HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES

Entwicklung einer Administratoroberfläche für Lehrveranstaltungen mit Photovoltaik-Anlagen in der virtuellen Realität (Kurztitel: Leitstand für die Lehre in VR)

Masterarbeit

im Masterstudiengang Informatik
der Hochschule Ruhr West

Sebastian Wientzek
10002572

Erstprüfer:	Prof. Dr. Marcus Rehm
Zweitprüfer:	Prof. Dr. Gordon Müller

Bottrop, August 2022

Kurzfassung

Diese Arbeit beschäftigt sich mit der Erstellung einer Administratoroberfläche für die Lehre bei Photovoltaik (PV)-Praktika in der virtuellen Realität (VR). Die erstellte Umgebung bietet, mittels Bildschirmspiegelungen, Möglichkeiten zur didaktischen Anleitung und Unterstützung der Studierenden. Das Thema wurde aufgrund einer bestehenden Lehranwendung in der VR bedeutungsvoll und zeigt deutliches Potenzial. Diese Lehranwendung wird bereits umfassend und verpflichtend in den Praktika eingesetzt. Sie bietet einen praxisnahen Aufbau von Solaranlagen und erhöht gefahrlos die Experimentierfreudigkeit. Mit ihr lassen sich die aufgebauten Anlagen technisch prüfen, simulieren und bewerten. Zudem werden die beiden Möglichkeiten zur Unterstützung der Studierenden beurteilt. Als Ergebnis wird die Umsetzung der nahezu automatisierten Administratoroberfläche verdeutlicht und ein Usability-Test aus den Praktika evaluiert.

Schlagwörter: Administratoroberfläche, Bildschirmspiegelung, C, Didaktik, immersiv, Oculus Quest 2, Photovoltaik, Python, Tkinter, virtuelle Realität

Abstract

This work focuses on the creation of an administrator interface for teaching photovoltaic (PV) practicals in the Virtual Reality (VR). The created environment provides the possibilities for didactic guidance and support of the students by means of screen mirroring. The topic became significant due to an existing teaching application in VR and shows clear potential. The teaching application is already extensively and mandatory used in the internships. It provides a hands-on construction of solar systems and safely increases the willingness to experiment. It can be used to test, simulate and evaluate the constructed systems. In addition, the two existing possibilities for supporting the students are evaluated up on this basis. As a result, the implementation of the almost automated administrator interface is clarified and a usability test from the practical courses is evaluated.

Keywords: administrator interface, C, didactic, immersive, Oculus Quest 2, photovoltaics, Python, screen mirroring, Tkinter, virtual reality

Danksagung

Ich bedanke mich an dieser Stelle bei allen Personen, die mich während der Fertigstellung der Masterarbeit unterstützt und motiviert haben.

Allererst gebührt mein Dank Herrn Prof. Dr. Marcus Rehm für die Betreuung der Arbeit und Begutachtung als Erstgutachter.

Ich bedanke mich bei Herrn Prof. Dr. Gordon Müller für die Betreuung und Begutachtung als Zweitprüfer.

Ein großer Dank geht an Herrn M. Sc. Eduard Graf, der mich als Kollege während des Studiums und bei der Nebentätigkeit mit seinem Wissen und seiner Unterstützung zur Seite stand.

Ein besonderes Dankeschön geht an alle Teilnehmer und Teilnehmerinnen meiner Befragung in den Praktika. Ohne sie hätte die Evaluierung in der Form nicht durchgeführt werden können. Ich bedanke mich für ihre Teilnahme- und Informationsbereitschaft, sowie die interessanten und hilfreichen Kommentare und Antworten auf meine Fragen.

Abschließend möchte ich mich bei meiner Familie für die Unterstützung, den Rückhalt und die Korrekturlesungen während des Studiums bedanken.

Bottrop, den 22.08.2022

Sebastian Wientzek

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

2D	Zweidimensional
3D	Dreidimensional
ADB	Schnittstelle ‚Android Debug Bridge‘
AR	Augmented Reality
AVI-GeBe	Projekt ‚AR, VR und IoT im Gebäudebetrieb 4.0‘
COMPLETE	Projekt ‚Collaborative spaces for Online-Meets-Physical LEarning and TEaching‘
GUI	Grafische Benutzeroberfläche (Graphical User Interface)
HRW	Hochschule Ruhr West
IDE	Integrierte Entwicklungsumgebung (Integrated Development Environment)
IoT	Internet der Dinge (Internet of Things)
LAN	Lokales Netzwerk (Local Area Network)
PUN	Photon Unity Networking
VR	Virtuelle Realität
WLAN	Wireless LAN

1 Einleitung

Die Vorteile der Virtuellen Realität (VR) wurden als moderne und weitreichende Technologie mit viel Potenzial bereits an der Hochschule Ruhr West (HRW) erkannt. Diverse interne Forschungs- und Entwicklungsprojekte wurden bereits in der VR bearbeitet und umgesetzt. Die vorliegende Arbeit basiert auf einer Lehranwendung in der VR. Die Entwicklung dieser Anwendung wurde mit dem Projekt *Augmented Reality (AR), VR und Internet of Things (IoT) im Gebäudebetrieb 4.0 (AVI-GeBe)* begonnen und befindet es sich aktuell mit dem Projekt *Collaborative spaces for Online-Meets-Physical LEarning and TEaching (COMPLETE)* in der Weiterentwicklung.

Im Folgenden werden einleitend die Hintergründe, sowie die Problemstellung zu der Lehranwendung in VR und bei der Verwendung dessen dargelegt. Dann wird die Motivation für die Erstellung einer Administratoroberfläche und dessen konkrete Zielsetzung genannt. Abgeschlossen wird die Einleitung mit einer klaren Abgrenzung zu der vorhandenen VR-Anwendung.

1.1 Hintergrund und Problemstellung

Das Potenzial der VR besteht unter anderem in einer kontrollierte Versuchsumgebung, Möglichkeiten für virtuelle Lehrgänge oder Kurse. Auch komplexe, gefährliche technische Verfahren, sowie kostenaufwendige, zeit- und arbeitsintensive Maßnahmen können mit ihr preiswert, sicher und unkompliziert dargestellt werden.

Aus diesen Gründen wurde für das Photovoltaik (PV)-Praktikum eine Anwendung in VR umgesetzt. Der genaue Aufbau und die einzelnen Funktionen können in dem Paper und den angegebenen wissenschaftlichen Arbeiten zu den Simulationen [1, 2] nachgelesen werden. Die VR-Anwendung wird nachfolgend kurz beschrieben und wurde bereits nach dem ersten Einsatz in einem Praktikum evaluiert [3].

Die Lernanwendung ermöglicht es den Studierenden, komplexe und schwer zugängliche Lernobjekte und Inhalte der PV-Anlangen zu erforschen. Hier können verschiedene Konfigurationen mit gängigen PV-Modultypen und unterschiedlicher String-Verschaltungen in den Praktika verwendet werden, um die Unterschiede zu ermitteln und ein Verständnis für die Auslegung von PV-Anlagen zu erhalten.

Dazu ist eine Vorauswahl an Modultypen in einer Datenbank abgespeichert und kann bei der Erstellung einer neuen PV-Konfiguration oder beim Austausch von Modulen aus dem Katalog in VR ausgewählt werden.

Der Vorteil dieser Implementierung besteht darin, dass die technischen Auswirkungen verschiedener handelsüblicher Modultypen in Echtzeit betrachtet und die Auswirkungen auf die Energieerträge über die Simulationen kurzfristig analysiert werden können.

Ohne VR waren diese Analysen bisher nur mit Simulationswerkzeugen in 2D möglich.

In der Anwendung werden 3D-Objekte des Daches und der Solaranlage der HRW in Bottrop verwendet, die per Fotogrammmetrie erstellt wurden.

Darüber hinaus können die Ausrichtung, die Neigung und die Gruppierung zu den einzelnen Strings sowie die Verschaltung mit den Wechselrichtereingängen flexibel definiert werden. Ein Beispiel für das mögliche Spawnen eines PV-Models auf dem Dach mit einem Azimutwinkel von 180° und dem Neigungswinkel der Aufständerung von 12° ist nachfolgend in Abbildung 1 dargestellt.

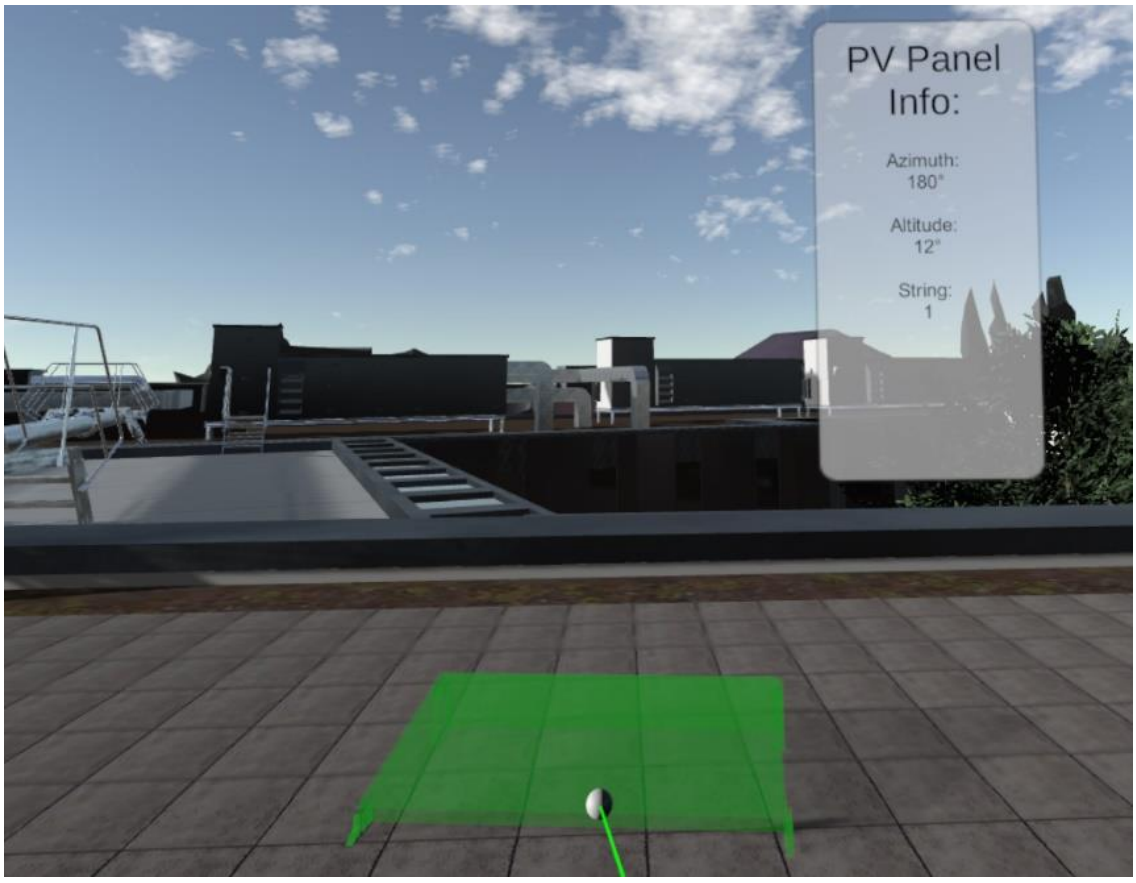

Abbildung 1: PV-Modul, das auf dem Boden mit einem Azimutwinkel von 180° platziert werden soll

Exemplarisch zeigt Abbildung 2 die Standard-Dachbelegung der PV-Anlage der HRW in Bottrop in *Unity*.

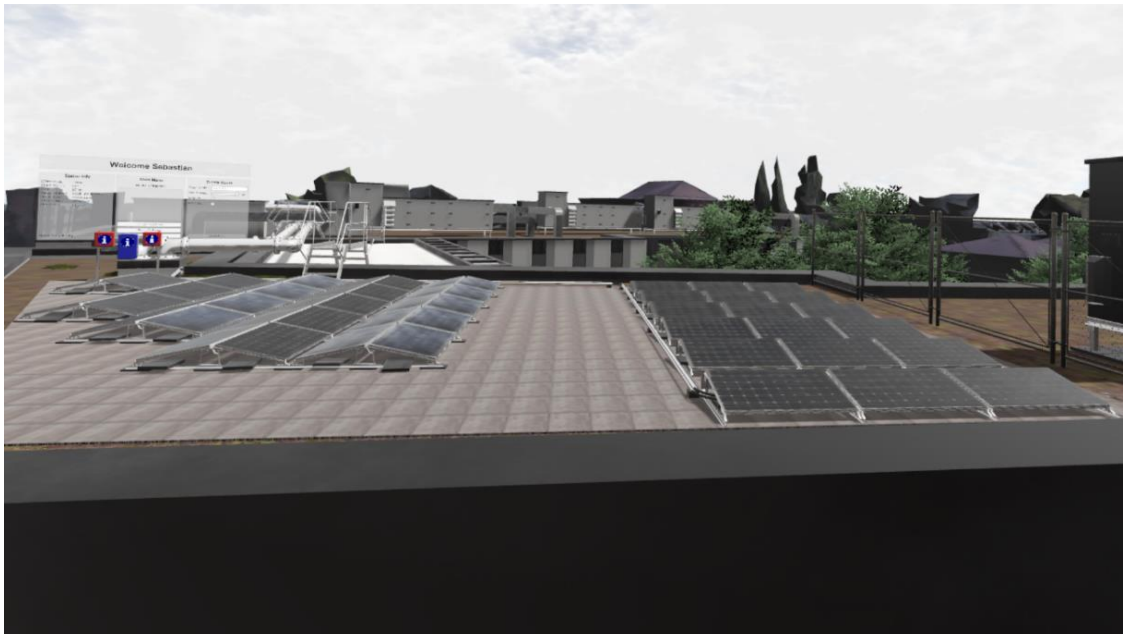

Abbildung 2: Standard-Dachbelegung der aktuellen PV-Anlage der HRW in Bottrop

Die App wird für Ausbildungszwecke und Lehrveranstaltungen eingesetzt, um die Qualität der Lehr- und Studienbedingungen zu optimieren. Außerdem bietet sie den Studierenden die Möglichkeit, neuartige und digitalisierte Lernmethoden zu erkunden. In ihrer ersten Umsetzung soll sie ein Pflichtpraktikum in einem Lehrmodul ersetzen.

In der Anwendung können den Studierenden die Planung, Funktion, technische Umsetzung und Fehlersuche von Solaranlagen vermittelt werden.

Um die Studierenden aktuell in der Anwendung anzuleiten, existieren grob drei Möglichkeiten für die Lehrenden.

Zum einen können sie sich selbst eine VR-Brille aufsetzen und in die Anwendung gelangen. Dadurch können sie die aktuelle PV-Konfiguration einsehen, sowie mit den Studierenden kommunizieren. Dies hat jedoch den Nachteil, dass sie nicht einsehen können, was die Studierenden im Detail gerade unternehmen oder welche Menüpunkte gerade geöffnet sind und ggf. bearbeitet werden.

Eine weitere Möglichkeit zur Unterstützung bei der Anwendung besteht darin, sich die jeweilige Brille der Studierenden aufzusetzen. Diese sind dann ihrerseits aber nicht in der Umgebung und sehen nicht, wie die Lehrenden eingreifen.

Die letzte Variante besteht darin, sich eine VR-Brille per Mirroring auf einem vorhandenen Computer freizugeben und somit einen ausgewählten Bildschirminhalt einzusehen oder zu teilen.

Eine direkte und unkomplizierte Möglichkeit, die Bildschirminhalte aller Studierenden einzusehen, existiert derzeit nicht.

1.2 Motivation und Zielsetzung

Bei den ersten durchgeführten Praktika ist mit der eben geschilderten Lehranwendung aufgefallen, dass die Einrichtung der VR-Brillen und die Unterstützung bei der Bedienung optimiert werden kann. Die Bildschirminhalte der Brillen können nur umständlich über optionale Software, über das Aufsetzen einer Brille und Begleiten der Studierenden in dem virtuellen Raum oder das Aufsetzen bzw. Anschließen der Brille des jeweiligen Studierenden eingesehen werden.

Hierauf aufbauend soll die Masterarbeit eine nützliche Erweiterung zur Anleitung und Darstellung der Aktionen der Studierenden ergeben. Diese soll auch zur Unterstützung bei der Brillen-Einrichtung und Problemstellungen bezüglich der Menüführung eingesetzt werden können.

Das Hauptziel besteht in der Entwicklung einer konstruktiven Umgebung für Praktika in der Lehre, die in der virtuellen Realität durchgeführt werden. Hierbei soll jeder in den einzelnen VR-Billen dargestellte Bildschirminhalt auf einem Computer in dem gleichen Netzwerk gespiegelt werden. Dies hat den Vorteil, dass die Studierenden direkt in ihrer Handlung mit der VR-Brille und der Anwendung geleitet und unterstützt werden können, sodass die Benutzerfreundlichkeit/ Usability erhöht und das Erreichen der Lehrziele optimiert wird.

Die Konzeptidee entsprang externen Bereichen der aufgeteilten Abbildungen von Überwachungskameras und der Darstellung der Anlagenteile in Leitständen in der Energiewirtschaft. Letzteres führte zu dem Kurztitel dieser Arbeit.

1.3 Abgrenzung

Zur bestehenden Umsetzung der Praktikums-Anwendung existiert eine eindeutige Abgrenzung, da bisher die gesamte Entwicklung in *Unity* in der Programmiersprache C# umgesetzt wurde.

Diese Anwendung ist auf dem Computer und auf der *Oculus Quest 1* und *2* [4] einsetzbar. Sie ermöglicht eine dynamische Einrichtung von PV-Modulen. Zusätzlich besteht die Möglichkeit, über einen Mehrspieler-Modus mit dem *Photon Unity Networking (PUN)* Framework mit mehreren Studierenden und dem Lehrenden in der virtuellen Umgebung auf dem Dach zusammen zu arbeiten.

Eine unkomplizierte und praktikable Funktionalität, um alle Bildschirminhalte gleichzeitig einzusehen oder vereinfacht auch nur einen Gezielten, ist bis auf die Lösung mit dem USB-Kabel derzeit nicht gegeben.

1.4 Vorgehensweise

Diese Arbeit ist in Anlehnung an das Wasserfallmodell der Softwareentwicklung untergliedert. Das Modell unterteilt die Phasen der Entwicklungsprozesse in die Anforderungsanalyse, den Architekturentwurf, den detaillierten Entwurf, die Programmierung, die Komponententests, die Systemtests und die Akzeptanz. [5]

Kapitel 2 enthält den Stand der Technik zu Möglichkeiten einer Umsetzung von dreidimensionalen (3D-) Anwendungen in der VR. Dazu werden zwei verschiedenen Engines und die verschiedenen Möglichkeiten der Verwendung betrachtet. Des Weiteren werden die unterstützten Plattformen zur Nutzung der 3D-Anwendungen dargestellt. Ferner wird die verwendete Netzwerkkomponente erläutert. Daraufhin werden die beiden Möglichkeiten zur Unterstützung der Studierenden dargestellt. Als eine der möglichen Optionen zur Unterstützung wird eine Übersicht über die existierenden Spiegelungstools für *Android*-basierte Geräte gegeben. Der Abschnitt enthält darüber hinaus die notwendigen Vorbereitungen zur Kompilierung einer der Spiegelungsanwendungen. Zuletzt enthält der Abschnitt die Beschreibung des Tools zur Versionsverwaltung mit *Git* und der Programmiersprache Python.

In Kapitel 3 wird sich, anlehnend an das Wasserfallmodell, mit der Implementierung der Hauptanwendung (der Administratoroberfläche) beschäftigt. Der Abschnitt beinhaltet die Anforderungsanalyse, Planung der Anwendung, die verwendeten Tools, Module und die Anpassungen an dem vorhandenen Spiegelungstool. Außerdem gehört die Konfiguration der Endgeräte, das Testen der Software und die Auswertung der Code-Qualität als Inhalt dazu. In Kapitel 4 werden Praxisdurchläufe mit der entwickelten Administratoroberfläche evaluiert und dadurch dessen Akzeptanz ermittelt. In Kapitel 5 wird das Resultat der implementierten Anwendung verdeutlicht. Im Kapitel 6 wird das Fazit gezogen. Es beinhaltet eine Zusammenfassung der Arbeit, sowie einen kurzen Ausblick im Hinblick auf mögliche Weiterentwicklungen und weitere mögliche Anwendungszwecke.

2 Stand der Technik

Die Handhabung von PV-Anlagen ist aufgrund des hohen Stroms (bis zu 50 A) und der hohen Spannung (bis zu 1.000 V) ein komplexes Thema [6]. So ist es unter realen Bedingungen schwierig, den richtigen Umgang mit PV-Anlagen zu erlernen und mit ihnen zu experimentieren. Das Hauptproblem ist, dass es sehr kostspielig ist, vor allem in Bezug auf die Anschaffung und die falsche Handhabung. Ein weiterer wichtiger Aspekt ist, dass bei falscher Handhabung Gefahren für Leib oder Leben nicht ausgeschlossen werden können.

Wie von Bailenson et al. beschrieben, macht es die VR günstiger, diese gefährlichen oder teuren Lektionen und Materialien zu simulieren, vor allem, wenn sie dupliziert werden müssen [7].

Zuvor konnten PV-Anlagen nur in Form von kostenpflichtigen Simulationswerkzeugen exemplarisch in 2D aufgebaut und simuliert werden. Bekannte Tools hierfür sind *PVsys* und *PVsol* [8]. Vorhandene 3D-Tools [9, 10] bieten nur eingeschränkte Möglichkeiten und Einblicke in den PV-Anlagenbau sowie in Simulationen. Die VR-Technologie bietet zusätzlich zu den bestehenden 3D-Tools ein immersives Erlebnis, als wäre man tatsächlich vor Ort, und ist außerdem fehlertolerant. Diese Technologie bietet die Möglichkeit, zusätzliche Informationen schnell und einfach darzustellen. Zudem bietet sie eine große Freiheit bei der experimentellen Auslegung einer PV-Anlage, da mögliche Fehler ohne großen Aufwand zurückgesetzt werden können. Die Anwendung ermöglicht den Studierenden in praxisorientierten Aufgabenstellungen optimale PV-Anlagen zu erforschen und dabei den Umgang mit Fehlerfällen zu erlernen. Sie bietet ihnen die Möglichkeit, komplexe Inhalte und ansonsten schwer zugängliche Lernobjekte zu erforschen. [11]

Die VR hat viele Möglichkeiten aufgezeigt, konstruktives Lernen zu unterstützen und die Lernergebnisse der Studierenden zu verbessern, wie von Hellriegel und Cubela beschrieben [12].

Diese Vorteile der VR und die zuvor genannten folgenschweren Nachteile der Umsetzung von realen PV-Anlagen zeigen deutlich, dass es sinnvoll war, die Lehranwendung in der VR zu entwickeln.

Um die Studierenden bei der Einrichtung und Verwendung dieser Lernanwendung zu unterstützen und anzuleiten, ist es notwendig, den Bildschirminhalt dieser einzusehen. Dieses Verhalten wird Bildschirmspiegelung (im englischen „Screen Mirroring“) genannt und ist für die Darstellung im selben Raum

angedacht. Demgegenüber steht das Screen Casting und Screen Sharing. Beim Screen Casting werden dagegen ausgewählte Medien wie z. B. Videos, Bilder oder Musik nicht auf dem Gerät (bspw. Smartphone) abgespielt und angezeigt, sondern direkt an den Fernseher übertragen und erst dort wiedergegeben. Das Screen Sharing hingegen beschreibt einen, der Bildschirmspiegelung, ähnlichen Vorgang. Hier liegt jedoch Unterschied darin, dass der Inhalt an einem anderen Ort angezeigt wird. [13]

2.1 Dreidimensionale Umsetzung

Für die Entwicklung von Spielen bzw. verallgemeinert Anwendungen in der 2D oder 3D Umgebung existieren hauptsächlich zwei Möglichkeiten [14, 15]. Diese sind Unity 3D und die Unreal Engine. Unity 3D hat einen Marktanteil von 48 % und die Unreal Engine liegt bei 13 % [15]. Unreal ist jedoch Unity kaum unterlegen, da sie sich funktional sehr ähneln und sich direkt konkurrieren. Entwickelt einer der beiden eine neue Funktionalität, folgt die Aktualisierung der anderen üblicherweise umgehend [14].

2.1.1 Unity

Unity bietet die Möglichkeit der Entwicklung von 2D- und 3D-Spielen und -Anwendungen für viele verschiedene Plattformen wie auch die virtuelle Realität. Dabei wird es als Game-Engine bzw. Spiel-Engine eingesetzt, dient aber auch als Entwicklerwerkzeug. Diese Engine erfüllt an Systemen alle Voraussetzungen „wie Grafik, Physik, Audio, Steuerung und Skripting“. Auch die Netzwerkunterstützung ist inkludiert, eine Funktionalität zum Abspeichern von Spielständen fehlt jedoch. Die Arbeit mit Unity erfolgt hauptsachlich im Unity Editor, ist demnach ein Autorensystem. Dieses ermöglicht es, alle Teile für Spiele in Szenen zusammenzustellen und damit komplette Spiele oder interaktive Anwendungen zu erstellen, denn mit Unity lassen sich mehr als nur Spiele entwickeln. Hier steht Unity für die sehr reaktionsschnelle und einfache Bedienbarkeit der Benutzeroberfläche. Das macht sie besonders anfängerfreundlich und führt in dem Fall zu einem geringeren Aufwand [15]. [16]

Für eine schnelle Prototyp-Entwicklung kann in Unity die Methode des nodebasierten Skriptings, das sogenannte Unity Visual Scripting (ehemalig Bolt), angewendet werden [17]. Dabei muss kein Code geschrieben werden. Alternativ dazu werden die Programmiersprachen JavaScript und C# unterstützt.
In dem Asset Store von Unity sind eine große Anzahl „fertige[r] 3D-Modelle von Charakteren, Texturen, Umgebungen, Sounds und Partikelsystemen“ erhältlich. Weiterhin reichen sie von „Animations- und [Graphical User Interface (GUI)]-

Generatoren über Erweiterungen für die KI-Steuerung bis hin zum […] Framework zur Erstellung von [Role-Playing-Games (RPGs)]“ werden viele Bereiche abgedeckt.
Ferner ist es bei der Unterstützung verschiedener Plattformen sehr vielseitig. Unterstützt werden hier: „iOS, Android, Windows Phone 8, Tizen, Android TV, Samsung Smart TV, Xbox One & 360, Windows PC, Mac OS X, Linux, Web player, WebGL, VR (inkl. Hololens), SteamOS, PS4, Playstation Vita und Wii U.“ [15]

2.1.2 Unreal Engine im Vergleich zu Unity

Vergleichsweise ist die *Unreal Engine* ebenso für die 2D- und 3D-Entwicklung vorgesehen. Hier erscheint die Benutzeroberfläche sehr überfüllt und komplex, obwohl bereits viele Optimierungen an ihr vorgenommen wurden. In vielen Bereichen ist die Verwendung von Unreal Zeitaufwendiger als Unity, somit liegt das Tool in der Benutzerfreundlichkeit weiterhin hinter Unity.
Unity und Unreal verwenden beide ein ähnliches visuelles Scripting, mit dem Unterschied, dass es sich bei letzterem jedoch Blueprints Visual Scripting nennt. Zudem wird die Programmierung in der Programmiersprache C++ unterstützt und ist daher komplexer als in Unity, somit weniger anfängerfreundlich.
Der Asset Store von Unreal enthält viele Assets, ähnlich wie bei Unity, jedoch im Vergleich wesentlich weniger.
Unreal unterstützt mit „iOS, Android, VR, Linux, Windows PC, Mac OS X, SteamOS, HTML 5, Xbox One und PS4“ ähnliche Plattformen wie Unity. [15]

2.2 Virtuelle Realität

Mit Unity ist es möglich, neben einfachen 3D-Anwendungen auch Programme in der virtuellen Realität zu erstellen.

Hierbei können zwei Unterteilungen vorgenommen werden. Zum einen die Entwicklung für den Windows-Rechner oder zum anderen autark für das *Android*-System, insbesondere die VR-Headsets *Oculus Quest 1 und 2*. Es müssen jeweils gewisse Konfigurationen beachtet werden, in der Benutzeroberfläche von Unity bei der Erstellung eines Projektes, innerhalb des Projektes und in den Build-Settings.

Beim Vergleichen der beiden Anwendungsarten muss man differenzieren, da die virtuelle Realität auf dem Computer für „technisch und grafisch aufwendigere Spiele und Apps“ verwendet wird [18].

Die mobilen *Android*-Geräte besitzen dagegen eine schwächere Hardware (die Quest 2 besitzt nicht mehr Leistung als ein neues Smartphone). Daher muss bei

mobilen Anwendungen mit einer verminderten Grafikqualität und Leistung gerechnet werden. [19]

Für konkrete Leistungsbewertungen werden Benchmarks eingesetzt. Aufgrund der verschiedenen Systemarchitekturen (*ARM* als mobile Prozessorarchitektur bei *Android* und *x64* bzw. *x86-64* bei *Windows*-Rechnern) werden hier architektur- und plattformübergreifende Benchmarks benötigt. Diese wurden teilweise von Heaney ausgeführt und beschrieben. Er beschreibt, dass die *Oculus Quest 2* in einem derartigen Benchmark *GeekBench 5* für die Prozessoren eine Multicore Leistung von 1.599 Punkten aufweist. Als zeitgemäßer Vergleich hat ein modernes Smartphone, wie das *Samsung Galaxy S22 Ultra*, eine Multicore Leistung von 3.433 Punkten [20]. Der dortigen Datenbank entsprechend, kommt ein aktueller *Intel Core i9-12900K* auf 17 299 Punkte [21]. Bei Annahme des Intel Desktoprozessors als Grundwert erreicht das Samsung Smartphone lediglich 20 % dessen Leistung. Die im Vergleich dazu etwas ältere Quest 2 liegt bei 9 %.
Für einen Vergleich der Grafikleistungen sind über einen ähnlichen Benchmark *GFXBench Aztec High Tier Offscreen* die folgenden Werte zu verzeichnen. Nach Heaney erreicht die *Quest 2* in einem dortigen Durchlauf etwa 1.142 Bilder. Den Daten des Benchmark Betreibers zufolge kommt das *Samsung Galaxy S22 5G* auf 2.014 Bilder [22] und eine *NVIDIA GeForce RTX 3080* auf 24.677 Bilder [23]. Hier verhalten sich die mobilen Geräte ähnlich leistungsschwach wie bei dem vorherigen Benchmark. Das Smartphone erreicht etwa 8 % der Leistung der Desktop-Grafikkarte und die *Quest 2* hingegen 5 %. [24]

Aufgrund der geschilderten geringen Leistung müssen bestimmte Aspekte in der Projekterstellung und -bearbeitung berücksichtigt und optimiert werden. Es wird daher empfohlen, das Windows-Projekt zu duplizieren und anschließend für die Quest zu optimieren. Zu diesen Feinabstimmungen der virtuellen Welten gehört beispielsweise das Backen der Beleuchtung („baking lighting“). Hierbei werden die komplexen Berechnungen für die Beleuchtung, einschließlich der Schatten im Unity Editor vorab ausgeführt und in sogenannten *Lightmaps* abgespeichert [25]. Wird die Anwendung gestartet, werden diese Daten geladen und dadurch der Aufwand für die Berechnungen während der Verwendung verringert. Weitere Optimierungen liegen im Mindern der geometrischen Komplexität der Welten. Dabei sollten Texturen so klein wie möglich gehalten werden. Weiterhin ist es sinnvoll, Transparenz zu vermeiden. Falls ein Avatar verwendet wird, müssen dort unnötige Elemente, sowie Knochen entfernt werden und dieser vom geometrischen Aufbau so einfach wie möglich gehalten werden. Auch dort ist Transparenz zu meiden und die Texturgrößen sind zu minimieren. Es ist nicht optimal, zwei inhaltlich gleiche, aber voneinander unabhängig optimierte Projekte für den

PC und die Quest zu nutzen. Der Vorgang bedarf Übung und Wiederholung. Er ermöglicht jedoch eine bestmögliche Kontrolle und Optimierbarkeit für die jeweilige Plattform. Möglicherweise ergibt sich in Zukunft eine Art und Weise, dieses Verhalten zu verbessern. Die Notwendigkeit dieses Vorgangs führt jedoch darauf zurück, wie Unity die Projekte strukturiert. [26]

Für die Kompilierung (das Build) der Anwendung ist in Unity für beide Plattformen eine ähnliche Vorgehensweise gegeben. Über die Oberfläche kann im Hauptmenü unter Datei in den Build-Einstellungen die Zielplattform flexibel ausgewählt werden. Dies kann zu einer zeitlich aufwendigen Veränderung der Konfiguration bzw. des Aufbaus des Projektes führen. Aufgrund der empfohlenen Optimierungen zwischen den beiden Geräten ist es jedoch zu empfehlen, beide Projekte getrennt bei den Zielplattformen zu belassen um die maximalen Leistungen und grafischen Qualitäten der Anwendungen zu erhalten.
Mit der Funktion Build wird bei beiden Plattformen ein lauffähiges Paket erstellt. Bei Windows enthält es eine ausführbare *Executable (EXE)*-Datei. Für *Android* wird eine Android-Paket-Datei (APK-Datei) erzeugt, die im Nachhinein beispielsweise mit SideQuest installiert werden kann. Alternativ dazu kann mit „Build And Run“ direkt das Paket erstellt und entweder auf dem PC ausgeführt (Windows-Version)
oder die Android-Version auf ein angeschlossenes, Android-basierendes Endgerät, wie die VR-Brillen, übertragen werden. [27]

Um die Anwendung zu testen, kann sie unter Windows live ausgeführt werden. Wenn eine *Oculus* VR-Brille verbunden ist, wird die Anwendung direkt auf die Brille gestreamt. Dies ermöglicht es, alle Funktionalitäten mit der VR-Brille über den PC auszuführen und zu testen. Für die endgültigen Tests unter Android muss sie erst installiert werden oder über die zuvor genannte Funktion „Build and Run“ automatisch übertragen und gestartet werden. Danach lassen sich auch ohne einen PC alle Funktionalitäten auf der VR-Brille ausführen und testen.

Unter Windows werden Informationen, Warnungen und Fehler im Live-Modus direkt auf der Oberfläche ausgegeben. Im besonderen Fehlerfall, der nur während des Betriebes der Android-Version auftritt, lassen sich diese Meldungen live über USB aus dem Systemspeicher der Geräte auslesen. Dazu wird die Kommandozeile benötigt um mit ADB den Befehl *adb logcat -c* auszuführen. Der Parameter *clear* (-c) leert den Speicher vor der Ausgabe, weshalb diese Kombination vor dem Start der Anwendung ausgeführt werden sollte. Bei einer Fehlersuche im Nachgang kann er ausgelassen werden. [28]

2.3 Netzwerkkomponente

Für *Unity* gibt es mehrere verschiedene Netzwerkkomponenten wie beispielsweise *Mirror* und *PUN*.

PUN ist ein Paket für Unity zur Erstellung von Multiplayer-Spielen. Flexible Mehrspielerräume bringen über das Netzwerk Spieler zusammen, in denen Objekte synchronisiert werden können. Einige der Funktionen sind *Remote Procedure Calls (RPCs), Custom Properties* oder „'low level' Photon Events". Die schnelle und (optional) zuverlässige Kommunikation erfolgt über eigene, dedizierte Photon-Server. Daher müssen sich die Clients nicht einzeln untereinander verbinden. *PUN* unterstützt im Grunde alle von Unity unterstützten Plattformen und wird in zwei Optionen angeboten. „PUN FREE" als kostenlose Variante mit vielen Demos, Beispielskripts und Dokumentation oder „PUN PLUS" als kostenpflichtige Variante. Dort ist der gleiche Inhalt wie bei „PUN FREE" enthalten, zusätzlich jedoch ein Paket mit bis zu 100 gleichzeitigen Nutzenden für die Photon Cloud. Diese beiden Varianten lassen nur maximal 16 Anwendende pro Raum zu, die kostenfreie Version jedoch eine Anzahl von 20 gleichzeitigen Anwendenden [29]. „PUN PLUS" ermöglicht dagegen bis zu 100 gleichzeitig verbundene Anwendende. [30]

Aufgrund vorheriger Erfahrungen mit *PUN* und aus dem Grunde, dass die kostenlosen Variante für die geplanten gleichzeitigen fünf bis zehn Benutzer pro Raum bzw. verteilt auf zwei Räume ausreicht, wurde in der VR-Anwendung auch dieses Paket verwendet.

2.4 Möglichkeiten zum Teilen der Interaktion

Es existieren zwei verschiedene Möglichkeiten zum Teilen der Interaktion mit anderen Studierenden oder den Lehrenden.

Eine Option wäre es, die Objekte und Handlungen mit den Teilnehmenden in der Software in Unity zu teilen. Alternativ dazu können die Bildschirme der Android-basierten Geräte gestreamt werden.

2.4.1 Teilen der Interaktion in der PV-Anwendung

Eine Möglichkeit zum Teilen der Objekte und Handlungen mit den Teilnehmenden besteht darin, sie softwareseitig in Unity über das Netzwerkpaket *Photon* zu implementieren. Dies würde zu einem hohen Programmier-, Konfigurier- und Wartungsaufwand führen und könnte in einer hohen Netzwerkbelastung enden,

weil viele Objekte und Bewegungen zwischen allen Anwendenden synchronisiert werden müssten.

2.4.2 Bildschirmspiegelung multipler Android-Geräte

Alternativ dazu ist es möglich, die Bildschirminhalte mehrerer Android-basierter Geräte (*Oculus Quest und Oculus Quest 2*) mittels verschiedener existierender Tools zu teilen. Diese Option ist wesentlich einfacher, schneller und fehlerfreier umzusetzen.

Diese Option ist in zweierlei Hinsicht universell einsetzbar. Durch die Anforderung des Android Betriebssystems kann sie einerseits mit verschiedenen Android-basierten Geräten, wie die *Oculus Quest* VR-Brillen, aber auch beispielsweise mit Smartphones verwendet werden. Andererseits eignet sie sich auch zur Unterstützung außerhalb der PV-Anwendung.

2.5 Tools für die Bildschirmspiegelung Android-basierter Geräte

Für die Durchführung der Bildschirmspiegelung und Erstellung der Administratoroberfläche müssen diverse Systeme betrachtet werden. Hierzu zählen als Betriebssysteme Android auf der Oculus Quest 1 und 2, sowie Windows auf den Computern. Als Verbindungswege sind die USB-Verbindung zur Konfiguration der VR-Brillen und zur Datenübertragung die Wireless Lan (WLAN) Verbindung von Bedeutung. Zur Übertragung der Bildschirminhalte können verschiedene Tools verwendet werden. Diese müssen verglichen und die Verwendbarkeit evaluiert werden.

Um im Abschluss weitere Funktionalitäten wie die Umschaltung eines Bildschirms in den Vollbildmodus zu ermöglichen, muss entweder eines der vorhandenen Tools weiterentwickelt oder ein zusätzliches Programm entwickelt werden. Zu Variante eins ist es für die Weiterentwicklung jedoch notwendig, dass das jeweilige Tool vom Quellcode her offen und verfügbar ist (Open Source Software).

Alternativ dazu müsste bei der zweiten Variante ein Zusatz entwickelt werden, der die Bedienoberfläche über ein zusätzliches Overlay (überlagertes Menü) darstellt und die Unterprogramme bedient.

Die weitere Vorgehensweise wird anhand des vorliegenden Standes der Technik festgelegt.

Standardmäßig sind die vorhandenen Bildschirmspiegelungstools für Android-basierte Geräte nur auf eine einzelne Verbindung ausgelegt. Daher ist die Verwendung mit nur einer Brille möglich (*Oculus Casting* [31, 32] oder *SideQuest mirroring*, verwendet *scrcpy*). Alternativ können jedoch manuell auch mehrere Android-basierte Geräte gleichzeitig gespiegelt werden, beispielsweise mit *ApowerMirror*, *TeamViewer*, *Vysor* oder *scrcpy*. Die Möglichkeiten und Einschränkungen der einzelnen Tools werden nachfolgend näher untersucht und dargelegt.

2.5.1 ApowerMirror

ApowerMirror ist eine App zur Spiegelung der Bildschirme von iOS- und Android-Geräten. Sie ermöglicht zudem die Audio-Übertragung und verwendet die „Mainstream-Spiegelungstechnologie“. Die Geräte können über USB oder WiFi auf den PC gespiegelt und in Echtzeit auf einen Fernseher übertragen werden. Auf den PC können Benutzer 4 Geräte gleichzeitig spiegeln. Die Geräte können vom PC aus mit der Maus und Tastatur gesteuert werden. Auch mobile Spiele und Anwendungen sind so ohne Emulatoren auf dem PC lauffähig. Zudem enthält das Tool in der kostenlosen Version ein Wasserzeichen und bietet einen Vollbildmodus. In dieser Version können jedoch nur zehn Minuten gespiegelt werden. Diese Einschränkungen sind dagegen bei der kostenpflichtigen Version für beispielsweise einmalig 59,95 € [33] nicht enthalten. Weiterhin lassen sich mit ihr aus dem hinzugefügten seitlichen Menü (einer Sidebar) einfach Screenshots erstellen. Zudem müssen die Bildschirmspiegelungen manuell auf dem Bildschirm verteilt und ausgerichtet. [34]

2.5.2 Scrcpy

Das Tool „screen copy“ [35] (abgekürzt und zusammengefasst *scrcpy*) bietet die Möglichkeit, Android-basierte Geräte zu spiegeln, also den gesamten Bildschirminhalt anzuzeigen. Darüber hinaus ist eine Steuerung der Android-Geräte möglich. Diese Funktionalität ist jedoch bei den Oculus Quest VR-Brillen nicht gegeben.

Die Datenübertragung von *scrcpy* erfolgt entweder über USB oder über TCP/IP. Aufgrund des Open-Source-Codes und der Tatsache, dass keine Eingriffe in das Betriebssystem benötigt werden (ein sogenannter Root-Zugriff), bietet es ideale Voraussetzungen für die weitere Verwendung.

Das Tool besitzt diverse Vorteile und Funktionen, jedoch auch geringfügige Nachteile. Zu diesen zählt, dass es manuell bei jedem Aufruf nur ein Gerät spiegelt, es keine eigene Benutzeroberfläche besitzt, sondern die Kommandozeile

mittels vieler verschiedener Argumente zur vielseitigen Konfiguration verwendet. Dagegen überwiegen die vielen Vorteile. Zu diesen gehören, dass es bis auf die fehlenden Benutzeroberfläche und Verwendung der Kommandozeile relativ unkompliziert in der Einrichtung ist, es sehr vielseitig konfiguriert werden kann, eine gute Leistung der Bildrate zwischen 30 bis 120 Frames-Per-Seconds (FPS) ermöglicht, hohe Qualitäten mit 1920 x 1080 Pixeln bietet und eine geringe Latenzzeit von etwa 35 bis 70 ms aufweist. Der größte Vorteil ist, dass es sich um eine kostenlose Open Source Software handelt und der Quelltext somit frei verfügbar, editierbar und weiterverwendbar ist. Er bietet demnach die beste Grundlage und am meisten Entwicklungspotenzial.

Zusätzliche Features sind die Möglichkeit der Videoaufzeichnung, des Kopierens und Einfügens, die Flexibilität in der Konfigurierbarkeit, Verwendbarkeit der Geräte als Webcam und physische Simulationen von Eingabegeräten wie der Tastatur und Maus. Auch eine Verwendung als simulierte physische Tastatur und Maus ist möglich, dies wird dort als *On-The-Go (OTG)* Modus bezeichnet.

Scrcpy selbst setzt jedoch ein weiteres Tool voraus, die *Android-Debug-Bridge (ADB).* Sie ist durch die Installation von *Unity* bereits auf dem System vorhanden.

Innerhalb von *scrcpy* gibt es hauptsächlich zwei Bestandteile, den in Java implementierten Server und den, in der Programmiersprache C, entwickelten Client.

Der Server verwendet 3 verschiedene Prozesse:

- der Hauptprozess kodiert das Video und streamt es an den Client,
- der Controller-Thread wartet auf Steuermeldungen (typischerweise Tastatur- und Mausereignisse) vom Client,
- der Empfänger-Thread (wird vom Controller verwaltet) sendet Gerätemeldungen an die Clients (wird derzeit nur zum Senden des Inhalts der Zwischenablage des Geräts verwendet).

Der Client dagegen verwendet 4 Prozesse:

- der Haupt-Thread führt die Grafik-Ereignisschleife aus (basierend auf dem *Simple Directmedia Layer (SDL)*),
- der Stream-Thread empfängt das Video und wird für die Dekodierung und Aufnahme verwendet,
- der Controller-Thread sendet die Steuermeldungen an den Server,
- der Receiver-Thread (wird vom Controller verwaltet) empfängt Gerätemeldungen vom Server.

2.5.2.1 Kompilierung von scrcpy als lokale Installation

Für die Kompilierung der Anwendung *scrcpy* werden verschiedene Wege angeboten und im Folgenden erläutert, die Abhängig vom Betriebssystem sind [36].

Unter Windows wird eine Kross-Kompilierung über Linux durchgeführt. Hierfür wird die Toolsammlung *MSYS2* benötigt. Diese enthält den Paketmanager *pacman* und *MinGW*, die native Windows-Portierung der *GNU COMPILER COLLECTION (GCC)*.

In dem Terminal *MSYS2 MinGW x64* werden weitere Pakete für die Laufzeitumgebung benötigt. Hierzu zählen die Grafik-Entwicklungsbibliothek *SDL2*, der Videokonverter *FFmpeg* und die Programmbibliothek für den Zugriff auf USB-Geräte, *libusb*.

Zur Kompilierung der finalen, ausführbaren Anwendung werden außerdem die folgenden Bibliotheken benötigt. Zur Pflege von Programm-Abhängigkeiten „Make“, zur Versionsverwaltung installierter Programmbibliotheken „pkg-config“, die Compiler *GCC* und das neuartige Build-System *Meson* [36].

Meson basiert auf der Programmiersprache Python und setzt somit die Installation von Python 3 voraus. Zudem verwendet es im Backend das kleine Build-System Ninja, das den Fokus auf schnelle Kompilierungen legt. [36]

Zur endgültigen Ausführung der Kompilierung und dem Starten des Tools wurde folgendes Bash-Skript entwickelt (Abbildung 3). Dieses führt zu einem Wechsel des Verzeichnisses in den Entwicklungspfad von *scrcpy* und führt daraufhin das Shell-Skript *install_release.sh* zum Bauen und Installieren der Spiegelungsanwendung aus. Diese wird im Nachgang in Zeile 4 bei erfolgreichem Build direkt ausgeführt. Im folgenden Quellcode und in Pfadangaben ist die Angabe „BENUTZERNAME“ an den jeweiligen Benutzernamen anzupassen.

```
1  #!/bin/bash
2  ScrcpyDevPath="C:/Users/BENUTZERNAME/Documents/GitHub/scrcpy_VSC_MinGW64"
3  cd $ScrcpyDevPath
4  ./install_release.sh && $ScrcpyDevPath/build-auto/app/scrcpy.exe
5  read -p "Press [Enter] key to continue..."
```

Abbildung 3: Bash-Skript zur Kompilierung von scrcpy

Dieses Skript ist im Verzeichnis *scrcpy-edit* unter *build* abgelegt und wird über eine Verknüpfung mit dem Ziel „C:\msys64\mingw64.exe C:\Users\BENUTZERNAME\Documents\GitHub\scrcpy_VSC_MinGW64\build\buildScrcpyVSCMingGW.sh“ aufgerufen.

Die Verknüpfung startet somit das eben beschriebene Bash-Skript mit der benötigten Anwendung und Umgebung *MSYS2 MinGW x64*. Dies ist eine relevante Festlegung, da in der Build-Beschreibung von *scrcpy* keine nähere Festlegung

getroffen wurde, aber nach Installation von *MSYS2* und *Git* mehrere Umgebungen bzw. Subsysteme verfügbar sind.

Zu diesen Subsystemen zählen die folgenden fünf: *MSYS 2 MinGW Clang x64, MSYS2 MinGW UCRT x64, MSYS2 MinGW x64, MSYS2 MinGW x86 und MSYS*. Durch *Git* ist zusätzlich die Umgebung *Git Bash* vorhanden. Diese ist für die weitere Vorgehensweise jedoch unbrauchbar, da sie den notwendigen Paketmanager *pacman* nicht enthält. Da für die Kompilierung von *scrcpy* die Compilersammlung *GCC* und eine 64-Bit Version benötigt wird, fallen die beiden Umgebungen *CLANG* (verwendet *UCRT* die Compiler *LLVM* anstatt *GCC*) und *MinGW x86* (32-Bit) bereits heraus [37]. Dementsprechend können von *MSYS2 MinGW* die Varianten *CLANG x64* und *x86* nicht verwendet werden. Auch wurde festgestellt, dass *UCRT x64* funktionsunfähig war. Diese Varianten erzeugen folgende Fehlermeldung:

"[scrcpy] Building client...

ERROR: This python3 seems to be msys/python on MSYS2 Windows, but you are in a MinGW environment

ERROR: Please install and use mingw-w64-x86_64-python3 and/or mingw-w64-x86_64-meson with Pacman".

Auch die Umgebung *MSYS2 MSYS* ist im vorliegenden Fall unbrauchbar, da sie die für das *scrcpy* Build benötigte Bibliothek *libavformat* nicht auffinden kann und daher zu einer Fehlermeldung führt.

"Run-time dependency libavformat found: NO (tried pkgconfig and cmake)

app/meson.build:96:4: ERROR: Dependency ‘libavformat’ not found, tried pkgconfig and cmake".

Diese Erkenntnisse lassen daraus schließen, dass für die Kompilierung die C-Bibliotheken der *Microsoft Visual C++ Runtime* mit dem Modul *msvcrt* benötigt werden und dementsprechend nur die, bereits oben in der Verknüpfung genannte, Umgebung *MSYS 2 MinGW x64* verwendet werden kann [37].

Bei der Kompilierung auf einem Windows-System ist es notwendig, in der Datei *install_release.sh* in Zeile 21 das Kommando *sudo* zum Erhalt von Administratorberechtigungen zu entfernen, da dies bei *MSYS2* aufgrund der fehlenden Funktionalitäten und Windows als Grundlage standardmäßig nicht implementiert wurde.

Für den ordnungsgemäßen Betrieb von *scrcpy* nach der Installation auf demselben System sind keine weiteren Schritte notwendig.

Soll das Tool jedoch auf einem anderen System zum Einsatz kommen, wie im vorliegenden Fall, so bedarf es weiteren Änderungen. Diese sind in den unterschiedlichen Pfaden und Umgebungen bedingt. Sie werden in nachfolgendem Abschnitt beschrieben.

2.5.2.2 Kompilierung von scrcpy als mobile Anwendung

Für die Kompilierung einer mobilen Anwendung (sogenannte portable Version) wurden drei Optionen betrachtet und geprüft. Sie werden nachfolgend erläutert. Eine weitere ungeprüfte Möglichkeit wird zudem im Anschluss genannt. Die auf einem Windows-System einfachste und empfehlenswerteste Erste besteht darin, dem *meson* Build-Kommando die Option „*-Dportable=true“* anzuhängen, damit ähnlich zum richtigen Release mit dem Shell Skript *release.sh* eine Mobile Anwendung gebaut wird.

Option zwei ist es, die Pfade im Quellcode zu verändern. Dazu muss in der Datei *app\meson.build* die Zeile 228 zur Umbenennung der Datei „icon.png“ zu „scrcpy.png“ mit einer Raute auskommentiert oder gelöscht werden. Zusätzlich muss in der Datei *app\src\icon.c* in den Zeilen 17 bis 18 von

„#define SCRCPY_DEFAULT_ICON_PATH \

PREFIX ‘/share/icons/hicolor/256x256/apps/scrcpy.png’”

der Inhalt ‚PREFIX‘ entfernt und der Pfad, sowie Dateiname der *scrcpy.png* geändert werden. In Zeile 17 verbleibt somit nur der Inhalt:

„#define SCRCPY_DEFAULT_ICON_PATH ‚icon.png’“. Daraufhin bedarf es in der Datei *app\src\server.c* einer ähnlichen Änderung. Auch dort muss in der Zeile 19:

„#define SC_SERVER_PATH_DEFAULT PREFIX ‚/share/scrcpy/’ SC_SERVER_FILENAME“

der Inhalt “PREFIX” und die Angabe des Verzeichnisses „‘/share/scrcpy/‘” entfernt werden.

Die letzte und zudem komplexeste Betrachtung besteht darin, das richtige Shell Skript für das Release *release.sh* anstelle der *install_release.sh* zu verwenden. Dabei besteht das große Problem, dass manche Funktionalitäten nicht mit *MSYS2* funktionieren und voraussichtlich ein richtiges Linux System voraussetzen. Um dennoch unter Windows ein richtiges Release kompilieren zu können, müssen die Funktionen zur Erstellung, zum Kopieren und Verwenden der 32-Bit Anwendung in der Datei *release.mk* entfernt werden. Dazu gehört Zeile 41 mit „cp ‚$(DIST)/$(WIN32_TARGET)‘ ‚$(RELEASE_DIR)‘“ und Zeile 45 mit „‘scrcpy-

win32-$(VERSION).zip‘ \“. Zusätzlich können diverse Tests nicht fehlerfrei ausgeführt werden, so wie auch der *scrcpy*-server für die Android basierten Geräte aufgrund eines Fehlers nicht kompiliert werden kann. Daher muss der vorkompilierte Server verwendet werden und er darf nach bzw. vor dem jeweiligen Kompilieren nicht gelöscht werden. Hierzu muss in Zeile 51 im Lösch-Kommando „rm“ die Pfad-Variable des Server-Build Ordners „"$(SERVER_BUILD_DIR)"“ entfernt werden. Auch Zeile 54 bis 64 zum Testen der Anwendung und Kompilieren des Servers sind zu entfernen.

Zu löschende Zeilen befinden sich weiterhin für das Bauen der 32-Bit Anwendung in den Zeilen 77 bis 84, 95 bis 112 für das Bauen des Servers und 133 bis 136 für das Archivieren der 32-Bit Version.

Zu guter Letzt sollte die Datei *icon.png* aus dem Verzeichnis app/data/ in das Hauptverzeichnis kopiert werden, damit die Anwendung sie nach dem Build in dem Verzeichnis auffinden kann.

Optional wäre es möglich, die Anwendung scrcpy direkt unter einem Linux-Betriebssystem zu kompilieren. Möglicherweise wäre dies bei einem vorhandenen Linux-System die einfachste und eine zugleich fehlerfreie Art des Builds. Sie ist in Zukunft noch zu prüfen.

Zusätzliche Änderungen an dem Quellcode von *scrcpy* waren für einen Versuch mehrerer Spiegelungen notwendig, sowie um die Maximierung der Fenster bei Doppelklick zu erhalten und wieder umzukehren. Diese werden in Kapitel 6.3 konkret thematisiert.

2.5.3 TeamViewer

Zwei verschiedene Möglichkeiten für den Fernzugriff auf Android-Geräte bietet *TeamViewer*. Die eine besteht aus dem *TeamViewer QuickSupport* für die kurzfristige Unterstützung und die andere mit dem *TeamViewer Host* für langfristige Zugriffe. [38]

Die private Nutzung ist kostenlos, für Unternehmen ist sie dagegen mit monatlichen Kosten verbunden. Da die HRW *TeamViewer* intern verwendet, sind bereits Lizenzen vorhanden. Für die Verwendung müsste geprüft werden, ob kostenpflichtig weitere Lizenzen notwendig sind. Die kostenlose Version kann jedoch nur ein Gerät gleichzeitig verbinden. Erst die Variante „Corporate“ ermöglicht drei gleichzeitige Verbindungen bzw. Kanäle. [39]

2.5.4 Vysor

Vysor bietet in der kostenfreien Variante eine kabelgebundene Spiegelung und Kontrolle der Geräte. Mit ihr lassen sich Screenshots erstellen. Sie bietet jedoch eine schlechte Bildqualität. Erst durch die Pro- oder Enterprise-Version für beispielsweise einmalige 40 $ wird der Vollbildmodus, eine höhere Qualität der Übertragung und die Funk-Übertragung ermöglicht. [40]

2.6 Versionsverwaltung mit Git

Git bietet eine Versionskontrolle, die es ermöglicht, Änderungen an einer oder mehreren Datei/-en aufzuzeichnen. Dadurch ist es möglich, spezifische Versionen im Nachhinein erneut zu öffnen bzw. zu ihnen zurückzukehren. Einzelne Änderungen können rückgängig gemacht werden und besser nachvollzogen werden. Weiterhin ist es möglich, Änderungen später zu vergleichen oder einzusehen, wer zu welchem Zeitpunkt welche Änderungen durchgeführt hat. So lässt sich nachvollziehen, wer zuletzt etwas geändert hat, das Probleme verursacht haben könnte. Allgemein können auch Probleme beschrieben und nachvollzogen werden.
Git wurde 2005 als kostenfreie Alternative zu *BitKeeper* für die Programmierung von Linux entwickelt, da *BitKeeper* zu der Zeit kostenpflichtig wurde. Mittlerweile ist *Git* derart fortgeschritten, dass es einfach zu bedienen ist und trotzdem die grundlegenden Qualitäten besitzt. Es ist sehr schnell, sehr effizient bei großen Projekten und hat ein einzigartiges Verzweigungssystem für die nicht-lineare Entwicklung.
Der große Unterschied zwischen *Git* und anderen Versionskontrollmethoden liegt im Umgang mit den Daten. Viele andere Systeme speichern die Daten als Liste mit dateibasierten Änderungen. Diese werden als eine Reihe von Dateien gespeichert.
Git arbeitet dagegen mit Schnappschüssen. Bei jeder übertragenen Änderung wird ein Abbild erstellt, das alle aktuell vorliegenden Dateien widerspiegelt. *Git* speichert dabei einen Verweis auf den Schnappschuss. Für die Effizienz werden unveränderte und nicht gespeicherte Dateien nicht erneut abgespeichert, sondern nur ein Verweis auf die vorherige, identische und bereits gesicherte Datei erstellt.
Darüber hinaus arbeitet *Git* überwiegend mit lokalen Dateien und Ressourcen. Es werden nicht unbedingt Informationen von einem anderen Computer aus dem Netzwerk benötigt. Im Vergleich zu anderen Systemen kann so die Netzwerklatenz umgangen werden und *Git* sehr schnell arbeiten. Änderungen können daher

offline durchgeführt werden und flexibel beispielweise erst bei Abschluss dessen hochgeladen werden. [41]

2.7 Python

Entwickelt wurde *Python* zu Beginn der 90er-Jahre von Guido von Rossum als Skriptsprache. 1994 wurde sie in der ersten Version veröffentlicht. Diese Programmiersprache veraltet nicht, sie wird stetig gewartet und durch Updates fortgeschrittener. Zum Beispiel wurde die Hauptversion *Python 3* bereits 2008 vielseitig aktualisiert. Nach aktuellem Stand von August 2022 liegt *Python* in Version 3.10 vor [42].
Viele Projekte und Informationen basieren noch auf *Python 2*. Aufgrund dessen, dass *Python 3* modernere Konzeptionierungen bietet und in vielen Bereichen nicht abwärtskompatibel ist, wird empfohlen, *Python 3* für neuartige Projektentwicklungen zu verwenden.
Nach aktuellem Stand ist *Python* im TIOBE-Popularitätsindex der Programmiersprachen auf Platz 1 vorgedrungen und hat die Programmiersprache C als Erstplatzierten des letzten Jahres übertroffen [43]. Damit bestätigt sich auch der von Inden festgestellte Trend von August 2021, bei dem *Python* möglicherweise auch den 1. Platz mit C übertrifft. Java wurde bereits im August 2021 als langfristiger 2. Platz überholt.
Wichtige Aspekte sind dabei die freie Verfügbarkeit für die diversen üblichen Betriebssysteme. Außerdem ist *Python* als Programmiersprache simpel in der Aneignung (für Einsteiger wesentlich einfacher als Java oder insbesondere C, sowie C++). Im Bereich der Forschung und Lehre erfährt *Python* ebenfalls an Popularität und an Gewichtung. Bedeutsam ist hier, dass *Python* sowohl die objektorientierte, als auch die funktionale Softwareentwicklung unterstützt und je nach Verwendungszweck flexibel entschieden werden kann. [44]

3 Implementierung

In diesem Kapitel werden die Anforderungen, der Aufbau und die Implementierung der Administratoroberfläche beschrieben. Zu Beginn wird die Anforderungsanalyse aufgezeigt. Darauffolgend wird die Systemarchitektur dargestellt. Es wird verdeutlicht, wie die Hauptanwendung aufgebaut und implementiert wurde. Weiterhin werden die notwendigen Änderungen an der Software *scrcpy* beschrieben. Darüber hinaus wird erläutert, wie die Konfiguration der Endgeräte funktioniert und zuletzt die Softwaretests aufgeführt und auf die Bewertung der Code-Qualität eingegangen. In diesem Kapitel handelt es sich einerseits um die Administratoroberfläche in Python und andererseits um das Spiegelungstool *scrcpy* in der Programmiersprache C. Nachfolgend werden die Anforderungen an die Tools näher aufgeführt.

3.1 Anforderungen

Die Anforderungen an diese Anwendung können von Leitständen aus der Energiewirtschaft abgeleitet werden. Zudem sind bei der weiteren Entwicklung des Tools um Funktionalitäten der Steuerung als vollwertige Administratoroberfläche sicherheitsrelevante Inhalte zu beachten.

Die generellen Anforderungen an Leitstände sind dabei der zuverlässige Betrieb dessen, eine erweiterbare Funktionalität, sowie ein kurzfristiges Ansprechverhalten (Echtzeitverhalten) [45]. Wird die Oberfläche zukünftig zur Steuerung der Geräte verwendet, so sind weitere Sicherheitsaspekte zu betrachten. Beispielsweise sollte die Steuerung nur von zugelassenen Geräten erfolgen und der Administrator muss authentifiziert werden [46]. Relevant ist außerdem die Protokollierung der administrativen Zugriffe [47].

Zu den daraus abgeleiteten Anforderungen gehören derzeit die Bedienbarkeit/ Benutzerfreundlichkeit, funktionale Anwendbarkeit, Flexibilität, Latenz, sowie ein guter Kompromiss aus Bildgröße und Bildqualität. Zudem sind der Aufwand der Implementierung, sowie die Kosten für die Verwendung der Anwendung nicht zu vernachlässigen.

Weiterhin soll die Bedienung so schnell und einfach wie möglich erfolgen, sodass es nicht notwendig ist, vor jedem Termin eine längere Zeit mit der Einrichtung des Tools zu verbringen.

Für diese Vereinfachung sind zwei Funktionalitäten angedacht. Erstens eine einfache Einrichtung der VR-Brillen und das Übertragen und Abspeichern der Daten im Hintergrund. Zweitens die Abspeicherung der gesamten Konfiguration in einer JSON-Datei. Diese Datei kann bei einem nächsten Termin und gleicher Konfiguration übernommen, oder mittels eines JSON-Editors einfach angepasst werden.

Das Tool soll die flexible Möglichkeit bieten, ein bis neun Android-basierte Geräte auf einem Bildschirm zu spiegeln und die Darstellungen optimal auszurichten.

Um außerdem so viele Geräte wie möglich auf einem Bildschirm zu spiegeln und die notwendige Bandbreite gering zu halten, ist es sinnvoll, das Bild bereits vor der Übertragung zu beschneiden. Es sollte nicht übermäßigen Bildschirminhalt in Anspruch nehmen, unnötige Werbung wie Wasserzeichen oder viel mehr als nur das benötigte Bild darstellen (wie beispielsweise die zweifache Bild-Übertragung aufgrund zweier Bildschirme bei den VR-Brillen). Bei den einzelnen Fenstern soll es zudem möglich sein, diese durch Doppelklick zu maximieren, um den jeweiligen Anwendenden optimal unterstützen zu können.

Hinzu kommt, dass die Bildschirmspiegelung mit möglichst geringer Latenzzeit erfolgen soll. Eine große Verzögerung kann sowohl bei der Demonstration der Funktionalitäten, als auch bei der Unterstützung zu Verwirrungen führen und daher nachteilig sein. Hierzu ist die gegebenenfalls eine Anpassung der Bildqualität, Auflösung und Bitrate notwendig.

Zuletzt sollte der Aufwand der Implementierung innerhalb des Rahmens dieser Arbeit liegen und die Kosten für die Verwendung so gering wie möglich gehalten werden.

Aufgrund der in Kapitel 2 genannten Eigenschaften, wird für diese Arbeit *scrcpy* verwendet.

3.2 Hauptanwendung der Administratoroberfläche

Für die Erstellung der Hauptanwendung wurden diverse Vorbereitungen getroffen. Für die ersten Abstimmungen mittels Exposés wurde bereits ein Programmablaufplan nach DIN 66001 mit den grundlegenden Funktionalitäten und für die Code- und Versionsverwaltung ein *GitLab* Repository erstellt. Außerdem wurden erste Tests mit Programmiersprachen und Funktionen ausgeführt. Bei diesen Tests hat sich für das Hauptprogramm die Programmiersprache Python positiv abgezeichnet, da sie die Steuerung im Vergleich zu C und C# am einfachsten, nachvollziehbarsten und besonders funktional macht. Dadurch ergibt sich an der HRW mit ihr eine hohe Wartbarkeit.

Für die Endfassung wurde der Programmablaufplan erweitert. Dieser wird im Folgenden dargestellt.

3.2.1 Programmablauf

Der folgende Programmablaufplan (Abbildung 4) verdeutlicht den ersten geplanten und umgesetzten Ablauf des entwickelten Programmes der Hauptanwendung. Er wird im Folgenden näher erläutert.

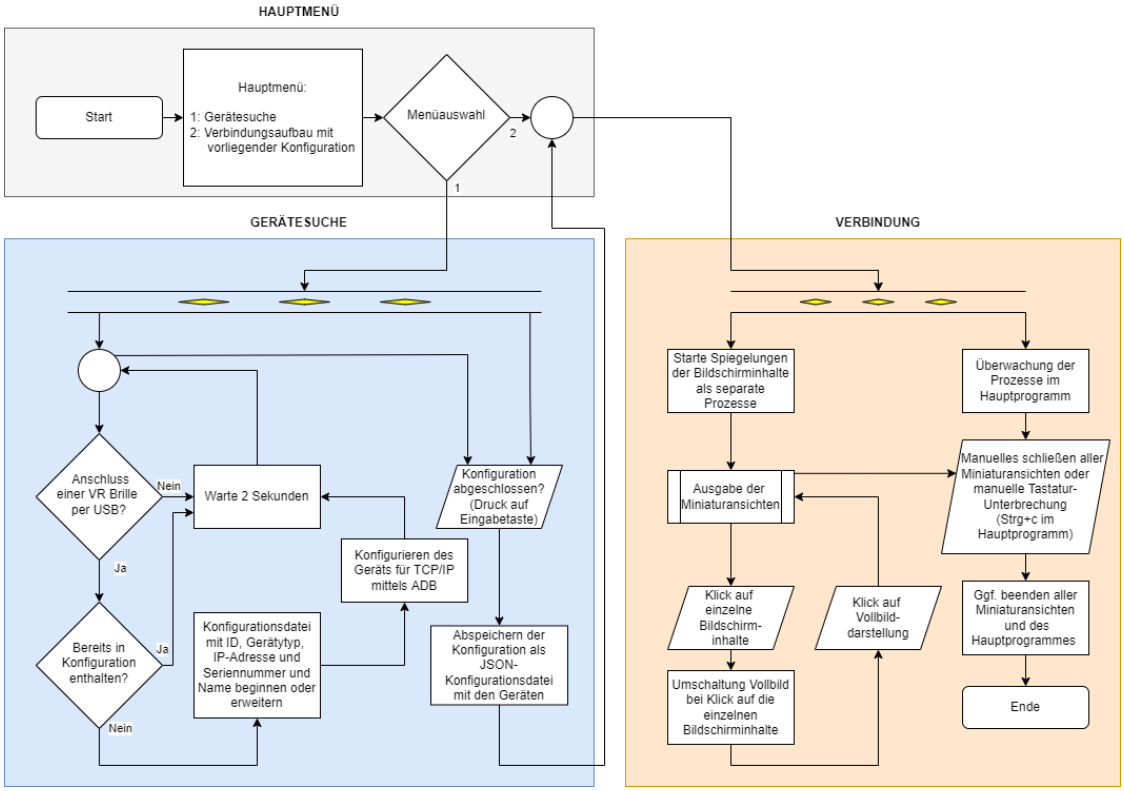

Abbildung 4: Programmablaufplan Version 1 nach DIN 66001

Bei Programmstart wird eine Kommandozeile geöffnet, die einem die grau dargestellte Menüauswahl zur Verfügung stellt. Dort können über Ziffern die Hauptfunktionalitäten gestartet werden. Mit der Ziffer ‚1' kann die Gerätesuche, mit der ‚2' der Verbindungsaufbau gestartet werden.

Bei Eingabe der ‚1' wird die Gerätesuche (linker blauer Prozess-Bereich) folgendermaßen ausgeführt. Es wird zuerst ein paralleler Subprozess gestartet, der die Betätigung der Eingabetaste abfragt. Hierdurch kann der Hauptprozess der Gerätesuche jederzeit unterbrochen werden. Anschließend wird bei der Verzweigung überprüft, ob ein Gerät per USB-Anschluss angebunden ist. Ist kein Gerät angebunden, so wird 2 Sekunden gewartet und dieses Verhalten erneut überprüft. Ist ein Gerät verbunden, so wird geprüft, ob es bereits in der Konfiguration enthalten ist. Ist dies der Fall, so wird die Gerätesuche ebenfalls nach 2

Sekunden von vorne gestartet. Ist das verbundene Gerät nicht in der Konfiguration enthalten, so wird vom Benutzer die Eingabe eines Namens abgefragt und abschließend durch Betätigung von Enter ein neuer Eintrag als Klassenobjekt zu dem Gerät angelegt. Dieser enthält die ID, den Gerätetypen, die IP-Adresse, die Seriennummer und den Namen. Dabei wird das Gerät über USB mit der ADB-Schnittstelle direkt für die Verwendung über TCP/IP konfiguriert. Bei Betätigung der Eingabetaste außerhalb der Namenseingabe werden alle vorhandenen Geräte in eine JSON-Konfigurationsdatei abgespeichert. Daraufhin wird der Prozess Gerätesuche und der Subprozess für die Abfrage der Eingabetaste beendet, sowie automatisch zum zweiten Prozess gewechselt.

Bei diesem Wechsel oder bei Eingabe der ‚2' wird die vorhandene JSON-Konfiguration eingelesen und es werden erneut parallele Subprozess gestartet. Diesmal handelt es sich bei den Subprozessen um die einzelnen Bildschirmspiegelungen, die als externe Prozesse gestartet werden. Das Hauptprogramm überprüft nach Start der Spiegelungsanwendungen über *scrcpy*, ob einzelne Spiegelungen geschlossen werden. Auch wird im Hauptprogramm geprüft, ob durch die Betätigung der Tastenkombination ‚Strg+C' alles beendet werden soll. Diese führt dazu, dass alle Subprozesse mit *scrcpy* und abschließend auch die Hauptanwendung beendet wird.

Zu Version 2 der Hauptanwendung gibt es zwei entscheidende Unterschiede, wie in folgender Abbildung 5 ersichtlich.

Zum einen wird dort anstatt der Kommandozeile für die Menüführung GUI verwendet. Zum anderen ist in Menüpunkt drei die relevanteste Funktionalität enthalten. Auch das Hauptmenü ist erweitert worden, es gibt drei Menüpunkte. Der erste Punkt hat sich bis auf die grafische Auswahl nur geringfügig verändert. Aufgrund der vereinfachten Erweiterbarkeit dient dort weiterhin die Kommandozeile zu Ein- und Ausgabezwecken. Die geringfügige Änderung hat sich dort im Aufbau der Gerätekonfigurationsdatei im JSON-Format ergeben. Hier sind Statusinformationen zu den Geräten vorhanden. Sie enthalten Infos zur Erreichbarkeit, Verbindung (Konfiguration) und Auswahl. Der direkte Verbindungsaufbau zu den Geräten verbleibt unverändert.

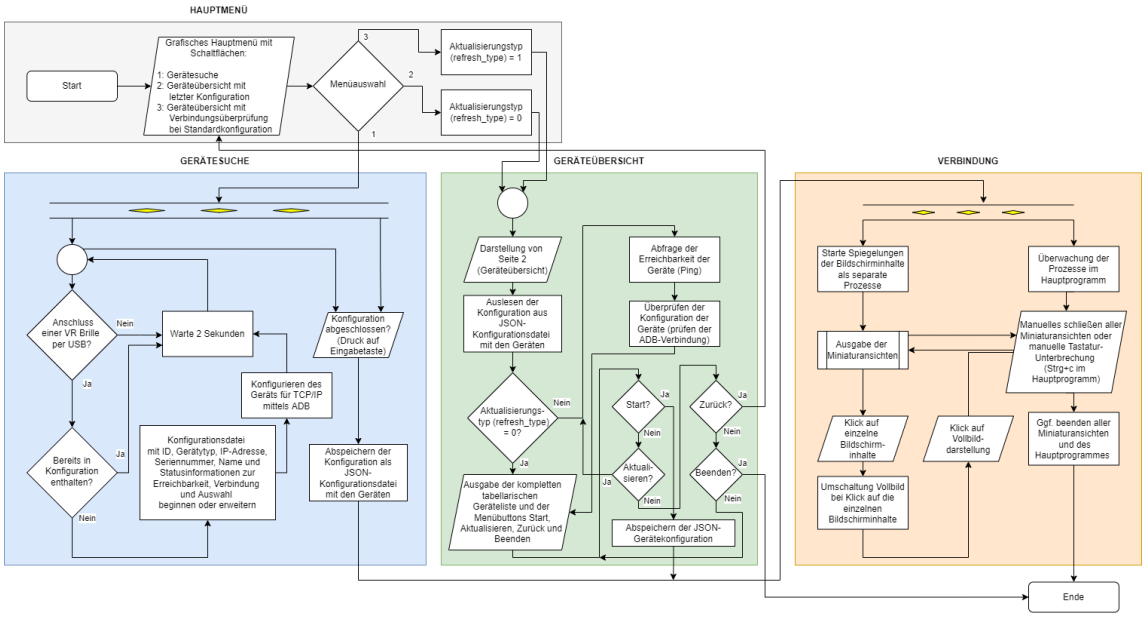

Abbildung 5: Programmablaufplan Version 2 mit GUI nach DIN 66001

Der Zweite Menüpunkt wurde um die Geräteübersicht und -Auswahl erweitert, sowie der dritte Menüpunkt ähnlich zum Zweiten aufgebaut.

Diese beiden Übersichten werden tabellarisch in der GUI dargestellt. Beim zweiten Menüpunkt wird bei Verwendung der letzten Konfiguration die Funktionalität der Übersicht aller Geräteinformationen in der GUI einschließlich der Auswahlmöglichkeit zur Spiegelung dazwischen dargestellt, sowie anschließend der Verbindungsaufbau durch die Startschaltfläche ermöglicht. Die Betätigung dessen führt dazu, dass die aktuelle Konfiguration vor dem Verbindungsaufbau als JSON-Datei gesichert wird. Zudem können die Gerätestatus mit der Schaltfläche „Aktualisieren“ erneuert werden, mit „Zurück“ wieder in das Hauptmenü gelangt werden und die Anwendung durch Betätigung der Taste ‚Beenden‘ oder einfaches Schließen dessen beendet werden. In diesem zweiten Menüpunkt werden die Gerätestatus in Gelb dargestellt, da sie zu dem Zeitpunkt nicht mehr aktuell bzw. ungeprüft sind. Lediglich die letzte Auswahl der Geräte wird mit aus der letzten Konfigurationsdatei übernommen.

Beim dritten Menüpunkt erfolgt mit der Variable des Aktualisierungstyps durch die Zuordnung der eins ein sehr ähnlicher Ablauf wie bei Punkt zwei. Der entscheidende Unterschied liegt hier bei der Aktualisierung der Gerätestatus. Denn beim Aufruf der Funktion werden die Erreichbarkeit und Verbindung zu den Geräten überprüft, somit aktualisiert und abschließend angezeigt. Ist ein Gerät erreichbar, aber die Verbindung über ADB nicht möglich, so muss es konfiguriert werden. Hierzu wurde ein frei verfügbares Windows-Stapelverarbeitungsskript (Batch-Skript) verwendet und erweitert, das in Kapitel 6.4 erläutert wird.

3.2.2 Verwendete Module

Für die vielen Funktionalitäten bei der Softwareumsetzung in Python werden die folgenden, bereits existierenden und öffentlich zugängigen Module benötigt. Sie müssen teilweise über den Python internen Paketmanager *pip* nachinstalliert werden.

Eine gesamte Übersicht der implementierten Klassen, Module, Methoden und Funktionen ist in Anhang A beigefügt.

Um zusätzliche Ausgaben zu Informationszwecken oder zur Fehlersuche zu generieren, kann einfach die *print*-Funktion verwendet werden, oder das Modul *Logging* eingesetzt werden. Dieses Modul hat den Vorteil, dass die mit ihr über die Funktion *logging.info("Text")* erstellten Informationen mit einer Konfiguration zu Beginn aktiviert oder deaktiviert werde können.

Zur Aktivierung kann in der Administratoroberfläche durch die Zuweisung von *DEBUG=True* die Funktion *logging.basicConfig(level=logging.INFO) aufgerufen werden.*

Warnungen lassen sich auch auf dieses Weise erstellen und unterscheiden. Diese werden unabhängig von der Debug-Konfiguration ausgegeben. Hierfür wird die Funktion *logging.warning("Text“)* verwendet.

Das Modul *time wird* benötigt, um kurze Wartezeiten innerhalb der Anwendung zu erzeugen. Beispielsweise bei dem Scanvorgang über den USB-Anschluss, um so die Abfragen desselben Gerätes zu reduzieren. Hierzu wird die Funktion *time.sleep(Anzahl_an_Sekunden)* verwendet.

In der verwendeten Python IDE PyCharm ist bereits ein sogenanntes linting-Tool zur statischen Quellcode-Analyse enthalten. Um die Funktionalität dessen zu erweitern und zur Konformität mit den Python Gestaltungsrichtlinien *PEP 8* wird zusätzlich das Python spezifische Modul *pylint* verwendet. Mit dem Befehl *pylint main.py* kann innerhalb des Projektordners die Startdatei überprüft werden. Mit *pylint scrcpy-handler* im übergeordneten Ordner lässt sich dagegen das gesamte Projekt prüfen. Aufgrund der Tatsache, dass externe Module nicht überprüft werden können, muss das Modul *win32gui* von der Prüfung ausgeschlossen werden. Dies geschieht mit dem Argument *–extension-pkg-whitelist=win32gui.*

In Bezug auf die Erfassung von Tastendrücken der Tastatur als Unterbrechungs-Ereignisse (Interrupts) hat Python einen Nachteil. Die Erkennung würde systemweit außerhalb der Anwendung, sowie in anderen Anwendungen ausgeführt werden. Um daher das aktuelle Fenster zu erfassen, kann mit dem Modul *win32gui* über die beiden verschachtelten Funktionen

GetWindowText(GetForegroundWindow()) der aktuelle Fenstertitel erkannt, bei Programmstart abgespeichert und bei Tastendruck verglichen werden. Dadurch wird das Problem gelöst.

Die Unteranwendung *scrcpy* muss parallel während der Verwendung der Python-Hauptanwendung der Administratoroberfläche gestartet und ausgeführt werden. Dafür wird das Modul *subprocess* verwendet und über die Funktion *subprocess.Popen("Befehl")* die einzelnen Prozesse gestartet. Zudem werden die einzelnen Prozesse der Variable *process* zugewiesen und für die weitere Verwendung durch die Funktionalität *processes.append(process)* in einer Prozessliste abgespeichert.

Mit einer Iteration über die Prozesse können diese durch die Funktion *process.terminate()* beim Beenden der Hauptanwendung einschließlich beendet werden.

Die Funktion *signal* aus dem gleichnamigen Modul wird benötigt, um die Erstellung von Tastendrücken der Tastatur als Unterbrechungs-Ereignisse zu erstellen.

Um die Python-Anwendung trotz einer Unterbrechung ordnungsgemäß zu beenden, wird die Funktion *exit* des Moduls *sys* benötigt und mit einer Null als Argument ausgeführt: *sys.exit(0)*.

Das Modul *keyboard* wird für die Betätigung von Tastendrücken während des Programmlaufes benötigt. Hierüber kann die Eingabetaste in der Kommandozeile als Hotkey während des Scanvorgangs abgefragt und verwendet werden. Ansonsten wäre es bis zum Ende der jeweiligen Vorgänge notwendig, die Eingabetaste zu halten.

Für die Verbindungsüberprüfungen, -konfigurationen und zum Abfragen der Android-basierten Endgeräte wird eine Kommunikation über ADB benötigt. Um die TCP/IP Konfiguration durchzuführen, die IP-Adresse, Seriennummer und den Gerätetypen auszulesen wird daher das Modul *Adbutils* verwendet.

Als letztes Modul wird Tkinter für die Erstellung, Darstellung und Bedienbarkeit der Benutzeroberfläche verwendet.

3.2.3 Modularisierung

Für eine bessere Übersicht wurde der Code in verschiedene Module aufgeteilt. Hierzu zählt das Hauptmodul *main.py* und die Konfigurationsdatei *config.py*, die die globalen Variablen enthält. Die Steuerung und die Eingaben erfolgen mit dem *input_controller.py* und das Verbindungsmodul zur Unteranwendung *scrcpy* ist der *scrcpy_connector.py*. Für die Ausführung der ADB Befehle zwischen den

Android-basierten Endgeräten und der Anwendung wird der ADB-Handler *adb_handler.py* verwendet und zuletzt für die Verarbeitung der Konfigurationen im JSON-Format, der *json_config_handler.py*. Dieser dient dem Importieren und Exportieren der Konfigurationsdateien.

3.2.4 Funktionen

Die Hauptfunktionalität ist im Hauptmodul *main.py* enthalten. Wird dieses ausgeführt, so wird die Main-Funktion aufgerufen.

Diese *Main*-Funktion erzeugt dem Benutzer eine Auswahl für das Hauptmenü. In Version 1 wird diese innerhalb der Kommandozeile und in Version 2 dagegen über eine GUI gezeigt.

Die Menüführung in Version 1 geschieht über die Kommandozeile und Eingaben der folgenden Ziffern. Bei Eingabe der ‚1' wird dort der Suchvorgang für Geräte über die Funktion *scan()* gestartet.

Die Eingabe einer ‚2' führt über die Funktion *connect()* zu einer Verbindung der Geräte und Aktivierung der Gerätespiegelungen anhand einer vorhandenen Konfigurationsdatei.

Die Eingabe einer anderweitigen Taste, sowie eine leere oder nicht vorhandene Konfigurationsdatei, führt zum sofortigen Abbruch.

In Version 2 erfolgt die Menüführung über die GUI mit dem Modul *Tkinter*. Das Resultat der GUI wird in Kapitel 5 unter Abbildung 10 dargestellt und dort näher erläutert. In der GUI werden im Hauptmenü drei Menüpunkte als Schaltflächen dargestellt. Die ersten beiden oben genannten und zusätzlich ein neuer Menüpunkt. Bei diesem handelt es sich um den automatischen Gerätescan und die Überprüfung der Geräte.

Für die Bedienung mit der GUI wird grob für die Funktionalitäten aus dem Hauptmenü jeweils eine Seite in der Hauptklasse *ScrcpyHanlderApp* im Hauptmodul implementiert.

Dazu zählen die Startseite (sozusagen Seite 0), Seite 1 für die manuelle Gerätesuche über USB und Seite 2 für die automatische Gerätesuche. Die Schaltflächen 2 und 3 verwenden beide die Seite 2 (*PageTwo*), werden jedoch über eine Variable zum Aktualisierungstypen (*refresh_type*) als Argument im Methodenaufruf der Seite mittels *switch_frame(Seite, Aktualisierungstyp)* differenziert. Schaltfläche 2 verwendet somit keinen Aktualisierungstypen, da die vorhandene Konfiguration verwendet werden soll. Dort ist *refresh*_type die „0" zugewiesen.

Schaltfläche 3 soll die Konfiguration aktualisieren, weshalb *refresh_type* die 1 zugewiesen wird.

Wird der manuelle Scanvorgang über USB mit Schaltfläche 1 gestartet, wird zuerst über das Modul *input_controller.py* mit der Funktion *add_hotkey_enter()* der Hotkey Eingabetaste mit dem Modul *keyboard* erstellt. Damit wird die Betätigung der Eingabetaste jederzeit erkannt und die Gerätesuche kann unterbrochen werden. Ohne diese Hotkey-Funktionalität müsste die Taste mehrfach innerhalb der Suche abgefragt werden oder in dem Moment einer abgeschlossenen Suche gedrückt oder gehalten werden.

Ist der Hotkey hinzugefügt worden, wird die Gerätesuche über das Modul *adb_handler* mit der Funktion *device_search_loop()* in einer Schleife gestartet. Die sucht alle 2 Sekunden aktiv nach einem Gerät.

Wird die Suche beendet, speichert das Modul *json_config_handler* mit der Funktion *write_config_to_json()* die aktuelle Konfiguration ab. Wie mit Eingabe der ‚2' (Version 1) oder Auswahl der Schaltfläche 2 und dem anschließenden Betätigen der Schaltfläche ‚Start' (Version 2) wird hier die Verbindung mit den Geräten aufgebaut und die Bildschirmspiegelungen werden gestartet.

Die Funktion *connect()* liest in den beiden genannten Fällen vorerst mit der Funktion *read_config_from_json()* die Konfiguration über das Modul *json_config_handler.py* aus der JSON-Datei aus. Dann instanziiert sie diese als Objekte der Klasse Device (näheres zu der Klasse folgt im nächsten Unterkapitel 6.2.7). Im Anschluss werden die Bildschirm-Spiegelungs-Prozesse mit dem Tool *scrcpy* als Unteranwendungen über die Funktion *scrcpy_process_starter()* des Moduls *scrcpy_connector* gestartet (aufgrund der Komplexität dazu nachfolgend mehr). Abschließend erfolgt über die Funktion *scrcpy_process_terminator()* des selben Moduls die Beendigung. Ohne diese Funktion würde sich das Hauptprogramm nach dem Starten der Unteranwendungen beenden und alle einzelnen Spiegelungen müssten manuell oder extern mit weiteren Funktionalitäten beendet werden.

Für die dritte Schaltfläche bei Zuweisung des Aktualisierungstyps zu der 1 wird die Funktion *autocheck_connected_devices()* aufgerufen. Diese liest die aktuelle Geräteliste aus und aktualisiert die Statusinformationen in der Geräteliste über die Funktion *device_checker()* des Moduls *adb_handler*. Dort wird über eine *for*-Schleife für jedes Gerät die Funktion *check_tcp_device()* aufgerufen. Sie überprüft über TCP/IP mit einem Ping die Erreichbarkeit der Geräte, sowie mittels *adb_con.connect("Geräte_IP_Adresse:Port")* die ADB Verbindungen (den

Konfigurationsstatus) zu den Geräten mit Standard-Port von 5555 und der Standard ADB-Verbindung *adb_con*.

Die Funktion *scrcpy_process_starter()* wird auf die per Argument übergebene Geräteliste angewendet und daher beim Starten der Bildschirmspiegelungen ausgeführt. Sie bildet die für die Unter-Anwendung *scrcpy* benötigten Argumente ab, sodass die jeweiligen Spiegelungen wie gewünscht dargestellt werden. Zu den benötigten Argumenten gehören die IP-Adresse, die Option „–no-control“ (-n) und die Bitrate in Megabitprosekunde (Mbps bzw. Mb/s). Auch die maximal zu übertragene Bildgröße in Pixeln (die größte Dimension wird begrenzt und die andere angepasst, sodass das bestehende Bildverhältnis beibehalten wird), die Bild-Zuschnitte („—crop“), wie auch die Bild-Koordinaten auf der X- und Y-Achse werden benötigt. Zuletzt ist die Angabe der darzustellende Bildgröße und -breite nicht zu vernachlässigen.

Für eine vernünftige Bildübertragung ohne große Latenzzeiten und ohne Bildaussetzer hat sich über Wireless LAN (WLAN) die Konfiguration auf eine Bitrate von 2 Mb/s und eine maximale Bildgröße von 800 Pixeln ergeben.

Zur Darstellung der unterschiedlichen Anzahlen wurden folgende unterschiedliche Konstellationen aufgestellt. Bei einem Gerät erhält es 100 % des Bildschirminhaltes, bei zwei Geräten jeweils 50 %, bei drei bis vier Geräten nur ¼ Inhalt, also 25 %. In dem Bereich zwischen fünf bis neun Geräten erfolgt die Aufteilung nur zu 1/9 (11,1 %).

Für die Koordinierung der Anwendung müssen zusätzliche Aspekte, wie die Verwendung eines Zweitbildschirmes oder Projektors und die Darstellung der Taskleiste, sowie der Titelleiste der Anwendung beachtet werden.

Die erstellte Formel (1) für den Beginn der horizontalen Fensterposition (X-Achse) in Python ist folgend definiert. Die Formel führt dazu, dass von der Geräte-Anzahl (beginnend bei 1) zunächst eins subtrahiert wird. Das Resultat der Modulo-Rechnung bildet bei zwei verschiedenen Faktoren sozusagen jeweils zeilenweise das ganzzahlige Intervall ab, um in jeder Zeile die notwendigen Fensterpositionen zu erzeugen.

Bei einem Faktor von 2 wird das ganzzahlige Intervall von [0, 1] erzeugt.

Bei dem Faktor 3 wird die Menge {0, 1, 2} gebildet.

Die Gerätezahl aus diesen Intervallen wird daraufhin mit der benötigten Fenstergröße aus der Matrix multipliziert, um das notwendige Raster zu erzeugen. Im Anschluss wird die Abweichung in X-Ausrichtung (für die Verschiebung auf einen zusätzlichen Bildschirm), sowie ein Pixel addiert, um den Rahmen des Fensters

ordnungsgemäß darzustellen und das Ergebnis auf ganze Zahlen gerundet. Zudem wird von einer Gleitkommazahl in eine Ganzzahl konvertiert. Abschließend erfolgt die Umwandlung zum String für die weitere Verkettung der Argumente.

Bei der verwendeten Matrix stellt die erste Dimension die Zeile für die Anzahl der Geräte dar, die zweite Dimension als Spalte die jeweilige Größe der benötigten Ausrichtung. Die Matrix wird nach der zweiten Formel ausführlicher beschrieben.

$$win_{pos,x} = str\Big(int\Big(round\big(1 + offset_x + \big((current\_device\_count - 1)\ \%\ factor\big) \cdot window\_resolutions[device\_count][0], 0\big)\Big)\Big) \quad (1)$$

mit:

$win_{pos,x}$	:	Fensterposition auf der X-Achse
str	:	Funktion zur Konvertierung eines Wertes zu einem String
int	:	Konvertierung eines Wertes zu einem Integer
round(a,b)	:	Funktion zum Runden auf ganze Zahlen (da b=0) mit dem Rückgabetyp Float
$offset_x$	:	Abweichung auf der X-Achse
current_device_count	:	Aktuelle Gerätenummer (beginnend bei 1)
%	:	Modulo-Operator
factor	:	Faktor je nach Geräteanzahl (bei Geräteanzahl ≤ 4: factor = 2; Geräteanzahl > 4: factor = 3)
window_resolutions	:	Matrix mit Fenstergrößen
device_count	:	Summe an Geräten (beginnend bei 0 für 1 Gerät)

Die nachfolgende Formel (2) stellt den Beginn der vertikalen Fenster-Position (auf der Y-Achse) dar. Hier werden, ähnlich wie bei der vorangegangenen Formel, die Anzahl des Gerätes durch die Subtraktion von 1 und der ganzzahligen Division (div) erneut ganzzahlige Intervalle, sozusagen spaltenweise, gebildet.

Bei dem Faktor 2 wird das Intervall von [0, 1].

Der Faktor 3 erzeugt das Intervall [0, 2].

Die ersten entweder 2 oder 3 Geräte (je nach Faktor) führen bei der ganzzahligen Division zu einer 0 und somit zur ersten Zeile. Diese jeweilige Zeile wird einmal mit der Höhe der Titelleiste und an zweiter Stelle mit der Fenstergröße auf der Y-Achse aus der Matrix multipliziert. Die Summe aus diesen beiden Multiplikationen, sowie der Abweichung in Y-Ausrichtung (falls der zweite Bildschirm über- oder oberhalb liegt) und einem Pixel für den Rahmen ergibt die vertikale Fensterposition. Das Ergebnis wird zuletzt auf ganze Zahlen gerundet, von einer Gleitkommazahl in eine Ganzzahl konvertiert und abschließend als String für die weitere Verkettung der Argumente umgewandelt.

$$win_{pos,y} = str\Big(int\Big(round\big(offset_y + 1 + \big((current\_device\_count - 1)\ div\ factor\big) \cdot title\_bar\_height + \big((current\_device\_count - 1)\ div\ factor\big) \cdot window\_resolutions[device\_count][1]\big)\Big)\Big) \quad (2)$$

mit:

$win_{pos,y}$	:	Fensterposition auf der Y-Achse
str	:	Funktion zur Konvertierung eines Wertes zu einem String
int	:	Konvertierung eines Wertes zu einem Integer
round(a,b)	:	Funktion zum Runden auf ganze Zahlen (da b=0) mit dem Rückgabetyp Float
$offset_y$	:	Abweichung auf der Y-Achse
current_device_count	:	Aktuelle Gerätenummer (beginnend bei 1)
div	:	Ganzzahlige Division
factor	:	Faktor je nach Geräteanzahl (bei Geräteanzahl ≤ 4: factor = 2; Geräteanzahl > 4: factor = 3)
title_bar_height	:	Höhe der Titelleiste in Windows (Standard 30 Pixel)
window_resolutions	:	Matrix mit Fenstergrößen
device_count	:	Summe an Geräten (beginnend bei 0 für 1 Gerät)

Für die Bestimmung der Bildgrößen wurde in Python eine 2D-Matrix mit einer Länge von neun Unterlisten verwendet (insgesamt 18 Elemente), da bis zu neun Spiegelungen ermöglicht werden. Jede Unterliste enthält zwei Elemente, eins für die X-Achse und eins für die Y-Achse. Das ergibt die folgenden Formeln.

Für die Spiegelung eines Geräts wird Formel 3 verwendet. Die Auflösung des Bildschirmes auf der X-Achse wird übernommen, für die Y-Achse wird von der entsprechenden Auflösung die Höhe der Titelleiste und der Startleiste subtrahiert.

$$window\_resolution[0] = \left[resolution_{width}, resolution_{height} - title\_bar_{height} - task\_bar_{height}\right] \quad (3)$$

mit:

window_resolution[0]	:	Fenstergrößen für ein Fenster als Liste ‚[X,Y]'
$resolution_{width}$	:	Horizontale Bildschirmauflösung (z. B. 1920 Pixel)
$resolution_{height}$	:	Vertikale Bildschirmauflösung (bspw.1080 Pixel)
$title\_bar_{height}$	:	Höhe der Titelleiste in Windows (Standard 30 Pixel)
$task\_bar_{height}$	:	Höhe der Windows Startleiste (Standard 40 Pixel)

Bei zwei Geräten teilen sie sich, wie in Formel 4 gezeigt, die Bildbreite.

$$window\_resolution[1] = \left[\frac{resolution_{width}}{2}, resolution_{height} - title\_bar_{height} - task\_bar_{height}\right] \quad (4)$$

mit:

window_resolution[1]	:	Fenstergrößen für zwei Fenster als Liste ‚[X,Y]'

Bei drei bis vier Geräten teilen sie sich, wie in Formel 5 gezeigt, die Höhe des Bildes und die Bildbreite auf zwei Zeilen und zwei Spalten. Daher erfolgen die beiden Divisionen der Auflösungen durch zwei. Weiterhin wird einmalig die Höhe der Titelleiste pro Fenster subtrahiert. Zusätzlich muss, je nach Anzahl der Zeilen, die Höhe der Startleiste durch die Anzahl der Zeilen dividiert werden, um eine ordnungsgemäße Aufteilung zu gewährleisten. Sie wird daher durch zwei dividiert.

$$window_{resolution}[2] = window_{resolution}[3] = \left[\frac{resolution_{width}}{2}, \frac{resolution_{height}}{2} - title\_bar_{height} - \frac{taskbar_{height}}{2}\right] \quad (5)$$

mit:

window_resolution[2] : Fenstergrößen für drei Fenster als Liste ‚[X,Y]'

window_resolution[3] : Fenstergrößen für vier Fenster als Liste ‚[X,Y]'

Bei fünf bis neun Geräten teilen sie sich, wie in Formel 6 gezeigt, die Höhe und die Breite des Bildes auf drei Zeilen und drei Spalten auf. Daher erfolgen die beiden Divisionen der Auflösungen durch drei. Die Höhe der Titelleiste wird weiterhin einmalig pro Fenster subtrahiert. Aufgrund der drei Zeilen wird somit die Höhe der Startleiste durch drei dividiert.

$$window_{resolution}[i] = \left[\frac{resolution_{width}}{3}, \frac{resolution_{height}}{3} - title\_bar_{height} - \frac{taskbar_{height}}{3}\right] \quad (6)$$

mit:

window_resolution[i] : Fenstergrößen für (i+1) Fenster als Liste ‚[X,Y]' bei i = 4, 5, … , 8

3.2.5 Klassen und Methoden

Zur Verwaltung der Geräte existiert in dem Ordner *classes* eine Klasse mit der Bezeichnung *Device* und dem Dateinamen *device.py*. Sie besitzt das Klassenattribut *_counter* für die Anzahl der vorhandenen Geräte.

Zum Anlegen dieser Geräte enthält sie als Methode einen Konstruktor, der alle benötigten Parameter bereitstellt und als Instanzvariablen anlegt. Bei diesen Parametern handelt es sich um *self* (für den Bezeichner), den Namen des Geräts, die Seriennummer, das Modell (beispielsweise Quest 2), die IP-Adresse und die drei Statusinformationen. Zu diesen zählen die Erreichbarkeit, Verbundenheit und Auswahl. Zudem wird das Klassenattribut _counter beim Initialisieren inkrementiert.

Weiterhin ist dort eine statische Methode *reset_counter* zum Zurücksetzen der Geräteanzahl beispielsweise bei Neuerstellung der Gerätelisten enthalten.

3.3 Anpassung des Bildschirmspiegelungstools scrcpy

Bei dem Bildschirmspiegelungstool *scrcpy* fehlten zwei Funktionalitäten, die benötigt werden und somit als Erweiterungen implementiert wurden. Zum Ersten sollen mit geringem Konfigurationsaufwand mehrere Geräte als nur eines gespiegelt werden können. Zweitens soll durch einen Doppelklick auf das gespiegelte Bild, selbiges maximiert bzw. umkehrend wieder minimiert werden. Es wurde daher geprüft, ob die Funktionalitäten dort nachträglich in der Programmiersprache C implementiert werden können.

Für diese erste Erweiterung zur Ausführung mehrerer Bildschirmspiegelungen wurde in der Hauptdatei unter *app\src\main.c* die Bibliothek *phtread* verwendet und vorerst zwei Thread getestet. Dazu wurden exemplarisch über die Zuweisung der Variable *args.opts.serial* zu zwei Seriennummern für die Spiegelungsgeräte (eine Seriennummer je Thread) manuell hinterlegt. Daraufhin kann *scrcpy* als Thread gestartet werden. Für die korrekten Zuordnungen von *scrcpy* bei mehreren Instanzen wurde in der Header-Datei *app\src\srcpy.h* in Zeile 10 die *scrcpy* Funktion von "scrcpy(struct scrcpy_options *options)" zu "scrcpy(struct scrcpy_options *options, int tid, SDL_Event event)" um eine Thread-ID und das SDL-Event erweitert. Dies wird dementsprechend beim Aufruf in der Hauptdatei *main.c* verwendet.

Außerdem wurde die Veränderung mit dem Maximieren der Fenster durch den Doppelklick derart implementiert. Die Funktionalität wurde durch das Hinzufügen von den in Abbildung 6 folgenden Zeilen im Quellcode des Eingabe-Managers von *scrcpy* unter *scrcpy-edit\app\src\input_manager.c* und anschließendem Build erzeugt.

```
673.    // on double-click in the window (not the outside/ black
        borders), switch to fullscreen
674.    else if (!controller && down) {
675.     printf("Double-click registered, switching to fullscreen!\n");
676.     sc_screen_switch_fullscreen(im->screen);
677.    }
```

Abbildung 6: Erweiterung des Quellcodes von scrcpy im Eingabe-Manager zur Maximierung der Spiegelungsfenster

Diese Zeilen prüfen, ob mit „!controller“ die Fernsteuerung unterbunden wird (somit das Argument „--no-control“ oder „-n“ verwendet wurde) und mit *down* die Maustaste gerade betätigt wird. Ist die Prüfung erfolgreich so wird mit der Funktion „sc_screen_switch_fullscreen(im->screen)“ die Anwendung in den Vollbildmodus maximiert. Diese Funktion wurde aus der Tastenkombination „Strg+F“ für die Umschaltung zwischen Fensteransicht und Vollbildmodus entnommen.

Die Abfrage der linken Maustaste und des Doppelklicks erfolgte bereits in einer vorher bestandenen *if*-Anweisung darüber mit „if (event->button == SDL_BUTTON_LEFT && event->clicks == 2)“ und konnte daher beibehalten und mit der eben genannten Funktion zur Bild-Umschaltung erweitert werden.

3.4 Konfiguration der Endgeräte

Die Konfiguration der Endgeräte kann manuell nach Anschluss an dem PC mit einem USB-Kabel über die Kommandozeile mit der ADB-Schnittstelle erfolgen. Dort muss nach Akzeptieren des Gerätezugriffes der Befehl *adb tcpip 5555* für jedes Gerät ausgeführt werden. Zur Vereinfachung existiert eine frei verfügbares Windows Stapelverarbeitungsdatei (ein Batch-Skript) zur Konfiguration der Geräte [48]. Dies ist als erste Version im *scrcpy-handler* unter ‚scrcpy-handler\Skripts\OriginalAdbWifi\AdbWifi.bat‘ einsehbar und besitzt die Funktionalität, nach Aufruf auf den Anschluss eines Gerätes zu warten, dieses anschließend zu konfigurieren und sich dann zu beenden.

Für die vorliegende Anwendung läuft das Tool im Hintergrund, überprüft die Konfiguration bzw. Verbindung zu angeschlossenen Geräten und nimmt die Veränderung der Konfiguration erst bei einem Fehlversuch vor. Diverse Status und Aktionsinformationen werden ausgegeben. Anschließend wird zwei Sekunden gewartet und von vorne mit der Überprüfung begonnen. Diese veränderte Variante wird in Anhang B: ADB TCP/IP Konfigurator dargestellt. Die erfolgten Veränderungen werden nachfolgend zusammengefasst.

In den Zeilen 3 bis 4 ist die Quelle der Rohfassung angegeben. Zudem wird in Zeile 12 der Standard-Port zur Konfiguration der Geräte mit der Nummer 5555 anstelle der vorherigen 5557 verwendet. Bei dem Aufruf des Skripts wird eine neue Beschreibung dessen Ausgaben, die in den Zeilen 14 bis 17 liegt. Zeile 18 ist der beschreibende Kommentar zur neuen Marke in der darauffolgenden Zeile. Diese Marke *start* führt in Zeile 33 mit dem Kommando *goto start* dazu, dass am Ende eines Programm-Durchlaufes die Ausführung an ihr fortgesetzt wird. In Zeile 22 bei der *for*-Schleife ist das ADB Argument '-d' hinzugekommen. Dieses führt das entsprechende Kommando zur Abfrage der IP-Adresse nur bei USB Geräten aus und verhindert es somit bei über TCP/IP verbundenen Geräten. Die Zeilen 24 bis 31 sind neu hinzugekommen, um die Verbindungsüberprüfung durchzuführen und die Ausgabe zu filtern. Die *for*-Schleife in Zeile 24 führt mit der Option ‚/F‘ und dem Argument ‚tokens=1-2‘ dazu, dass von der Rückgabe des enthaltenen *adb connect* Kommandos nur die ersten beiden Wörter in der Variable *Response* abgelegt werden. Zur Fehlersuche kann diese Variable durch das Entfernen der Kommentar-Anweisung in der Zeile 25 ausgegeben werden.

Die Zeilen 26 bis 32 filtern mit *if*- und *else*-Anweisungen diese Ausgabevariable. Sie geben dabei zusätzliche Informationen zum Verhalten des Programmes und zur weiteren Handhabung für den Anwendenden aus. Zeile 33 fügt eine Wartezeit von zwei Sekunden ein, bis mit Zeile 33 der nächste Durchlauf gestartet wird.

Diese Neuerungen erzeugen einen endlosen Durchlauf, bis der Anwendende das Programm mit der Unterbrechung ‚Strg+C' die Ausführung unterbricht oder das Fenster schließt. Ohne derartige Änderungen würde sich das Skript nach der ersten Geräte-Konfiguration beenden, was im vorliegenden Fall bei mehreren Geräten unpraktikabel ist.

3.5 Softwaretests und Code-Qualität

Zur Überprüfung der Funktionalität der entwickelten Software werden mehrere manuelle Testfälle entwickelt und durchlaufen.

Hierbei werden alle einzelnen Funktionen der Software-Tools getestet und in einem Excel-Dokument festgehalten.

In der ersten Testkategorie wird geprüft, ob die Funktion der Hauptanwendung gegeben ist. Dazu zählt mit der erste Unterkategorie die Menüführung einschließlich der Erstellung und dem Auslesen der Konfiguration.

In der zweiten Unterkategorie wird die Bildübertragung der ein bis neun Geräte und korrekte Darstellung der einzelnen Fenster getestet.

In Kategorie zwei wird das Spiegelungstool *scrcpy* getestet. Hier ist in der ersten Unterkategorie entscheidend, ob die Fenster und Inhalte korrekt dargestellt werden. Dazu zählt die Bildübertragung aller Geräte, ob sie mit ausreichender Geschwindigkeit (ohne große Verzögerung) und akzeptabler Qualität gegeben ist.

Die zweite Unterkategorie stellt dar, ob das benötigte Bild, je nach Gerät, übertragen und korrekt geschnitten wird. Aufgrund der doppelten Bilder bei den VR-Brillen wird dort das zweite Bild und der Ausschnitt der Augen abgeschnitten.

Die dritte Kategorie zeigt auf, ob die Funktionalität der Änderung von *scrcpy* gegeben ist.

In der Administratoroberfläche sind aufgrund des Menüs drei Tests notwendig. Diese decken die Menüführung, Konfigurationserstellung und das Erstellen bzw. Laden der Konfigurationen ab. Für die korrekte Größenbestimmung und Ausrichtung der Fenster folgen neun weitere Tests.

Für das Spiegelungstool *scrcpy* sind weitere Tests notwendig. Diese enthalten den Verbindungsaufbau mit der Bildübertragung (Tests für die Funktion, Qualität

und Geschwindigkeit), die Differenzierung der Geräte (Quest 1, Quest 2 und anderweitigen Geräten wie z. B. Smartphones), den Test der Bildzuschnitte bei den drei verschiedenen Geräten und zuletzt den Wechsel zwischen Fenster- und Vollbildmodus in beiden Richtungen durch das Doppelklicken auf das Bild als letzte Tests.

Die vorab genannten Testfälle ergeben in Summe eine Anzahl von 23 Tests. Sie wurden mit dem letzten Stand der Software vom 07.08.2022 uneingeschränkt bestanden und anhand der Vorlage in Anhang C: Manuelle Testfälle für den Scrcpy-Handler dokumentiert.

In Bezug auf die Code-Qualität wurde in Kapitel 3.2.2 das verwendete Modul *pylint* beschrieben. Bei der Überprüfung mit dieser statischen Quellcodeanalyse wird mit 9,96 von 10 Punkten eine sehr gute Bewertung erzielt, wie folgend in Abbildung 7 dargestellt.

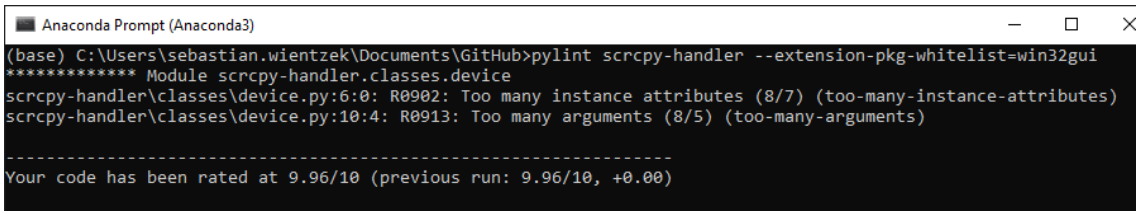

Abbildung 7: Statische Quellcodeanalyse mit dem Python Modul Pylint

Die dort dargestellten zwei Hinweise „Zu viele Instanz-Attribute“ und „Zu viele Argumente“ in der Klasse Device lassen sich nicht ohne Weiteres lösen, da für die Geräte als Klassenobjekte derart viele Informationen benötigt werden. Die Argumente in der Konstruktor-Methode würden sich kürzen lassen, indem dort die Geräte beispielsweise mit dem Namen, der Seriennummer und IP-Adresse erzeugt werden. Zusätzliche Informationen wie das Modell und die Statusinformationen zur Erreichbarkeit, Verbindung und Auswahl würden sich mit weiteren Methoden hinzufügen lassen. Dies steigert jedoch in den anderen Bereichen, beim Hinzufügen der Geräte nach dem Scan oder Auslesen aus der Konfiguration, die Komplexität und wurde daher nicht weiter betrachtet.

4 Usability-Test

Die Anwendung wurde in einem Praktikum mit N = 20 Studierenden in fünf Terminen evaluiert und dessen Vor- und Nachteile bewertet. Hierbei waren weibliche und männliche Teilnehmer vorhanden. Im Schnitt haben pro Termin vier Studierende an der Anwendung und den Fragebögen teilgenommen.

Bei dem Praktikum handelt es sich um ein Pflichtpraktikum aus dem Modul „Erneuerbare Energiesysteme“, das von dem Labor Virtuelle Realität PV handelt. Beispielhaft wird in Abbildung 8 ein Foto von dem PV-Praktikum dargestellt. Es zeigt die Verwendung der VR-Brillen mit Studierenden und laufenden Bildschirmspiegelungen zur Unterstützung und Anleitung.

Abbildung 8: Foto bei der Verwendung der VR-Brillen mit Studierenden und laufenden Bildschirmspiegelungen zur Unterstützung und Anleitung innerhalb eines Praktikums

Hierfür wurde ein Fragebogen erstellt und ausgewertet. Zur Untersuchung der verschiedenen Bereiche wurden insgesamt 17 bewertbare und sechs offene Fragen in folgenden sechs Kategorien gestellt. Die Bewertung erfolgte auf einer 5-stufigen Likert-Skala (beginnend bei 1=“stimme gar nicht zu“ bis hin zu 5=“stimme voll und ganz zu“).

Zur Vorerfahrung, sowie dem Interesse an VR-Anwendungen und PV-Anlagen wurden drei Fragen gestellt. In Bezug auf die Einrichtung, Anwendung der VR-Hardware (VR-Brillen) und Menüführung (Bedienung einschließlich der Bewegung) jeweils Eine. Vier Fragen zielten auf die einzelnen Funktionalitäten der Anwendung ab. Fünf Fragen bezogen sich auf didaktische Aspekte, wie die Verständlichkeit der Aufgabenstellung und die erfolgte Zusammenarbeit. Abschließend konnte auch der gesamte Eindruck bewertet werden.

Zu den Themen Fragen und Probleme zur bzw. mit der Anwendung wurden offene Fragen erstellt. Diese wurden in Bezug auf die Problemlösung, zu Besonderheiten bei der Kommunikation und zur Zusammenarbeit in der Gruppe erweitert. Auch allgemeine Verbesserungsvorschläge wurden offen angefragt.

Der komplette Fragenkatalog für die Studierenden wird in Anhang D: Fragenkatalog für das Praktikum dargestellt. Im Folgenden wird er in die einzelnen Bereiche aufgeteilt und die Nutzererlebnisse anhand der Mittelwerte (M) und Standardabweichungen (SD) bewertet.

In folgender Tabelle 1 wurde die Vorerfahrung mit VR-Anwendungen, PV-Anlagen, sowie das Interesse an erneuerbaren Energien bewertet. Hier zeigt sich deutlich, dass aufgrund des Mittelwertes 1,9 überwiegend keine Erfahrungen mit VR-Anwendungen bestanden. Hier ist jedoch mit 1,2 der größte Wert der Standardabweichung vorhanden. Dieser zeigt, dass das Ergebnis im Durchschnitt um +/-1,2 variiert. Die meisten Studierenden hatten somit keine Vorerfahrungen in der VR. Die Vorkenntnisse zu PV-Anlagen wurden mit mittelmäßig bewertet und auch die Abweichung ist mit 0,8 gering.

Tabelle 1: Auswertung der Vorerfahrung, sowie des Interesses an VR-Anwendungen und PV-Anlagen

Haben Sie…		
(1=stimme gar nicht zu; 5=stimme voll zu)	**M**	**SD**
Erfahrung mit VR-Anwendungen?	1,9	1,2
Vorkenntnisse zu PV-Anlagen?	3,2	0,8
Interesse an erneuerbaren Energien und PV-Anlagen?	4,5	0,7

Das Interesse an erneuerbaren Energien und PV-Anlagen liegt dagegen mit 4,5 im Mittelmaß zwischen „stimme voll und ganz zu“ und „stimme zu“. Es deutet demnach auf ein ordentliches Interesse und mit 0,7 SD auch auf geringe Abweichungen hin.

In der nachfolgenden Tabelle 2 wurde die Einrichtung der VR-Brillen mit 4,1 und einer Abweichung von 0,8 daher überwiegend gut bewertet.

Tabelle 2: Auswertung der Einrichtung

Funktionierte…		
(1=stimme gar nicht zu; 5=stimme voll zu)	**M**	**SD**
die Einrichtung der VR-Brillen gut (Guardian/ Festlegung Spielbereich)?	4,1	0,8

Die Inhalte der Tabelle 3 verdeutlichen die Erfahrungen mit der Bedienung der Anwendung. Dazu gehört die Bewegung innerhalb, die Drehung und die Menüführung. Diese Punkte wurden mit 3,8 bis 4,2 bei einer geringen SD von 0,6 bis 0,7 bewertet und liegen daher in einem guten Bereich.

Tabelle 3: Auswertung der Bedienung, Bewegung, Drehung und Menüführung

Funktionierte…		
(1=stimme gar nicht zu; 5=stimme voll zu)	**M**	**SD**
die Bedienung (Laufen, Drehen, Menüführung) gut?	4,2	0,6
Bewegung in der VR?	3,8	0,7
Menü verständlich aufgebaut und gut strukturiert?	4,2	0,7

Die Drehung wurde zudem damit kommentiert, dass sie teilweise nicht korrekt funktionierte (ein zu großer Drehwinkel vorlag), damit zu schnell und die Bewegung selber hingegen oft zu langsam ablief. Die Drehung führt häufig zu einem Schwindelgefühl und Übelkeit. Es soll angenehmer sein, sich in der Realität selbst mitzubewegen. Zukünftig ist dementsprechend angedacht, die softwareseitige Drehung über den Controller zu deaktivieren und die manuelle Drehung über die Kopfbewegung zu nutzen.

Neben dem ordentlichen Interesse an erneuerbaren Energien, sowie PV-Anlagen in Tabelle 1, ist in Tabelle 4 mit 4,4 die gute, verständliche und schnelle Unterstützung ersichtlich, die mit einer SD von 0,5 die geringste Abweichung erhielt.

Tabelle 4: Auswertung der Unterstützung bei der Anwendung

War die…		
(1=stimme gar nicht zu; 5=stimme voll zu)	**M**	**SD**
Unterstützung gut, verständlich und schnell?	4,4	0,5

In Tabelle 5 wurde die Bewertung der Funktionalitäten dargestellt. Die Verständlichkeit ist mit 3,8 gut gelungen. Sie weist jedoch wie das Platzieren, Rotieren und Neigen von PV-Modulen mit 3,6 eine leichte Tendenz ins neutrale auf. Beide Werte unterliegen mit einer SD von 0,9 bis 1,0 einer geringen Abweichung und tendieren daher zwischen gut und neutral.

Die Funktion des Andockens ist mit 3,3 etwas stärker neutral bewertet worden, ferner ist die Abweichung mit 1,1 etwas größer. Hier lässt sich anhand der Kommentare aufzeigen, dass das Andocken mit den Einzelaufständerungen im Gegensatz zu den Doppelaufständerungen gut geklappt hat. Die Funktion bei den Doppelaufständerungen ist zu der Zeit aufgrund der Komplexität und des stark abweichenden Verhaltens im Gegensatz zur Einzelaufständerung nicht implementiert. Dies lässt einen Zusammenhang zu dieser Bewertung vermuten. Die Verschaltung ist mit dem Mittel von 4,0 und der geringen SD von 0,8 im guten Rahmen einzuordnen.

Tabelle 5: Auswertung der Funktionalitäten der Anwendung

Waren die Funktionalitäten zur/ zum…		
(1=stimme gar nicht zu; 5=stimme voll zu)	**M**	**SD**
PV-Anwendung verständlich?	3,8	1,0
PV-Module platzieren, rotieren und neigen?	3,6	0,9
Andocken gut?	3,3	1,1
Verschaltung gut?	4,0	0,8

Nachfolgend wird die Bewertung didaktischer Aspekte dargestellt (Tabelle 6). Die Abweichungen liegen mit 0,7 bis 0,9 im geringfügigen Rahmen. Sie werden aus diesem Grunde zusammengefasst und nicht näher betrachtet.

Tabelle 6: Auswertung didaktischer Aspekte und des Gesamteindruckes

Wie gut war…		
(1=stimme gar nicht zu; 5=stimme voll zu)	**M**	**SD**
Verständlichkeit der Aufgabenstellung?	4,1	0,9
Konzentration auf die Aufgabenstellung?	3,6	0,8
die Umsetzbarkeit der Aufgabe?	3,6	0,9
die Zusammenarbeit?	3,8	0,8
der Gesamteindruck?	4,0	0,7

Die Verständlichkeit der Aufgabestellung wurde mit 4,1 gut erreicht. Die Konzentration auf die Aufgabenstellung ist mit 3,6 eher gut einzuordnen. Anhand der Kommentare könnte diese Bewertung aufgrund der zu großen Gruppen (Empfehlung der Studierenden liegt bei 2-3 Personen) derart ausgefallen sein.

Weiterhin ist die Umsetzbarkeit der Aufgabe mit 3,6 ähnlich bewertet worden. Hier wurde angemerkt, dass es eine vordefinierte und in der Anwendung dargestellte Aufgabenstellung geben sollte. Dies wäre eine Erweiterung der bisherigen Vorgehensweise, weil die Aufgabenstellungen vorab an die Studierenden verteilt wurden. Eine bessere Verständlichkeit der Aufgabe und das Vertraut machen in die Anwendung wurden weiterhin als Anmerkungen in den Freitextfeldern angegeben.

Eine mögliche Überlegung dazu wäre, neben einer Demonstration der Funktionalitäten, kleine Aufgaben zu jeder einzelnen Funktion der Anwendung durchführen zu lassen.

Die Zusammenarbeit ist mit dem Mittel von 3,8 ebenfalls gut bewertet worden, laut Freitext ist jedoch nicht immer die gesamte Gruppe bei der Bearbeitung der Aufgabenstellung beteiligt und es werden wie oben kleinere Gruppen empfohlen.

Der Gesamteindruck ist mit 4,0 gut bewertet worden, liegt mit der Abweichung von 0,7 in dieser Kategorie am geringsten. Insgesamt ist dies der zweitniedrigste Wert bei der Abweichung. Der Gesamteindruck schwankt zwischen neutral und sehr gut.

Wesentliche Probleme haben sich mit dem Tracking und Guardian bei den neuen Oculus Quest 2 VR-Brillen ergeben. Diese Fehlerquellen können durch die Sonneneinstrahlung in das Büro entstehen. Diese Vermutung liegt nahe, weil das Modell Quest 2 durch einen anderen Aufbau stärker für schwache Lichtverhältnisse optimiert sein soll. Unter stärkeren Sonnenverhältnissen konnten im Außenbereich unter Pavillons sehr schlechte Trackingeigenschaften festgestellt werden. Eine Bedienung war dort nicht möglich.

Zudem gab es am Anfang Probleme durch ein plötzliches Updaten der VR-Brillen mitten in der Anwendung. Dies ist vermutlich auf die geringfügige Zwischenlagerung zurückzuführen, weil ein notwendiges Update von Oculus durchgeführt wurde.

Insgesamt gab es folgende positive Kritik. Die neue Technik bietet viel Spaß beim Lernen, sowie eine gute Erfahrung und mit der Lernanwendung einen guten Einblick zur Realität. Auch die Unterstützung ist sehr gut gelungen.

Negativ dagegen viel auf, dass ein Konzentrationsnachlass nach etwa einer Stunde VR-Nutzung auftrat, wodurch häufigere Pausen notwendig waren. Leichte Probleme gab es mit der Bedienung und Menüführung, sodass PV-Module ungewollt bei der Menübedienung entfernt wurden. Zudem mussten beispielsweise Probleme mit dem Guardian oder dem Tracking teilweise durch einfaches Aus- und Wiedereinschalten der VR-Brille (kurzes in den Standby-Modus

versetzen) gelöst werden. Größere Probleme gab es mit der grafischen Darstellung innerhalb der VR-Brille nach den Updates. Sie mussten durch einen kompletten Neustart der Brillen gelöst werden. Vermutlich wurden dort die Grafiktreiber geupdatet. Alternativ könnte auch die Aktualisierung des Nachrichtenkanals (*Feeds*) oder die Umstellung zum Multitasking-Modus im *Quest Build v39* vom 11. April 2022 dazu geführt haben [49]. Nach eigener Überprüfung ist anzumerken, dass die Updates innerhalb von Apps normalerweise unterbunden werden. Gegebenenfalls wird das Verhalten noch nach Dringlichkeit differenziert. Zudem ist anzumerken, dass die Oculus Versionshinweise im deutschen seit Version 38 vom 07. März 2022 nicht mehr geupdatet werden. Die Versionshinweise bis derzeit Version 43 müssen demnach über die Auswahl der amerikanisch-englischen Sprache eingesehen werden.

Als Wünsche und Anmerkungen wurde angegeben, dass die Orientierung sichtbar gemacht werden soll. So soll beispielsweise mittels eines Kompasses die Nord- und Südausrichtung dargestellt werden. Zudem könnten die technischen Daten der Module und Wechselrichter innerhalb der Anwendung an notwendigen Stellen einsehbar sein. Weiterhin sollten die Räume in der Anwendung für Pausen mit vorhandener Konfiguration zwischengespeichert werden können, oder die VR-Brillen nicht zu lange in den Standby-Modus verfallen. Zusätzlich wurde ein Sprachchat vorgeschlagen, der jedoch bereits implementiert ist. Hier ist fraglich, ob der Sprachchat bei einer Anwendung in der Gruppe vor Ort zu Vorteilen führen kann oder eher zu Nachteilen. Ersten Tests zufolge ergab sich aufgrund der Latenz ein Echo in der Kommunikation, sodass die anderen Anwendenden vor Ort zeitnah gehört wurden und mit geringer zeitlich Verzögerung erneut über das Headset. Auch die Verwendung von On-Ear-Kopfhörern mit Ohrmuscheln verbesserte dieses Verhalten nicht.

5 Ergebnis

In Kapitel 3.3 wurden die Anpassungen des Tools scrcpy beschrieben. Im ersten Teil ging es um den Versuch, das Tool in der Programmiersprache C um die Funktionalitäten der Mehrfachspiegelungen zu erweitern. Das Resultat des Vorgangs bestand darin, dass weiterhin ein Gerät ordnungsgemäß gespiegelt werden kann und für das zweite Gerät auch versucht wird, eine Verbindung aufzubauen. Dies gelang soweit, dass in der Datei server.c die Funktion "server->cbs->on_connected(server, server->cbs_userdata);" aufgerufen wird und über die Datei *scrcpy.c* mit "PUSH_EVENT(EVENT_SER-VER_CONNECTED);" zur Aktivierung eines Events führt. Aus unerklärlichen Gründen wurde es jedoch in der Funktion "await_for_server(SDL_Event event) unter dem Fall "case EVENT_SERVER_CONNECTED" nicht mehr angenommen. Aufgrund der Komplexität des Tools und des unergründlichen Fehlers war es nicht möglich, die für die Administratoroberfläche notwendigen Funktionalitäten in dem Tool selbst zu implementieren. Daher wurde diese Variante verworfen.

Das stattdessen aus dieser Arbeit hervorgehende implementierte Ergebnis besteht hauptsächlich aus der in Kapitel 3.2 beschriebenen Administratoroberfläche als Hauptanwendung, dem *Scrcpy-Handler*. In diesem lassen sich die verschiedenen Menüpunkte auswählen und starten. Da im Laufe der Tests und Evaluation weitere hilfreiche Änderungen notwendig waren, sind zwei unterschiedliche Versionen entstanden. In Version 1 gibt es zwei Menüpunkte, die über die Kommandozeile gestartet wurden. In Version 2 existieren dagegen drei Menüpunkte, die über eine grafische Oberfläche auswählbar sind.

In nachfolgender Abbildung 9 wird die erste Version demonstriert. Über die Eingabe der Ziffern 1 oder 2 wird die Menüauswahl getätigt. Mit der 1 erfolgt die Überprüfung nach Geräten (der Scanvorgang) über den USB-Anschluss.

Ist bereits eine Konfiguration vorhanden, werden durch Eingabe der 2 automatisch die Bildschirmspiegelungen anhand der vorliegenden Konfiguration gestartet.

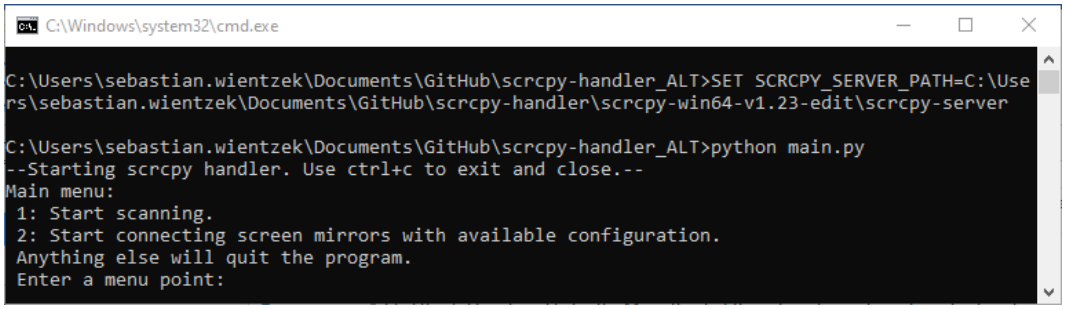

Abbildung 9: Version 1 des Scrcpy-Handlers

Durch die erste Ausführung, sowie der nahezu einmaligen Konfiguration der Geräte und der Oberfläche, ist die erste Version derart entstanden. Zudem hat sie bei den ersten Praktika ihre Funktionalität bewiesen. Trotzdem hat auch diese Version Optimierungspotenzial aufgezeigt.

Dieses Potenzial bestand darin, den VR-Brillen feste IP-Adressen zuweisen zu lassen. Dies vereinfacht die Zuordnung der Geräte erheblich, da bereits bei den ersten Verbindungen anhand dieser Adresse, eine feste Zuordnung zur ID der VR-Brille ausgeführt werden kann. Vorab war dies nur durch einen gesteigerten Aufwand, über eine Zuordnung der Seriennummern der Brillen in Python in einem Dictionary und dem Auslesen dieser Nummern über den USB-Anschluss mittels der ADB-Schnittstelle möglich.

Zusätzlich kam die Überlegung auf, eine Vorauswahl der Geräte zur Spiegelung anhand der Verbindungsüberprüfung zu treffen und die endgültige Auswahl den Anwendenden zu überlassen.

Dieses sehr vereinfachte Verhalten wurde in Version 2 umgesetzt und wird folgend in Abbildung 10 demonstriert.

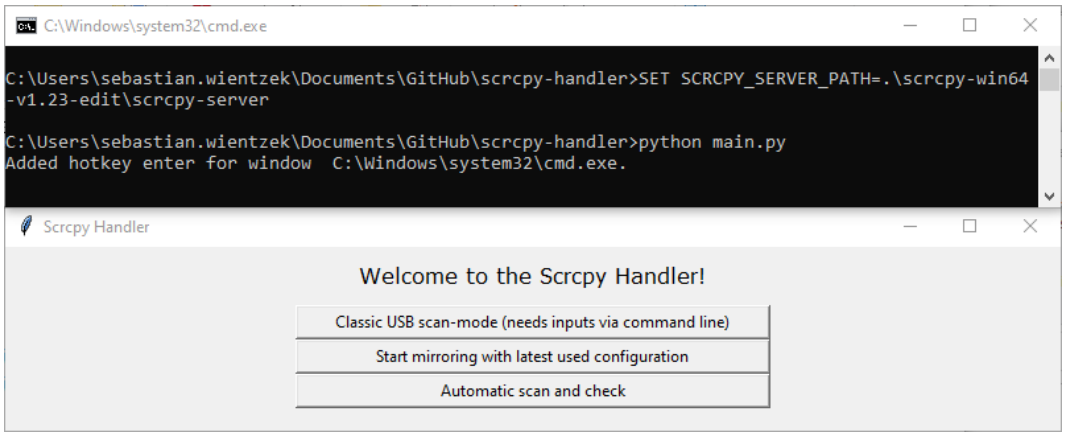

Abbildung 10: Version 2 des Scrcpy-Handlers

Hier wird direkt die Auswahl der 3 Menüpunkte ermöglicht. Als erster Punkt wird die klassische Gerätesuche über den USB-Anschluss mittels Kommandozeile,

als Zweites die Verwendung der letzten Konfiguration oder als letzten Punkt der automatische Scan- und Prüfvorgang ermöglicht.

Bei Verwendung des ersten Menüpunktes erfolgt die standardgemäße Text Aus- und Eingabe weiterhin über die Kommandozeile um die alten Funktionalitäten im Bedarfsfall beizubehalten. Dieses Verhalten ist in folgender Abbildung 11 ersichtlich. Wird ein Gerät gefunden, kann in der Kommandozeile ein Name vergeben werden und anschließend mit der Eingabetaste die Suche weiter durchgeführt werden. Mit erneuter Betätigung der Eingabetaste wird die Suche beendet und es erfolgt die Bildschirmspiegelung mit allen gefundenen Geräten. In der grafischen Oberfläche ist es jederzeit möglich, mittels der Taste „Zurück“ wieder in das Hauptmenü zu gelangen.

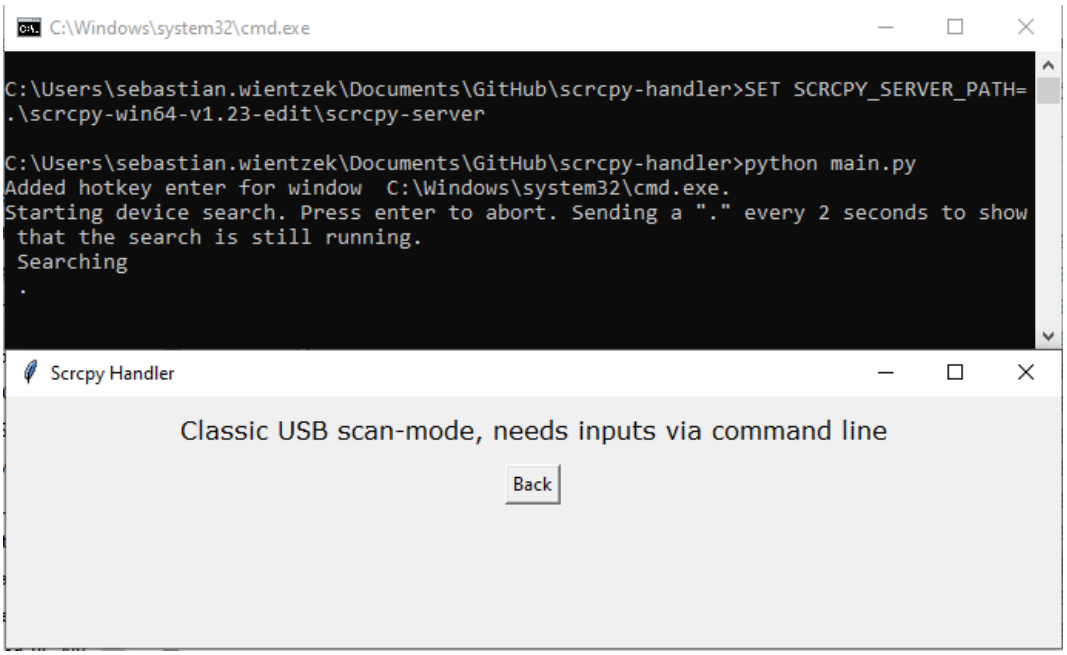

Abbildung 11: Menüpunkt 1 des klassischen Scanvorgangs in Version 2 (Kombination aus grafischer Oberfläche und Kommandozeile)

Wird der nächste Menüpunkt ausgewählt, so öffnet sich das Menü zur Verwendung der letzten verwendeten Konfiguration (Abbildung 12). Hier werden die Geräte tabellarisch aufgelistet. Dies erfolgt mit der jeweiligen Beschriftung, der IP-Adresse, dem Status der Erreichbarkeit, dem Konfigurationsstatus, sowie der Auswahlmöglichkeit zur Bildschirmspiegelung. Um dieses Menü so schnell wie möglich verwenden zu können, wurde auf die Überprüfung der Verbindungen verzichtet. Deshalb sind die Statusfelder orange dargestellt. Wurde bei der letzten Anwendung die Spiegelung durchgeführt, wird sie beim nächsten Start wieder über die Kontrollkästchen ausgewählt (siehe letztes Gerät mit der Nummer 13).

Anschließend bestehen die Möglichkeiten, den Status zu aktualisieren (mit der Schaltfläche ‚Refresh') oder mit Start die Bildschirmspiegelungen der ausgewählten Geräte zu starten. Mittels zurück wird das Hauptmenü wieder dargestellt. ‚Quit' führt zur Beendigung des Programmes.

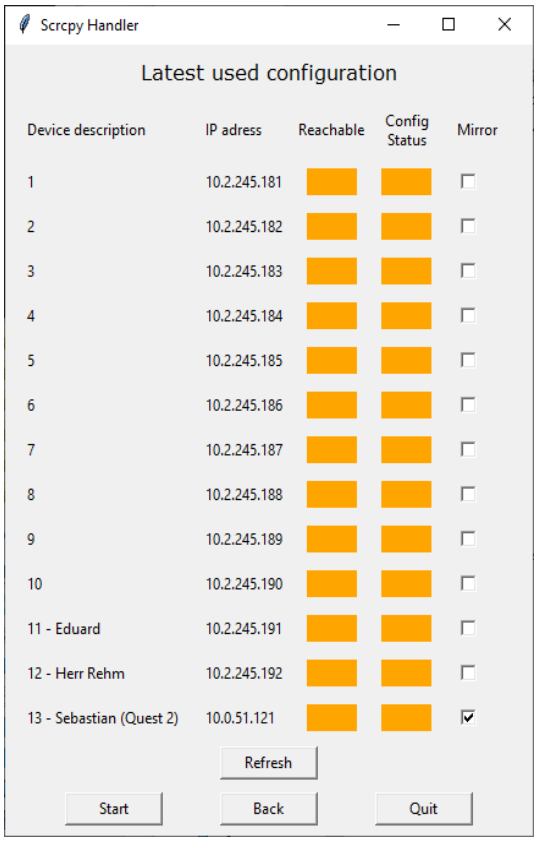

Abbildung 12: Menüpunkt 2 der neuen Übersicht zur Verwendung der zuletzt eingesetzten Konfiguration

Bei einer Aktualisierung erfolgt der automatische Wechsel zum folgenden Menüpunkt 3, dem automatischen Scan- und Überprüfungsvorgang (Abbildung 13). Dieses ist vom Aufbau her identisch zum zweiten Menüpunkt, unterscheidet sich jedoch vom Ablauf. Vor der Darstellung der Geräte wird der Scan- und Überprüfungsvorgang durchgeführt und anschließend der jeweilige Status, bei erfolgreicher Verbindung grün, oder bei fehlgeschlagener Verbindung in Rot, dargestellt. Ist die Konfiguration der Geräte in Ordnung, wird die Vorauswahl zur Bildschirmspiegelung automatisch getroffen. Ist das Gerät erreichbar, die Konfiguration aber nicht in Ordnung, so muss das Gerät per USB an den Computer angeschlossen werden. Über ein separates Batch-Skript muss dann der TCP/IP-Modus für die ADB-Schnittstelle aktiviert werden. Die Beschreibung dieses Tools ist in Kapitel 6.4 zur Konfiguration der Endgeräte zu finden und das Tool selbst im Anhang B: ADB TCP/IP Konfigurator beigefügt.

Wie in der Abbildung gezeigt wird, sind nahezu alle Geräte nicht erreichbar, bis auf das Gerät 13. Daher wurde nur diese eine Vorauswahl zur Bildschirmspiegelung getroffen.

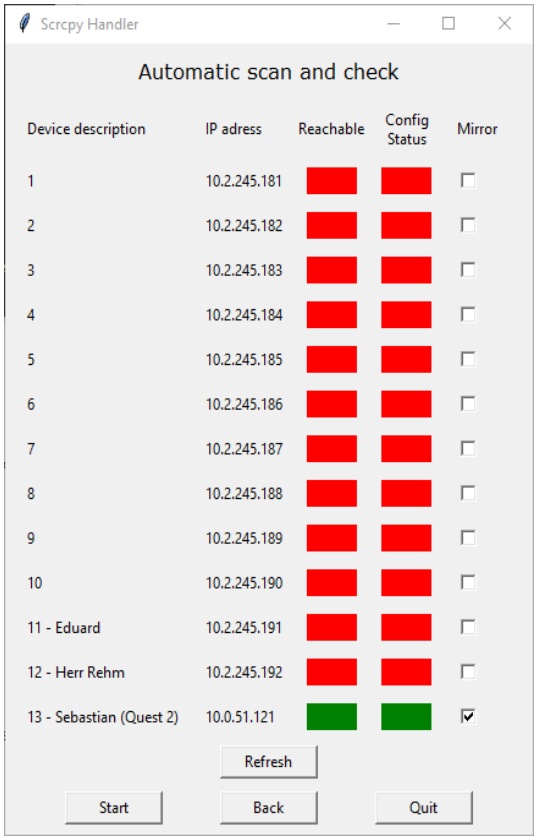

Abbildung 13: Menüpunkt 3 des neuen automatischen Scan- und Überprüfungsvorgangs

Wurde eine Bildschirmspiegelung gestartet, wird in diesem Menüpunkt verblieben. Zusätzlich werden die Bildschirmspiegelungen, wie in Kapitel 6.2.6 unter der Funktion *scrcpy_process_starter()* beschrieben, gestartet und ausgerichtet. Dies ist mit fünf Geräten exemplarisch bei Verwendung der VR-Anwendung innerhalb eines Praktikums in Abbildung 14 dargestellt. Dort ist ersichtlich, dass die Studierenden mit der Erstellung der PV-Anlage und mit der korrekten Auswahl der Module beschäftigt sind.

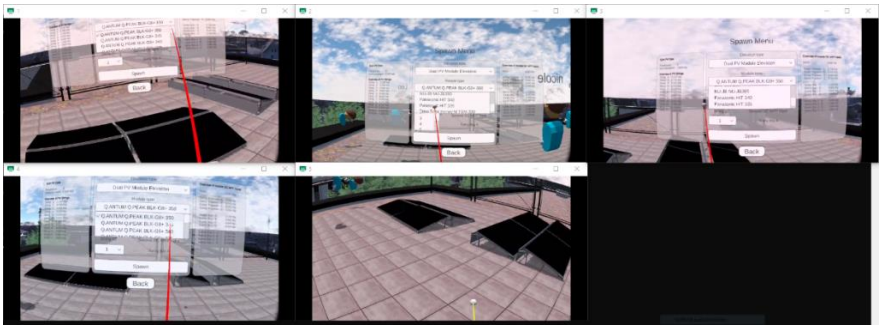

Abbildung 14: Exemplarische Darstellung der Bildschirmspiegelung mit fünf VR-Brillen beim Aufbau der PV-Anlage und der Auswahl der korrekten Module

6 Fazit

In diesem Kapitel erfolgt eine kurze Zusammenfassung dieser Arbeit, es wird ein Resümee gezogen und ein Ausblick auf mögliche Erweiterungen der entwickelten Administratoroberfläche gegeben.

6.1 Zusammenfassung

Bei dieser Arbeit wurden zwei Möglichkeiten zur Unterstützung der Studierende bei Photovoltaik PV-Praktika in der VR untersucht und beurteilt. Aufgrund der Fehleranfälligkeit, Komplexität und geringeren Funktionalität der anderen Option erfolgte die Entwicklung der nahezu automatisierten Administratoroberfläche. Die Entwicklung dieser Software wurde anhand der gängigen Phasen des Wasserfallmodells ausgeführt. Die Umgebung bietet mittels Bildschirmspiegelungen eine kostenfreie und einfache Möglichkeit zur didaktischen Anleitung und Unterstützung der Studierenden bei den verpflichtenden Praktika. Auch der Einsatz der Administratoroberfläche erfolgt bereits. Die Lehranwendung bietet einen praxisnahen Aufbau von Solaranlagen zur technischen Überprüfung, Simulation und Bewertung dieser Anlagen. Weiterhin wurde ein Usability-Test in den Praktika durchgeführt und abschließend evaluiert.

6.2 Resümee

Abschließend lässt sich Bezug auf die Zielsetzung nehmen. Hier ging es um die Erstellung einer konstruktiven Umgebung für die Lehre in der VR zur Anleitung und Unterstützung der Studierenden. Sie sollte mittels Bildschirmspiegelungen ausgeführt werden. Im Hinblick auf die Ergebnisse lässt sich das Resümee ziehen, dass dieses Ziel mit der Administratoroberfläche erreicht wurde. Zudem wurde sie von der intuitiven Bedienbarkeit her zwischen Version 1 und Version 2 wesentlich optimiert.

Es hat sich gezeigt, dass sich mit dieser Leitstand-Anwendung die verschiedensten Szenarien abbilden lassen. Es ist somit einerseits möglich, nur ein Gerät als Demonstrationsobjekt zu spiegeln. Es können bei Bedarf dennoch gleichzeitig beispielsweise fünf VR-Brillen der Studierenden und eine VR-Brille des Lehrenden gespiegelt werden. Derzeit ist die Anzahl der Geräte zur Spiegelung auf maximal neun Geräte beschränkt, um einen optimalen Kompromiss aus Bildgröße und Ausnutzung der Bildschirmfläche zu erlangen.

Die ersten Testdurchläufe mit Version 1 haben gezeigt, dass die Anleitung und Unterstützung insgesamt gut gelungen ist. Unerwartete Probleme, aufgrund der neuen Technik der VR-Brillen Oculus Quest 2 im Vergleich zur Oculus Quest 1, mussten kurzfristig festgestellt und so gut wie möglich gelöst werden.

Zudem hat sich dort gezeigt, dass die Ersteinrichtung, so wie die erneute Einrichtung und Verwendung der Oberfläche relativ problematisch war. Das dort gezeigte Verbesserungspotenzial hat sich direkt auf die zweite Version der Anwendung ausgewirkt und konnte bereits umgesetzt werden um die gesamte Anwendung wesentlich zu vereinfachen.

6.3 Ausblick

Für die Zukunft ist es im Rahmen der Implementierung vorstellbar, die Bildschirmspiegelung auch über mehrere Bildschirme verteilen zu können. Hierdurch können schnell bis zu 18 oder auch mehr Geräte dargestellt werden. Somit würden zwei Gruppen von jeweils fünf Studierenden ohne jegliche Probleme in Echtzeit unterstützt werden.

Zudem ist eine Erweiterung der Darstellung und Übertragungen denkbar und sinnvoll. Beispielsweise ist es eine nützliche Ergänzung, die aktuell verbleibenden Akku-Kapazitäten, die Spielernamen aus der Unity-Lernanwendung, und auch Funktionen zum Herunterfahren oder Neustarten der Geräte darzustellen. Hier müssen jedoch auf beiden Seiten (server- und clientseitig) Schnittstellen implementiert werden, die vorhandene angepasst oder eine weitere, eigenständige Anwendung in Unity entwickelt werden.

Eine andere Möglichkeit besteht darin, die vorhandene Anwendung für einen Lehrenden zu erweitern und beispielsweise auf dem Computer mittels Maus und Tastatur lauffähig zu machen, anstatt auf der VR-Brille. Darauf aufbauend kann diese Anwendung zur Erstellung der Räume, zum Ausführen bestimmter Aktionen wie dem Leeren des Raumes, oder Zurücksetzen zur Standardbelegung dienen. Zusätzlich können dort vorsorglich die verbleibenden Akkukapazitäten der VR-Brillen angezeigt werden, sowie dauerhaft eine Übersicht über die verwendeten Module, die Verschaltung und das Simulationsmenü eingeblendet werden.

Eine Ausweitung der Anwendung auf andere Bereiche, wie beispielsweise die Verwendung von Smartphones, AR-Geräten oder anderweitige VR-Anwendungen zu Lehrzwecken ist machbar.

Anhang A: Übersicht über die Implementierung

1 Extern verwendete Module und Funktionen

- **Adbutils** — Modul für die Kommunikation mit Android-Endgeräten über ADB
- **keyboard** — Modul für die Betätigung von Tastendrücken als Hotkey
- **logging** — Modul zur Informationsausgabe (für Entwickler oder Anwender)
- print() — Funktion zur Ausgabe von Informationen in der Standardausgabe (Konsole)
- **pylint** — Modul zur statischen Quellcode-Analyse
- **signal** — Modul zur Verwaltung asynchroner Ereignisse
  - signal() — Funktion zur Erkennung von Tastendrücken als Unterbrechungs-Ereignisse
- **subprocess** — Modul zum Zugriff auf Prozesse
  - append() — Funktion zum Hinzufügen von Prozessen als Objekte zu einer Liste
  - popen() — Funktion zum Starten des als Parameter erwarteten Befehls als Prozess
  - terminate() — Methode zum Beenden von Prozessobjekte
- **sys** — Modul für Systemweite Parameter und Funktionen
  - exit(0) — Funktion zum korrekten Beenden bei Unterbrechungs-Ereignissen
- **time** — Modul für die Verwendung von Zeitfunktionen
  - time.sleep() — Funktion zur Erzeugung kurzer Wartezeiten in Sekunden
- **Tkinter** — Modul für die Erstellung, Darstellung

- **win32gui** Modul zur Erkennung von Fenstertiteln, insbesondere aktiver Fenster
  - GetForegroundWindow() Erhalte das aktive Fenster (aus dem Vordergrund)
  - GetWindowText() Ausgabe des Fenstertitel eines übergebenen Fensters

2 Eigens implementierte Klassen, Module und deren Funktionen

- **adb_handler.py** **Kommunikation mit Android-basierten Endgeräten**
  - adb_con.connect() Prüft die ADB Verbindungen (den Konfigurationsstatus) zu den Geräten per IP-Adresse mit Standard-Port von 5555
  - check_or_add_usb_device() Prüfe ob das USB-Gerät bereits in der Konfiguration enthalten ist oder füge es hinzu
  - check_or_add_tcp_device Prüfe ob das TCP-Gerät bereits in der Konfiguration enthalten ist oder füge es hinzu
  - check_tcp_device() Prüft mit einem Ping über TCP/IP die Erreichbarkeit der Geräte und über ADB die Konfiguration
  - device_checker() Prüft die Erreichbarkeit und Konfiguration der Geräte in einer Schleife
  - device_search() Gerätesuche als Vorgang
  - device_search_loop() Start der manuellen Gerätesuche als Dauerschleife und Überprüfung der Eingabetaste
  - ping_device() Prüft die Erreichbarkeit eines Gerätes mittels Ping
- **config.py** **Konfigurationsdatei für die globalen Variablen**
- **device.py** **Modul zur Verwaltung der Geräte**
  - **Device** **Klasse zur Verwaltung der Geräte als Objekte**
  - **reset_counter()** **Statische Methode zum Zurücksetzen der Geräteanzahl**
- **input_controller.py** **Modul für Steuerungen und Eingaben**

- add_hotkey_enter() — Funktion zur Erstellung des Hotkeys Eingabetaste mit der Bibliothek *keyboard*
- choose_menu_point — Eingabemöglichkeit eines Menüpunktes in der Kommandozeile
- ctrl_c_exit_sigint_handler() — Überprüfung des Interrupts/ des Unterbrechungs-Ereignisses Strg+C
- handle_keypress — Überwachung der Eingabetaste als Event-Hotkey

- **json_config_handler.py** — Im- und Export der Konfigurationen im JSON-Format
  - read_config_from_json() — Funktion zum Auslesen der Konfiguration aus der JSON-Datei
  - write_config_to_json() — Funktion zum Abspeichern der aktuellen Konfiguration als JSON-Datei

- **main.py** — Hauptmodul für die Funktionalitäten und GUI
  - ScrcpyHandlerApp — Klasse für die Erzeugung und Bedienung der GUI
    - switch_frame() — Methode zum Wechseln der dargestellten Seite (des Menüs)
    - update_gui() — Methode zur manuellen Aktualisierung der GUI bei Durchführung anderer Funktionalitäten
  - autocheck_connected_devices() — Funktion zum Auslesen der aktuellen Geräteliste und Aktualisieren der Statusinformationen der Geräte
  - connect() — Verbindung der Geräte und Aktivierung der Gerätespiegelungen anhand einer vorhandenen Konfigurationsdatei
  - DeviceTable — Klasse zur Erstellung der Gerätetabelle

- main() — Hauptfunktionalität, erzeugt dem Benutzer eine Auswahl für das Hauptmenü (V1: Kommandozeile, in V2: GUI)
- PageOne — Klasse zur Darstellung der Seite 1 für die manuelle Gerätesuche über USB
- PageTwo — Klasse für die Seite 2 zur automatischen Gerätesuche und Differenzierung der Aktualisierung der Gerätestatus über Aktualisierungstyp (refresh_type)
  - all_states() — Methode zum Auslesen aller Kontrollkästchen
  - buttons() — Methode zur Erstellung der Menübuttons in den Untermenüs
  - refresh_function() — Aktualisierung der Geräteliste
- read_configuration() — Funktion zum Aufrufen der Auslesefunktion aus dem Modul json_config_handler
- scan() — Startet den Suchvorgang nach Geräten
- StartPage() — Erstellung des Hauptmenüs der drei Menüpunkte in der GUI
- state() — Methode zur Abfrage des Zustands eines Kontrollkästchens
- write_configuration() — Funktion zum Aufrufen der Speicherfunktion aus dem Modul json_config_handler

- **scrcpy_connector.py** — Verbindungsmodul zur Unteranwendung *scrcpy*
  - scrcpy_process_starter() — Funktion zum Starten der Bildschirm-Spiegelungs-Prozesse mit dem Tool *scrcpy* als Unteranwendungen
  - scrcpy_process_terminator() — Funktion zur Beendigung der einzelnen Prozesse

Anhang B: ADB TCP/IP Konfigurator

```
1  @echo off
2  setlocal

3  REM How can I connect to Android with ADB over TCP? [closed]
4  REM Source: https://stackoverflow.com/questions/2604727/how-can-i-connect-to-an-
   droid-with-adb-over-tcp/14172202#14172202

5  REM Use a default env variable to find adb if possible
6  if NOT "%AndroidSDK%" == "" set PATH=%PATH%;%AndroidSDK%\platform-tools

7  REM If off is first parameter then we turn off the tcp connection.
8  if "%1%" == "off" goto off

9  REM Set vars
10 set port=%1
11 set int=%2
12 if "%port%" == "" set port=5555
13 if "%int%" == "" set int=wlan0

14 echo ADB TCP/IP Configurator for Android based devices. E.g. for Oculus
   Quest VR headset.
15 echo.
16 echo Now you can connect a device to the USB port of the computer.
17 echo Trying to establish a TCP connection to the USB device.

18 REM Start label of the actions, if the program ends in line 33 it continues here
19 :start

20 REM Get IP Address from device
21 set shellCmd="ip addr show %int% | grep 'inet [0-9]{1,3}(\.[0-9]{1,3}){3}' -oE |
   grep '[0-9]{1,3}(\.[0-9]{1,3}){3}' -oE"
22 for /f %%i in ('adb -d wait-for-device shell %shellCmd%') do set IP=%%i

23 REM Connect ADB to device

24 for /f "tokens=1-2" %%i in ('adb connect %IP%:%port%') do set Response=%%i %%j

25 ::echo "%Response%"

26 if "%Response%" == "cannot connect" (
27  echo Enable TCP on USB device. && adb -d wait-for-device tcpip %port%
28  ) else (
29  if "%Response%" == "already connected" (
30   echo Already connected to device with IP: %IP%, you can unplug it now and
     plug in another one.
31  )
32 )

33 timeout /t 2 /NOBREAK > NUL

34 goto start

35 :fail
36 echo adbWifi [port] [interface]
37 echo adbWifi off
38 goto end

39 :off
40 adb -d wait-for-device usb

41 :end
42 pause
```

Anhang C: Manuelle Testfälle für den Scrcpy-Handler

Manuelle Tests für den Scrcpy-Handler

	Status	Anmerkungen
1. Hauptanwendung		
1.1. Menüführung und -funktionen		
1. Hauptmenü	☐	
2. Menüpunkt 1: manuelle Erstellung einer Konfiguration über USB	☐	
3. Menüpunkt 2: Verwendung der vorherigen Konfiguration	☐	
4. Menüpunkt 3: automatische Konfiguration	☐	
1.2. Funktion - korrekte Größenbestimmung und Ausrichtung der Fenster		
1. 1 Gerät	☐	
2. 2 Geräte	☐	
3. 3 Geräte	☐	
4. 4 Geräte	☐	
5. 5 Geräte	☐	
6. 6 Geräte	☐	
7. 7 Geräte	☐	
8. 8 Geräte	☐	
9. 9 Geräte	☐	
2. Spiegelungstool scrcpy		
2.1. Hauptfunktionalität der Spiegelung		
1. Darstellung der Fenster	☐	
2. Bildübertragung der 1 bis 9 Geräte	☐	
3. Ausreichende Geschwindigkeit (ohne große Verzögerung)	☐	
4. Akzeptable Qualität	☐	
2.2. Bildausschnitt und -übertragung		
1. Oculus Quest 1: 1 vernünftiger Bildausschnit (mittig)	☐	
2. Oculus Quest 2: 1 vernünftiger Bildausschnit (mittig)	☐	
3. Smartphone: das gesamte Bild wird dargestellt	☐	
2.3. Funktionalität der Änderung von scrcpy		
1. Maximieren eines Spiegeluns-Fensters	☐	
2. Minimieren eines Spiegeluns-Fensters	☐	
3. Konfigurationstool		
1. Mindestens 2 Geräte werden konfiguriert		
Tester:		
Datum:		
Status: Alle Tests bestanden/ nicht bestanden	☐	

Anhang D: Fragenkatalog für das Praktikum

Los geht´s...

	stimme gar nicht zu				stimme voll und ganz zu
1. Hatten Sie bereits Erfahrung mit VR-Anwendungen gesammelt?	○	○	○	○	○
2. Hatten Sie Vorkenntnisse zu PV-Anlagen?	○	○	○	○	○
3. Wie groß ist Ihr Interesse an erneuerbaren Energien und PV-Anlagen?	○	○	○	○	○
4. Wie gut funktionierte die Einrichtung der VR-Brillen (mit dem Guardian/ Festlegung des Spielbereiches)?	○	○	○	○	○

5. Welche Fragen oder Probleme gab es während der Anwendung?

6. Wie wurden sie gelöst?

	stimme gar nicht zu				stimme voll und ganz zu
7. Wie gut funktionierte die Bedienung (Laufen, Drehen, Menüführung)?	○	○	○	○	○
8. War die Unterstützung gut, verständlich und schnell?	○	○	○	○	○
9. War die Aufgabenstellung verständlich?	○	○	○	○	○
10. Konnten Sie sich gut auf die Aufgaben konzentrieren?	○	○	○	○	○
11. Konnte die Aufgabe gut umgesetzt werden?	○	○	○	○	○
12. Wie gut funktionierte die Bewegung in der virtuellen Realität?	○	○	○	○	○

13. War die Drehung oder das Laufen zu schnell oder zu langsam?

14. War das Menü verständlich aufgebaut und gut strukturiert oder gibt es dazu Verbesserungsvorschläge?

	stimme gar nicht zu				stimme voll und ganz zu
15. Waren die Funktionalitäten zur PV-Anwendung verständlich?	○	○	○	○	○

Für die Studierenden **1/2**

	stimme gar nicht zu				stimme voll und ganz zu
16. Konnten die PV-Module gut platziert, rotiert und geneigt werden?	○	○	○	○	○
a. Hat das Andocken der Module gut funktioniert?	○	○	○	○	○
17. Wie gut funktionierte die Verschaltung?	○	○	○	○	○
18. Wie gut funktionierte die Zusammenarbeit in der Gruppe?	○	○	○	○	○

19. Gab es Besonderheiten bei der Kommunikation oder Zusammenarbeit in der Gruppe? Welche Probleme, Auffälligkeiten gab es?

	stimme gar nicht zu				stimme voll und ganz zu
20. Was ist Ihnen besonders positiv aufgefallen?	○	○	○	○	○

21. Welche negativen Aspekte sind Ihnen besonders aufgefallen?

22. Wie war der gesamte Eindruck?

23. Gibt es Ideen und Empfehlungen für Verbesserungspotenzial? Wünsche oder Anmerkungen für Erweiterungen oder Änderungen?
    a. Bei der Hardware
    b. Software
    c. Kommunikation
    d. Aufgaben
    e. Unterstützung
    f. Fragen
    g. Andere Punkte

Für die Studierenden **2/2**

Literaturverzeichnis

[1] S. Machnik, „Dynamische Simulation von Energieerträgen und Ermittlung von Kennzahlen einer Photovoltaik-Anlage basierend auf Daten einer virtuellen Umgebung (VR) in Python“. B.S. thesis, University of Applied Sciences Ruhr West, 2021.

[2] T. van der Heusen, „Simulation der Abschattungsverluste bei Photovoltaik-Systemen mit Hilfe der Programmiersprache Python“. B.S. thesis, University of Applied Sciences Ruhr West, 2021.

[3] A. Arntz *et al.,* „Walking on the Bright Side: Evaluating a Photovoltaics Virtual Reality Education Application“, *IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR) 2021*, Nr. 41, 2021.

[4] L. Facebook Technologies, *Features von Oculus Quest | Oculus.* Verfügbar unter: https://www.oculus.com/quest/features/.

[5] D. Bell, *Software engineering for students: a programming approach,* 4. Aufl. Harlow [u.a.]: Addison-Wesley, 2005. [Online]. Verfügbar unter: http://digitale-objekte.hbz-nrw.de/storage2/2018/06/16/file_381/7939228.pdf

[6] *Low-voltage electrical installations – Part 7-712: Requirements for special installations or locations – Photovoltaic (PV) systems*.

[7] Jeremy N. Bailenson, Nick Yee, Jim Blascovich, Andrew C. Beall, Nicole Lundblad und Michael Jin, „The Use of Immersive Virtual Reality in the Learning Sciences: Digital Transformations of Teachers, Students, and Social Context“, *Journal of the Learning Sciences*, Jg. 17, Nr. 1, S. 102–141, 2008, doi: 10.1080/10508400701793141.

[8] A. W. Putra, E. Kamandika, S. Rosyadi, A. Purwadi und Y. Haroen, „Study and design of hybrid off-grid PV-generator power system for administration load and communal load at three regions in Indonesia“ in *2016 3rd Conference on Power Engineering and Renewable Energy (ICPERE)*, 2016, S. 57–62, doi: 10.1109/ICPERE.2016.7904850.

[9] H. Veldhuis und A. Reinders, „Real-time irradiance simulation for PV products and building integrated PV in a virtual dynamic environment“ in *2011 37th IEEE Photovoltaic Specialists Conference*, 2011, S. 193, doi: 10.1109/PVSC.2011.6185879.

[10] P. Abichandani, W. Mcintyre, W. Fligor und D. Lobo, „Solar Energy Education Through a Cloud-Based Desktop Virtual Reality System“, *IEEE Access*, Jg. 7, S. 147081–147093, 2019, doi: 10.1109/ACCESS.2019.2945700.

[11] A. Arntz, D. Kessler und S. C. Eimler, „EnLighten: A Photovoltaics Learning Environment in Virtual Reality“ in *2021 International Conference on Advanced Learning Technologies (ICALT)*, S. 221–223, doi: 10.1109/ICALT52272.2021.00072.

[12] J. Hellriegel und D. Čubela, „Das Potenzial von Virtual Reality für den schulischen Unterricht - Eine konstruktivistische Sicht“, *MedienPädagogik: Zeitschrift für Theorie und Praxis der Medienbildung*, Jg. 2018, Occasional Papers, S. 58–80, 2018. [Online]. Verfügbar unter: https://www.medienpaed.com/article/view/659, DOI=10.21240/mpaed/00/2018.12.11.X

[13] ScreenBeam, „The Difference Between Screen Mirroring, Screen Casting and Screen Sharing“, *ScreenBeam*, 17. Juli 2020, 2020. [Online]. Verfügbar unter: https://www.screenbeam.com/learn-more/wireless-display/the-difference-screen-mirroring-casting-sharing/. Zugriff am: 1. August 2022.

[14] D. Maier, *Artikel: Unity Software – die große Aktienanalyse | Goldesel.de.* [Online]. Verfügbar unter: https://goldesel.de/Artikel/unity-software-die-grosse-aktienanalyse (Zugriff am: 13. April 2022).

[15] 3DMaster, „Unreal Engine vs. Unity: welche Software sollten Gamedeveloper wählen?“, *VisCircle GmbH*, 25. Okt. 2018, 2018. [Online]. Verfügbar unter: https://viscircle.de/unreal-engine-vs-unity-welche-software-sollten-gamedeveloper-waehlen/. Zugriff am: 13. April 2022.

[16] J. Chittesh, *Das Unity-Buch: 2D- und 3D-Spiele entwickeln mit Unity 5,* 1. Aufl. Heidelberg: dpunkt-Verl., 2015.

[17] K. Í. Knutsen, „Visual Scripting in Game Development“. B.S. thesis, Metropolia University of Applied Sciences, 2021. [Online]. Verfügbar unter: https://www.theseus.fi/bitstream/handle/10024/500439/Knutsen_Krist%c3%b3fer.pdf?sequence=2&isAllowed=y

[18] B. Danneberg, „Oculus Link im Test: PC-VR für Oculus Quest (2) mit offiziellem USB-Kabel“, *MIXED*, 4. Sep. 2021, 2021. [Online]. Verfügbar unter: https://mixed.de/oculus-link-test/. Zugriff am: 22. April 2022.

[19] vrnerds, „Meta Quest 2 im Langzeittest – Die beste VR-Brille auf dem Markt“, *VR-Nerds*, 16. Nov. 2020, 2020. [Online]. Verfügbar unter:

https://www.vrnerds.de/oculus-quest-2-im-langzeittest-die-beste-vr-brille-auf-dem-markt/. Zugriff am: 22. April 2022.

[20] S. Segan, „Galaxy S22 Benchmarked: Apple Still Beats Samsung“, *PCMag UK*, 11. Feb. 2022, 2022. [Online]. Verfügbar unter: https://uk.pcmag.com/mobile-phones/138674/galaxy-s22-benchmarked-apple-still-beats-samsung. Zugriff am: 18. August 2022.

[21] Hardware-Helden, „Intel Core i9-12900K schlägt Ryzen 9 5950X im Geekbench“, *Hardware-Helden*, 27. Aug. 2021, 2021. [Online]. Verfügbar unter: https://hardware-helden.de/geekbench-intel-core-i9-12900k-schneller-als-ryzen-9-5950x/. Zugriff am: 18. August 2022.

[22] The cross-platform performance site, *GFXBench - Unified cross-platform 3D graphics benchmark database.* [Online]. Verfügbar unter: https://gfxbench.com/device.jsp?did=106142444 (Zugriff am: 18. August 2022).

[23] The cross-platform performance site, *GFXBench - Unified cross-platform 3D graphics benchmark database.* [Online]. Verfügbar unter: https://gfxbench.com/device.jsp?benchmark=gfx50&did=88649062&os=Windows&api=gl&hwtype=dGPU&hwname=NVIDIA+GeForce+RTX+3080 (Zugriff am: 18. August 2022).

[24] D. Heaney, „Oculus Quest 2 Benchmarked: Just How Powerful Is It?“, *UploadVR*, 23. Okt. 2020, 2020. [Online]. Verfügbar unter: https://uploadvr.com/oculus-quest-2-benchmarks/. Zugriff am: 18. August 2022.

[25] U. Technologies, *Unity - Manual: Light Mode: Baked.* [Online]. Verfügbar unter: https://docs.unity3d.com/Manual/LightMode-Baked.html (Zugriff am: 19. August 2022).

[26] VRChat, *Cross-Platform Setup.* [Online]. Verfügbar unter: https://docs.vrchat.com/docs/cross-platform-setup (Zugriff am: 22. April 2022).

[27] Unity Technologies, *Unity - Manual: Building for Android.* [Online]. Verfügbar unter: https://docs.unity3d.com/Manual/android-BuildProcess.html (Zugriff am: 4. August 2022).

[28] aresz, *How can I see my debug logs on Unity Android?* [Online]. Verfügbar unter: https://stackoverflow.com/questions/29989532/how-can-i-see-my-debug-logs-on-unity-android/29989675#29989675 (Zugriff am: 4. August 2022).

[29] *Photon PUN Pricing Plans | Photon Engine.* [Online]. Verfügbar unter: https://www.photonengine.com/PUN/Pricing (Zugriff am: 1. August 2022).

[30] *Introduction | Photon Engine.* [Online]. Verfügbar unter: https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro (Zugriff am: 10. Juni 2022).

[31] Facebook Technologies LLC, *Oculus.* [Online]. Verfügbar unter: https://www.oculus.com/casting (Zugriff am: 27. Januar 2022).

[32] C. M. Costa, „Cast Oculus Quest 2 to your PC“, *techtipsVR*, 22. Nov. 2020, 2020. [Online]. Verfügbar unter: https://techtipsvr.com/news/cast-oculus-quest-2-to-pc/. Zugriff am: 28. Januar 2022.

[33] Apowersoft, *Kaufen Sie ApowerMirror Online.* [Online]. Verfügbar unter: https://www.apowersoft.de/store/apowermirror.html?appgaid=GA1.2.1642094209.1659626073&appgeo=1486376860&appvisitor=a65a4b841a33a18152a1d7a7bbd9e751&applang=de (Zugriff am: 4. August 2022).

[34] Apowersoft, *ApowerMirror – Smartphone Display in Echtzeit spiegeln/steuern.* [Online]. Verfügbar unter: https://www.apowersoft.de/phone-mirror (Zugriff am: 4. August 2022).

[35] Genymobile, *GitHub - Genymobile/scrcpy: Display and control your Android device.* [Online]. Verfügbar unter: https://github.com/Genymobile/scrcpy (Zugriff am: 27. Januar 2022).

[36] Genymobile, *scrcpy/BUILD.md at master · Genymobile/scrcpy.* [Online]. Verfügbar unter: https://github.com/Genymobile/scrcpy/blob/master/BUILD.md (Zugriff am: 25. Juli 2022).

[37] MSYS2 Team, *MSYS2 - Environments.* [Online]. Verfügbar unter: https://www.msys2.org/docs/environments/ (Zugriff am: 9. August 2022).

[38] TeamViewer Germany GmbH, *TeamViewer App für Google Android.* [Online]. Verfügbar unter: https://www.teamviewer.com/de/download/android/ (Zugriff am: 12. August 2022).

[39] TeamViewer Germany GmbH, *Wählen Sie Ihre Lizenz.* [Online]. Verfügbar unter: https://service.teamviewer.com/de-de/overview/c?billing_switch_mode=M#Single (Zugriff am: 12. August 2022).

[40] Vysor, *Vysor.* [Online]. Verfügbar unter: https://www.vysor.io/ (Zugriff am: 19. August 2022).

[41] S. Chacon und B. Straub, *Pro Git*. Second Edition. Berkeley, CA: Apress: Springer Nature, 2014. [Online]. Verfügbar unter: https://doi.org/10.1007/978-1-4842-0076-6

[42] Python Software Foundation, *Download Python.* [Online]. Verfügbar unter: https://www.python.org/downloads/ (Zugriff am: 16. August 2022).

[43] TIOBE Software BV, *TIOBE Index - TIOBE.* [Online]. Verfügbar unter: https://www.tiobe.com/tiobe-index/ (Zugriff am: 16. August 2022).

[44] M. Inden, *Einfach Python*. dpunkt.verlag, 2021. [Online]. Verfügbar unter: http://www.content-select.com/index.php?id=bib_view&ean=9783969106464

[45] T. Otterbein, „Anforderungen an Leitstände“ in *Eine objektorientierte Architektur für Leitstände zur Feinplanung*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, S. 41–52, doi: 10.1007/978-3-642-51474-6_4.

[46] „Secure system administration“, *National Cyber Security Centre*, 15. Sep. 2020, 2020. [Online]. Verfügbar unter: https://www.ncsc.gov.uk/collection/secure-system-administration/protect-your-administration-interfaces. Zugriff am: 22. August 2022.

[47] *Best Practice für die sichere Administration der IT.* [Online]. Verfügbar unter: https://www.comconsult.com/sichere-administration-it/ (Zugriff am: 22. August 2022).

[48] Felizk, *networking - How can I connect to Android with ADB over TCP? - Stack Overflow.* [Online]. Verfügbar unter: https://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/14172202#14172202 (Zugriff am: 7. August 2022).

[49] Meta, *Meta Quest release notes.* [Online]. Verfügbar unter: https://store.facebook.com/en-us/help/quest/articles/whats-new/release-notes/ (Zugriff am: 19. August 2022).

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht. Die vorgelegte Arbeit hat weder in der gegenwärtigen noch in einer anderen Fassung schon einem anderen Fachbereich der Hochschule Ruhr West oder einer anderen wissenschaftlichen Hochschule vorgelegen.

Bottrop, 22.08.2022 Sebastian Wienzek

Ort, Datum Unterschrift