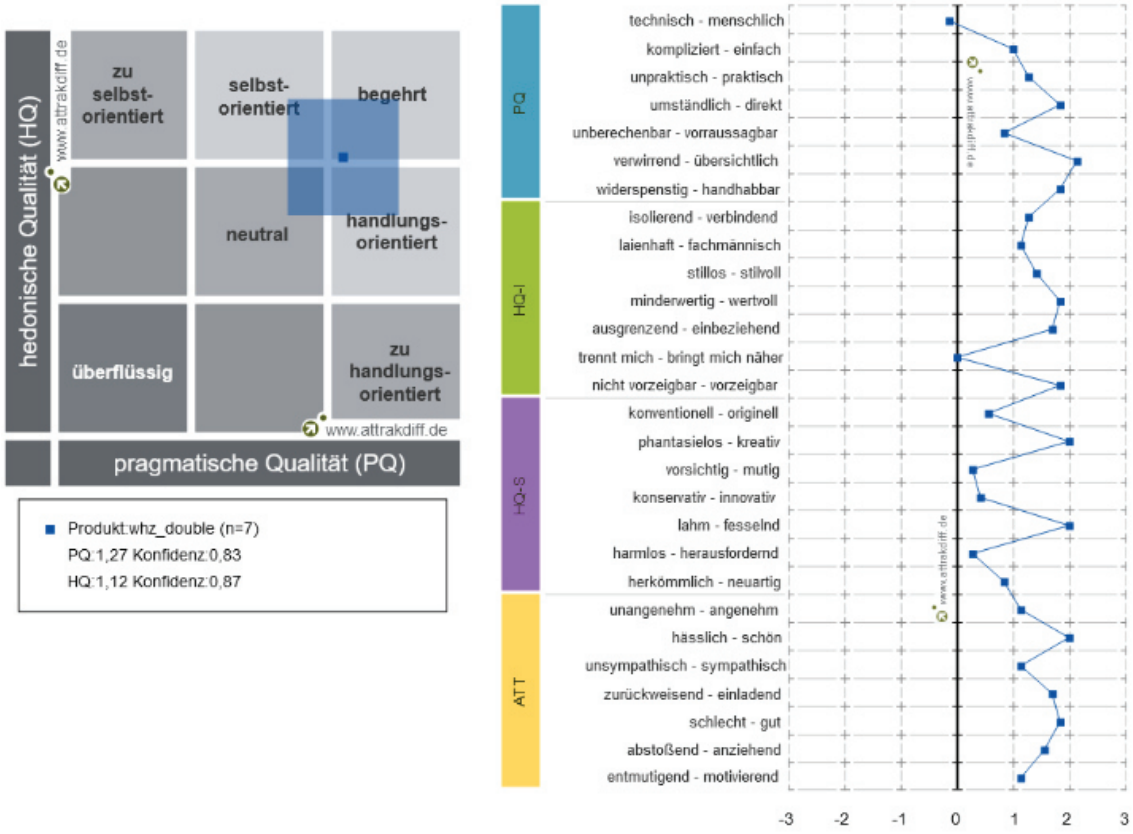

Abbildung 3: Ergebnisse der Evaluation mittels AttrakDif.. Der Prototyp wies eine hohe hedonische und pragmatische Qualität auf, die Probanden empfanden die Interaktion also als positiv.

Die Probanden schätzten außerdem die Umgebung als tauglich für die Anbindung eines maschinellen Lernmodells ein. Zusätzlich ist nach Aussage der Probanden der Prototyp nützlich für Lehrzwecke im Bereich maschinelles Lernens. Auch diese Untersuchung ist demnach positiv beschieden. Somit sollte es möglich sein, eine fertige Version des Prototypen für digitales Lernen im Zusammenhang mit KI zu empfehlen, sowie für das Training eines maschinellen Lernmodells zu nutzen.

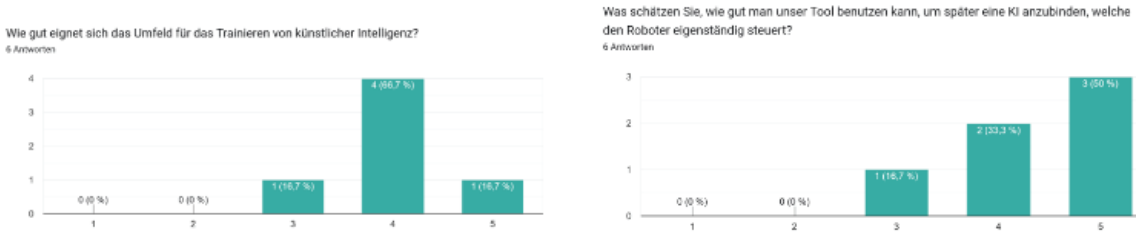

Abbildung 4: Ausschnitt der Ergebnisse der Evaluation aus Google Forms.
Links: Wie gut eignet sich das Umfeld für das Training von KI?
Rechts: Wie gut kann man eine KI an das Tool anbinden? Bewertung: 1-sehr schlecht 5-sehr gut

7 Fazit

Im Rahmen des Lehrkonzepts im Modul *Wissenschaft und Wirtschaft* von (Ritter et al. 2019) al.\cite{Finanzmars} enstand eine projektorientierte Zusammenarbeit zwischen der Hochschule Mittweida und der Data Science Research Group der Westsächsischen Hochschule Zwickau.

Dabei erfolgte die digitale Nachbidlung der Räumlichkeiten der DSRG in der Unity Engine. Dazu enstand eine Programmierschnittstelle auf der Basis des d3-sdk, um eine Kommunikation zwischen dem *Double* Roboter und der Simulation/Webanwendung zu verwirklichen. Eine Evaluation des Prototypen zur Überprüfung, ob sich die Echtzeitsimulation für digitales Lernen im Zusammenhang mit KI, sowie für das Training eines maschinellen Lernmodells eignet wurde durchgeführt und bestätigte eben diese These.

Die Anwendung kann nun auf einem Server der DSRG eingesetzt und dort als Lehrmittel für Studenten zur Verfügung gestellt werden. Außerdem ist die Anbindung eines Machine Learning Modells über die API möglich. Diese kann in der Simulationsumgebung trainiert und später auf den echten Double Roboter übertragen werden.

Quellen

Arayici, Y. & Hamilton, A. (2005). Modeling 3d scanned data to visualize the built environment. In *Ninth International Conference on Information Visualisation (IV'05)*, pages 509–514.

Mahdjoubi, L., Moobela, C., & Laing, R. (2013). Providing real-estate services through the integration of 3d laser scanning and building information modelling. *Computers in Industry*, 64(9):1272–1281.

Ritter, M., Roschke, C. & Tolkmit, V. (2019). Finanzmars im Kosmos von Blended Learning. *Beiträge des CARF Luzern 2019*, pages 327–344.

Rudraraju, N. V. & Boyannapally, V. (2019). *Data quality model for machine learning*. Faculty of Computing, Blekinge Institute of Technology, Karlskrona, Sweden, Master of Science in Software Engineering

Nordhaus, W. D. (2007). Two centuries of productivity growth in computing. *The Journal of Economic History*, 128–159.

Tatarchuk, N. (2009). Advances in real-time rendering in 3d graphics and games.

Wu, X. & Fu, Q. (2022). Science unreal domed screen film-making and application based on unreal engine technology. In *Proceedings of the 3rd Asia-Pacific Conference on Image Processing, Electronics and Computers, IPEC '22*, page 206–210, New York, NY, USA. Association for Computing Machinery.

Attrakdiff. https://attrakdiff.de/. Abgerufen am 2023-01-28.

Autodesk CAD. https://www.autodesk.de/solutions/cad-software/. Abgerufen am: 2022-12-15.

Blender. https://www.blender.org//. Abgerufen am: 2022-12-15.

d3-sdk. https://github.com/doublerobotics/d3-sdk. Abgerufen am: 2023-02-27.

Double Robotics. https://www.doublerobotics.com/. Abgerufen am: 2022-12-15.

Saxony5 Projekt. https://saxony5.de/. Abgerufen am: 2023-02-27.

Substance painter.https://www.adobe.com/products/substance3d-painter.html. Abgerufen am: 2023-02-27.

Unity. https://unity.com/de. Abgerufen am: 2022-12-15.

Unity Documentation, Baked Light. https://docs.unity3d.com/Manual/LightMode-Baked.html. Abgerufen am: 2022-12-15.