Machine-Vision als Grundlage zur Entwicklung innovativer Konferenzsysteme

Ein Forschungsprojekt unter dem Aspekt der technischen Informatik

Besondere Lernleistung

unter fachlicher Betreuung durch
Frau Haase
Herrn Meisch-Peschew

von
Benjamin Wagner
Berlin, 29.01.2024

Treptow-Kolleg Berlin
Institut zur Erlangung der Hochschulreife

Danksagung

Besonderer Dank geht an meine Frau, die mir mit Geduld und Verständnis zur Seite stand und meine Kinder, die Teile der Software, so beispielsweise ein Pong-Spiel[1], das durch maschinelles Sehen gesteuert wird, mit viel Freude getestet haben.
Mein Dank gilt auch den Fachbereichen Physik und Informatik sowie dem Förderverein des Treptow-Kollegs, die meiner Arbeit, besonders im Hinblick auf die Entwicklung des Präsentationsstischs selbst, viel Aufmerksamkeit und Unterstützung geschenkt haben.
Zuletzt danke ich den Teilnehmenden der Informatik AG, die mit großem Interesse meine Begeisterung für die Informatik teilen und mich ebenso unterstützten.

[1] Demonstration siehe Anhang, Datenträger: `./video/demo-pong.mp4`

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Listing-Verzeichnis

Abkürzungsverzeichnis

1 Einleitung

Diese Arbeit gliedert sich in vier Bestandteile, wobei der erste Teil die Einleitung darstellt. Der zweite Abschnitt der Arbeit bildet den theoretischen Teil. Darin werden die Begriffe definiert und erläutert, Forschungsgrundlagen und der Stand der Technik beschrieben. Basierend auf den Ergebnissen aus dem zweiten Teil, wird im dritten Abschnitt der Prototyp eines Präsentationstischs sowie die Software zur adaptiven Informationsdarstellung entworfen und implementiert. Schließlich werden im vierten Kapitel die Ergebnisse bewertet und zusammengefasst.

1.1 Motivation

Digitale Medien gewinnen im Alltag, im Beruf und auch an Bildungs- und Kultureinrichtungen immer mehr an Bedeutung. Insbesondere die Informationsgewinnung profitiert deutlich von der technischen Entwicklung digitaler Systeme wie Netzwerkinfrastrukturen, digitale Bibliotheken oder Multimedia.
Konferenzen, also sämtliche anlassbezogenen mediengestützten Treffen mehrerer Personen im öffentlichen Raum, stellen einen wichtigen Bestandteil in Unternehmen und Bildungseinrichtungen dar. Sie dienen der Wissensvermittlung, dem Austausch und der Darstellung von Informationen mittels Vorträgen, Präsentationen, Experimenten und Vorführungen. Im Zuge technischer Entwicklungen, insbesondere die der Personalcomputer, verlagerte sich die immer wichtiger werdende ergänzende Mediennutzung von beispielsweise Overheadprojektoren (1931 von Carl Zeiss entwickelt [9]) auf die Verwendung digitaler Präsentationsformen. Jedoch blieb das grundlegende Konzept, nach dem sich eine Präsentation in Folien gliedert, gleich. Die Informationsaufbereitung und -darstellung im Rahmen der Unternehmenskommunikation und der Wissensvermittlung an Bildungseinrichtungen erfolgt trotz digitaler Transformation nahezu vollständig über klassische nicht-adaptive Konferenzkonzepte, die technisch bedingt nicht das volle Potential digitaler Infrastrukturen ausnutzen. Anders verhält sich das in der Industrie, aber auch im Dienstleistungssektor, wo bereits seit über 40 Jahren zunehmend digitale Systeme zur Erfassung und Darstellung von Informationen zum Einsatz kommen. Beispielhaft zu nennen sind automatisierte Sortieranlagen in Briefzentren, Kameraüberwachung mit automatischer Personenerfassung an Bahnhöfen oder Pfandautomaten in Supermärkten.
Bis Anfang 2023 stellte das Deutsche Technikmuseum Berlin eines der interessantesten Machine-Vision-System-basierten Entwicklungen der Musikbranche, den sogenannten „Reactable“ aus (siehe Abb. 1 und [12]). Das von Martin Kaltenbrunner und anderen an der Universität Barcelona entwickelte System erhebt den Anspruch, ein Musikinstrument zu sein, das so intuitiv zu bedienen sein solle, dass es keiner Anleitung bedarf (vgl. [7]). Dieses System stellt sich als Kandidat für die Grundlage der Forschung nach einer Lösung für ein modernes, intuitives Konferenzsystem heraus, das das bis heute dominierende klassische lineare Folienkonzept ablösen und genauso einfach wie der „Reactable“ zu bedienen sein soll.

Abbildung 1: Reactable Live. Reactable Systems [4]

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Konzipierung und Entwicklung eines ganzheitlichen Konferenzsystems, dem ein Machine-Vision-System (MVS)[2] zugrunde liegen soll, um konferenzbezogene Inhalte adaptiv und kollaborativ darzustellen. Das beinhaltet die Abstraktion

- der Markerobjekte/-signale zu allgemeinen Positionsinformationen (Existenz, Nähe, 2D, Richtung)
- der Anbindung an Erweiterungen zur anwendungsabhängigen Interpretation der Positionsinformationen

Das MVS soll demnach an verschiedene Anwendungsfälle anpassbar sein, was ein modulares Konzept zur Funktionserweiterung notwendig macht. Für diese Arbeit wird eine Anwendung im Bildungssektor untersucht, bei der Lehrende und Lernende mithilfe des MVS-basierten Konferenzsystems Unterrichtsinhalte darstellen und Informationen austauschen. Zwei der drei Subsysteme - zum einen das Hardwaresystem einschließlich Sensoren und Aktoren des MVS, zum anderen die Steuerungssoftware - stehen im Fokus der Arbeit. Die künftige Anbindung an ein Datenbanksystem über ein Webinterface, das das dritte Subsystem darstellt, ist nicht Untersuchungsgegenstand dieser Arbeit.

1.3 Forschung auf ähnlichem Gebiet

Maschinelles Sehen und künstliche Intelligenz in der Agrar- und Ernährungswirtschaft zur Unterstützung allgemeiner Aufgaben: Maschinelles Sehen und künstliche Intelligenz (KI) spielen eine immer bedeutendere Rolle in der Agrar- und Ernährungswirtschaft, um allgemeine Aufgaben zu unterstützen. Durch den Einsatz von Bilderkennungstechnologien können Maschinen visuelle Daten wie Pflanzenwachstum, Krankheiten oder Unkraut identifizieren, was Landwirten dabei hilft, präzisere Entscheidungen bei der Bewirtschaftung ihrer Felder zu treffen. Künstliche Intelligenz ermöglicht zudem die Analyse großer Mengen von agrarwissenschaftlichen Daten, um Ertragsprognosen zu verbessern,

[2] nähere Erläuterungen im Abschnitt 2

optimierte Anbaustrategien zu entwickeln und Ressourcen effizienter einzusetzen. Insgesamt tragen maschinelles Sehen und künstliche Intelligenz dazu bei, die Effizienz, Produktivität und Nachhaltigkeit in der Landwirtschaft zu steigern. [2]

Computergestützte Erkennung von Gesichtern und Objekten zur benutzerfreundlichen Interaktion: Die computergestützte Erkennung von Gesichtern und Objekten dient dazu, benutzerfreundliche Interaktionen zwischen Menschen und Maschinen zu ermöglichen. Diese Technologien verwenden Bildverarbeitungsalgorithmen und KI, um Gesichter und Objekte in Echtzeit zu identifizieren und zu verfolgen. Im Bereich der benutzerfreundlichen Interaktionen können sie beispielsweise in Gesichtserkennungssystemen für Authentifizierungszwecke, Zugangskontrollen oder personalisierte Nutzererlebnisse eingesetzt werden. Darüber hinaus ermöglicht die Erkennung von Objekten die Interaktion mit der Umgebung, sei es in der Augmented Reality, Automatisierung von Prozessen oder bei der Verbesserung von Mensch-Maschine-Schnittstellen. Diese Technologien tragen dazu bei, die Interaktion zwischen Mensch und Computer intuitiver, effizienter und sicherer zu gestalten. [13]

1.4 Anforderungen der Untersuchungsaspekte

Für die zuvor beschriebenen Ziele dieser Arbeit sollen die Integration von Sensordaten und die Anpassungsfähigkeit an verschiedene Anwendungsfälle eines MVS-basierten Konferenzsystems untersucht und entwickelt werden. Daraus ergeben sich funktionale und qualitative Anforderungen, an denen sich die Entwicklung des Systems orientiert.

1. Funktionale Anforderungen:
   Hierzu zählen die Verknüpfung verschiedener Sensoren- und Aktorensysteme, um eine umfassende und effiziente Datenerfassung und -verarbeitung zu ermöglichen. Ebenso wird die Implementierung eines multimediafähigen Grafik- und Benutzer-Interfaces angestrebt, um die ansprechende und benutzerfreundliche Interaktion sicherzustellen.

2. Qualitative Anforderungen:
   Das zu entwickelnde System wird daraufhin geprüft, ob es auf andere Anwendungsfälle übertragbar ist. Darüber hinaus werden Maßnahmen ergriffen, um die Wartbarkeit und Erweiterbarkeit des Systems sicherzustellen, damit es flexibel und zukunftsorientiert bleibt.

Nachdem die Schlüsselanforderungen an das MVS-basierte Konferenzsystem festgelegt wurden, wird im folgenden Abschnitt detailliert auf die Grundlagen eingegangen, die als Basis für die Umsetzung der identifizierten Anforderungen dienen. Dabei werden technische Aspekte, methodische Ansätze und weitere fundamentale Prinzipien eingehend erläutert, um einen umfassenden Einblick in die Struktur und Funktionsweise des geplanten Konferenzsystems zu gewähren. Diese Grundlagen bilden den Auftakt für den Entwicklungsprozess, der den funktionalen und qualitativen Anforderungen gerecht wird.

2 Machine-Vision-Systeme

2.1 Definition

Machine-Vision ist die Fähigkeit von Computersystemen, mithilfe von Sensoren selbstständig Informationen über die Umgebung zu erhalten. Diese Informationen werden elektronisch verarbeitet und beispielsweise über Aktoren als Antwort an die Umgebung zurückgegeben.

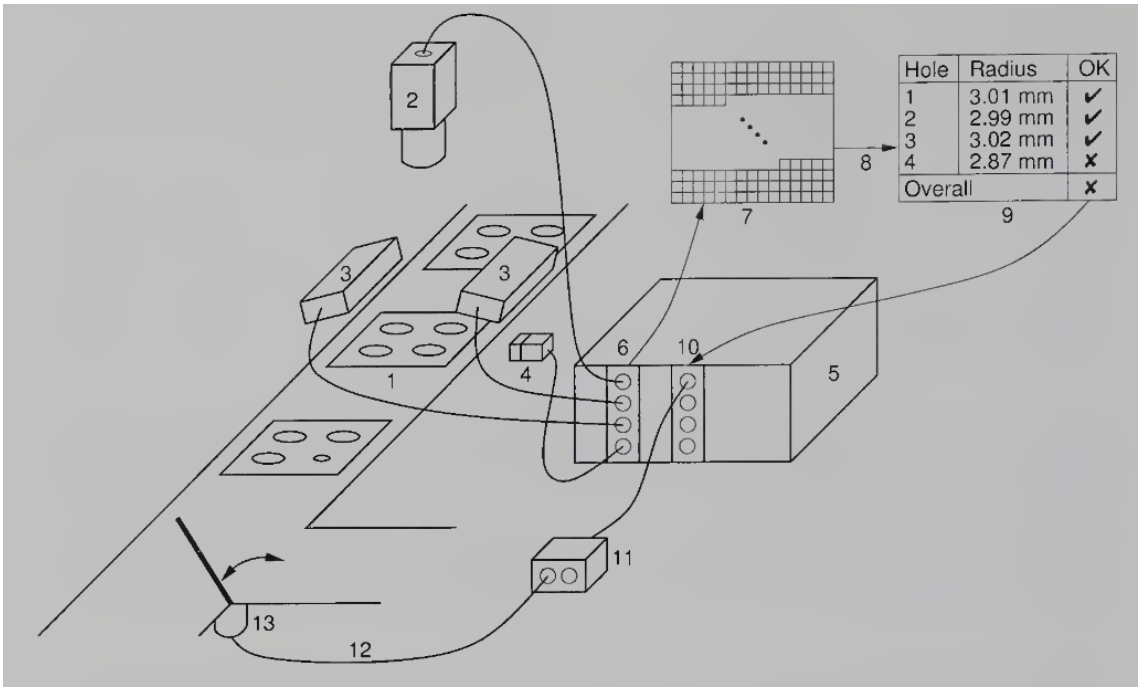


Abbildung 2: Schematische Darstellung eines Machine-Vision-Systems [11]

Das in Abbildung 2 dargestellte MVS arbeitet nach folgendem Muster: Das betrachtete Objekt (1) wird mit der Kamera (2) erfasst. Die fotoelektrischen Sensoren (3) lösen die Kamera aus, sobald sich ein Objekt in Position befindet. Über das Kamera-Computer-Interface (6) wird das Bild zum Computer übertragen. Eine Machine-Vision-Software (8) untersucht die Bilddaten (7) und gibt die Objektauswertung (9) an einen Aktor, zum Beispiel an einen programmierbaren Logikbaustein (11), zurück. Dieser schaltet abhängig vom Ergebnis die Weiche (13) so um, dass das Objekt entweder passieren kann oder aussortiert wird. Entscheidend für die Qualität der Bilddaten ist neben der Kamera auch die Ausleuchtung des zu betrachtenden Objekts. Diese wurde in der schematischen Darstellung vernachlässigt.

2.2 Das Machine-Vision-System „Reactable“

Der eingangs vorgestellte „Reactable“ arbeitet als MVS nach dem gleichen Prinzip: Eine Kamera erfasst sogenannte Touch-Gesten, Markerobjekte und geometrische Objekte auf einem semitransparenten Projektionstisch. Die MVS-Software verarbeitet die erfassten Bilddaten, wandelt diese in Signale zur Steuerung von Digital Audio Workstation (DAW)-Software[3] um und gibt ein optisches Feedback über einen Beamer an die Projektionsfläche zurück. Abbildung 3 zeigt den schematischen Aufbau:

[3]Software zum Arrangieren und Komponieren von Musik.

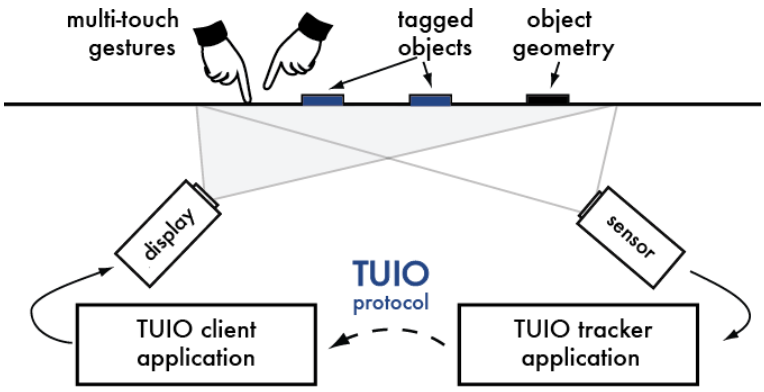


Abbildung 3: Schematische Darstellung: Reactable [5]. Objekte und Gesten werden von einer Kamera (sensor) erfasst und an die Tracking-Software (TUIO tracker application) weitergeleitet, dort ausgewertet und in Steuersignale umgewandelt. Die Anwendersoftware (TUIO client appilcation) liest diese Steuersignale aus und arbeitet entsprechend der Steuerinformationen bestimmte Prozesse ab. Einer dieser Prozesse ist die Ausgabe von Grafiken über einen Beamer (display) an die Projektionsfläche.

2.3 Übertragung auf diese Forschungsarbeit

Im Gegensatz zum allgemeinen MVS aus Abbildung 2 dient die Tischfläche hier zwei Funktionen, nämlich der Erfassung von Objekten und der Projektion optischer Inhalte, was eine bidirektionale Lichtstrecke zur Folge hat. Daher können sich die Lichtwege gegenseitig so beeinflussen, dass es zu Fehlern bei der Erfassung kommt. Die Kamera kann beispielsweise nicht unterscheiden, ob ein Markerobjekt tatsächlich auf dem Projektionstisch liegt oder ob es nur vom Beamer projiziert wurde. Die Lösung liegt in der Beschränkung der Empfindlichkeit der Kamera für nicht sichtbares Licht. Dazu ist es erforderlich, einen Blick auf die relevanten Wellenlängenbereiche zu werfen:

Tabelle 1: Wellenlängenbereiche im elektromagnetischen Spektrum

Bereich	**Bezeichnung**	**Wellenlänge**	**Verwendung**
Sichtbar	Tageslicht	$380nm$ - $780nm$	Display
Infrarot	IR-A	$780nm$ - $1,4\mu m$	Sensor (ca. $850nm$)
	IR-B	$1,4\mu m$ - $3\mu m$	
	IR-C	$3\mu m$ - $50\mu m$	

Die an das sichtbare Licht angrenzenden Bereiche sind der Ultraviolett- und der Infrarotbereich. Ultraviolettes Licht ist aufgrund seiner kurzen Wellenlänge sehr energiereich. Denn die Energie (E) einer elektromagnetischen Welle ist umgekehrt proportional zu seiner Wellenlänge (λ). Es gilt: $E \propto \frac{1}{\lambda}$. Energiereiche Strahlung führt langfristig zu Gesundheitsschäden, ähnlich wie zu häufiges Sonnenbaden Hautkrebs auslösen kann. Daher muss die Kamera zur Erfassung der Objekte empfindlich für Infrarotlicht sein. Trotz der besseren Verträglichkeit mit biologischen Organismen sollte die Lichtleistung der Infrarotbeleuchtung nicht unnötig hoch sein. Denn obwohl eine hohe Lichtleistung die Qualität der durch die Kamera erfassten Bilddaten verbessert, indem

1. geringe Lichtempfindlichkeiten (ISO-Wert) das Bildrauschen reduzieren,
2. stark geschlossene Blenden die Schärfeebene verbreitern und
3. kurze Belichtungszeiten schnellere Bewegungen von Markern erlauben,

wird jedoch gleichzeitig die Strahlungsbelastung auf den menschlichen Körper erhöht, was unter Umständen zu Schwindel und Kopfschmerzen führen kann. Verschiedene DIN-Normen (z. B. DIN EN 12198, DIN EN 14255-2 und BGI 5006) empfehlen daher insbesondere bei intensiven Lichtquellen, direktem Blick in die Beleuchtung und über Stunden hinweg sogenannte Einhausungen zu verwenden und industrielle IR-Beleuchtungen zeitlich zu takten, um unnötige Lichtemissionen und auch Sicherheitsdiskussionen zu vermeiden. Bei kleineren industriellen LED-Beleuchtungen sei dies üblicherweise nicht zutreffend, dennoch solle die effektiv erforderliche Lichtleistung nicht unnötig überschritten werden.
Zur Berechnung der wahrscheinlich benötigten Lichtleistung der Infrarotbeleuchtung müssen einerseits die technischen Daten der IR-LEDs und die der Kamera, insbesondere die ISO-Lichtemfindlichkeit, bekannt sein. Die tatsächlich benötigte Lichtleistung hängt allerdings auch von den Gegebenheiten der Umgebung, wie verwendete Stoffe, Raumgröße und ähnliches, ab, deren Einfluss auf die effektiv benötigte Lichtleistung nur durch Versuchsaufbauten ermittelt werden kann. Jedoch hilft die Berechnung dabei, einen groben Erwartungswert zu erhalten, der als Ausgangspunkt für die Ermittlung des Idealwerts am Versuchsaufbau dient.

Die ISO-Lichtempfindlichkeit ist eine technische, willkürlich festgelegte Größe zur Berechnung der Gesamtbelichtung einer Aufnahme. Der Abstand von einem ISO-Wert zum nächstgrößeren entspricht der Lichtdurchlassmenge einer Blende. Die Blendenzahl ergibt sich aus der Formel für die Kreisfläche $A = d^2 \cdot \frac{\pi}{4}$. Soll nun die Lichtmenge, die durch das Objektiv auf den Kamerasensor trifft, halbiert werden, muss sich die Fläche der Blende vergrößern. Wegen der Kreisform muss der Faktor von $\sqrt{2}$, nicht aber 2, verwendet werden, was ungefähr $1,4$ entspricht. Die Skala der ISO-Empfindlichkeit ist - wie auch die Skala der Belichtungszeit - genau nach diesem Faktor eingeteilt, um bei Änderungen der Werte eine einheitliche Gesamtbelichtung beibehalten zu können.

Tabelle 2: ISO-Schritte und Blendenzahl bei Erhalt der Gesamtbelichtung

ISO	Blendenzahl in f/p (gerundet)	Belichtungszeit in $\frac{1}{q}s$
100	$1,0$	30
200	$1,4$	30
400	$2,0$	30
800	$2,8$	30
1600	$4,0$	30
3200	$5,6$	30

Tabelle 2 zeigt, wie für jede ISO-Wert-Erhöhung die Blendenzahl um jeweils einen Schritt

erhöht werden muss (die Blende wird mit Erhöhung der Blendenzahl weiter geschlossen), damit die Gesamtbelichtung[4] gleich bleibt. Die Belichtungszeit wird hier nicht geändert, um den Zusammenhang zwischen ISO-Lichtempfindlichkeit und Blendenzahl zu verdeutlichen.

2.3.1 Berechnung der benötigten Lichtgesamtleistung

Zur Berechnung der Gesamtleistung werden die Formeln für die Berechnung des Raumwinkels eines IR-Leuchtmittels, dessen Lichtintensität und Beleuchtungsstärke sowie die Formel für die eigentliche Berechnung der Gesamtleistung in Abhängigkeit der Lichtempfindlichkeit des Kamerasensors benötigt.

Daten der IR-LEDs: Zur Ausleuchtung der Projektionsfläche mit Infrarotlicht werden IR-LEDs mit diffuser und flächendeckender Ausleuchtung eingesetzt, die eine Blendung der Kamera verhindern sollen. Die wichtigsten Kenndaten für die weitere Berechnung sind: Öffnungswinkel $\theta = 60°$, Nennleistung $P = 4W$, Abstand zur Projektionsfläche $r \approx 0,4m$ und Projektionsfläche $A \approx 0,6m^2$

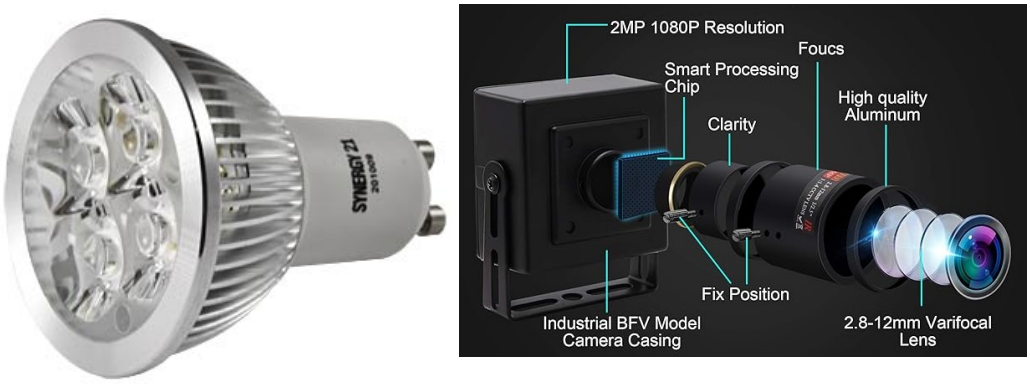


Abbildung 4: Links: SYN 86948 Infrarotstrahler, Sockel GU10. Rechts: ELP 1080P USB-Kamera mit Objektivmount

Daten der Kamera: Zur Erfassung der Bilddaten wird eine ELP 1080P USB-Kamera mit Objektivmount für Wechselobjekte und einem 2,8-12mm Vario-Objektiv verwendet, das manuellen Zoom und Fokus ermöglicht. Mit bis zu $100\frac{\text{Bilder}}{\text{Sekunde}}$ und RAW-Data werden die Bildsignale flüssig und nahezu verzögerungsfrei übertragen, was die Reaktionsfähigkeit der zu entwickelnden Software deutlich verbessert. Die Kamera verfügt außerdem über einen großen Sensorempfindlichkeitsbereich. Für die weitere Berechnung wird ein Wert von $l = \text{ISO400}$ für typische Helligkeiten in Innenräumen und eine Blende von $f/5,6$ für eine ausreichend große Schärfeebene gewählt.

[4] Die Gesamtbelichtung kann als Produkt aus ISO-Lichtempfindlichkeit, Blendenzahl und Belichtungszeit betrachtet werden.

Berechnung des Raumwinkels Ω

$$\Omega = 2\pi \left(1 - \cos\left(\frac{\theta}{2}\right)\right) \tag{1}$$

$$\Omega = 2\pi \left(1 - \cos\left(\frac{60°}{2}\right)\right) \tag{2}$$

$$\Omega \approx 1,884\,\mathrm{sr} \tag{3}$$

Berechnung der Lichtintensität I

$$I = \frac{P}{\Omega} \tag{4}$$

$$I = \frac{4W}{1,884\,\mathrm{sr}} \tag{5}$$

Berechnung der Beleuchtungsstärke E

$$E = \frac{I}{r^2} \tag{6}$$

$$E = \frac{4W}{1,884\mathrm{sr} \cdot (0,4m)^2} \tag{7}$$

Berechnung der wahrscheinlich benötigten Lichtgesamtleistung P_{ges}

$$P_{\mathrm{ges}} = E \cdot A \tag{8}$$

$$P_{\mathrm{ges}} = \frac{4W \cdot 0,6m^2}{1,884\mathrm{sr} \cdot 0,16m^2} \tag{9}$$

$$P_{\mathrm{ges}} \approx 7,96W \tag{10}$$

Die Einheit **sr** (Steradiant) ist definiert als $1\mathrm{sr} = 1\frac{m^2}{m^2} = 1$ (vgl. Gleichungen 3 und 9), daher bleibt als Einheit W für die Leistung übrig. Das Ergebnis (vgl. Gleichung 10) zeigt, dass eine ausreichende Beleuchtung der Projektionsfläche mit zwei IR-LEDs zu je $4W$ umsetzbar ist. Unter Berücksichtigung der angestrebten Blende von $f/5,6$ (also 5 Blendenzahlschritte von $f/1$) und ISO-Lichtempfindlichkeit ISO400 (entspricht 3 Blendenzahlschritten von $f/2,8$ auf $f/1,0$) erhält man:

$$P_{\mathrm{ges}} = P \cdot \text{Blendenzahlerhöhung} \div \text{Blendenzahlveringerung} \tag{11}$$

$$P_{\mathrm{ges}} = P \cdot \left(\frac{5}{1}\right)^2 \cdot \left(\frac{1}{3}\right)^2 \tag{12}$$

$$P_{\mathrm{ges}} \approx 7,96W \cdot \left(\frac{5}{3}\right)^2 \tag{13}$$

$$P_{\mathrm{ges}} \approx 22,11W \tag{14}$$

Diese Lichtgesamtleistung lässt sich mit fünf bis sechs IR-LEDs zu je $4W$ erreichen. Der tatsächliche Endwert kann jedoch durch zusätzliches Erhöhen der ISO-Lichtempfindlichkeit weiter gesenkt werden, um die Strahlungsbelastung auf den menschlichen Körper so niedrig wie möglich zu halten. Für ISO800 wird beispielsweise nur eine Leistung von rund $12,44W$ benötigt, daher wird nun auch diese Einstellung angestrebt.

2.3.2 Bildakquise

Das vorgestellte MVS „Reactable“ verwendet die ReacTIVision Engine [6], eine eigens entwickelte MVS-Engine, das das Tangible User Interface Object (TUIO)-Protokoll als Datenprotokoll nutzt. TUIO ist unter der GNU Lesser General Public License (LGPL)[5] quelloffen lizenziert und wird von den Entwicklern aktiv für die Verwendung in eigenen Projekten beworben. Diese Engine kann demnach uneingeschränkt für die Bildakquise im Rahmen dieser Forschungsarbeit verwendet werden.
Die Tracking-Software ReacTIVision ist zuständig für die Erfassung und Interpretation der Bilddaten, die mittels Open Sound Control (OSC)-Protokoll an ein Netzwerk übertragen werden. Erfasst werden dabei Objektformen (Blob), Touch-Eingaben (Cursor) und sogenannte Amöbenmarker (Object).

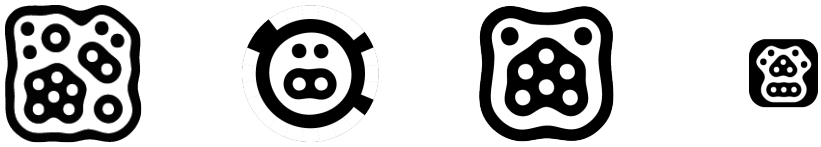
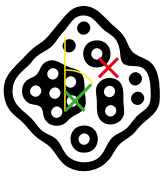


Abbildung 5: Symbole von links nach rechts: Classic, Yamaarashi, Mini und Small

Die Markerarten unterscheiden sich hinsichtlich Performance, Anzahl möglicher unterschiedlicher IDs und Druckgröße. Obwohl Yamarashi-Marker die beste Erfassungsperformance, geringe Mindestmaße und mehr als 10^6 ID-Kombinationen aufweisen, wird nach aktuellem Forschungsstand vom Entwicklerteam von der Nutzung im Produktionsbetrieb abgeraten, da sich das Projekt derzeit noch in der Entwicklungsphase befindet: „You can also additionally enable the new (beta) Yamaarashi fiducal symbols for testing.“ (Martin Kaltenbrunner. Dokumentation, ReacTIVision. 2022)
Da sich die Classic-Marker bereits in Produktionsumgebungen bewährt haben (vgl. Reactable, Abb. 1), werden für diese Arbeit ebenfalls die Classic-Marker verwendet, da vor allem gegen die Verwendung der Mini- und Small-Marker dessen geringe Anzahl unterschiedlicher IDs spricht.

Aufbau und Funktionsweise der Markersymbole: Jedes Markersymbol ist so aufgebaut, dass neben seiner Marker-ID auch die relative Position zum Kamerabild und der Rotationswinkel erfasst werden kann. Wie sich die ID ableiten lässt, zeigt Abb. 6.
Zur Berechnung des Rotationswinkels werden zwei Punkte in Relation gesetzt: ① das durchschnittliche Zentrum aller weißen Punkte und ② das durchschnittliche Zentrum aller schwarzen Punkte. Anhand der Bilddaten ist der Software bekannt, wo im akquirierten Bild Norden ist. Von dieser Abweichung lässt sich der Rotationswinkel ableiten.



Die in diesem Abschnitt erläuterten Grundlagen und gewonnenen Erkenntnisse werden im Folgenden zunächst zum Aufbau und Design des Projektionstischs genutzt. Anschließend

[5] Die LGPL erlaubt den Entwicklern und Firmen das Verwenden und Einbinden von LGPL-Software in eigene (sogar proprietäre) Software, ohne durch ein starkes Copyleft gezwungen zu sein, den Quellcode der eigenen Software-Teile offenzulegen.[3]

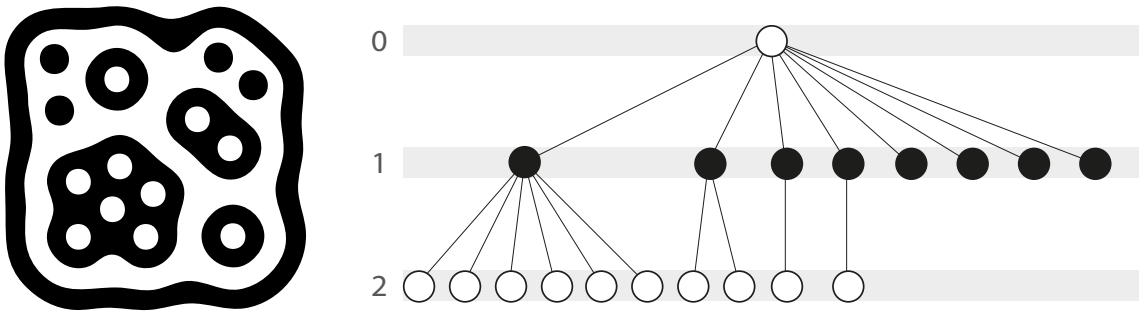


Abbildung 6: Amöbenmarker in Knotendarstellung. Anhand der Knoten und deren Helligkeit lassen sich die Marker in Ebenen aufteilen. Die numerische Darstellung zeigt, wie die Software jedem Marker eine eindeutige ID zuordnen kann. Der abgebildete Marker hat den Fingerabdruck 0122222212212121111 und ist der Symbol-ID 0 zugeordnet.

wird der Fokus auf den Softwareentwicklungsprozess gelegt, wobei speziell die Anforderungen der zuvor untersuchten Aspekte berücksichtigt werden. Diese aufeinander aufbauenden Schritte ermöglichen eine umfassende Betrachtung und Umsetzung des Konferenzsystems, angepasst an die spezifischen Anforderungen und Zielsetzungen.

3 Design und Softwareentwicklung

Sowohl Hardware als auch Software des Konferenzsystems sollen modular aufgebaut sein, um größtmögliche Flexibilität in Bezug auf Anwendungsfall und Einsatzort gewährleisten zu können. Um das zu erreichen, sind neben einer offenen Struktur der Software vor allem auch besondere Anforderungen an den Aufbau des Projektionstischs zu stellen. Dazu zählen einerseits funktionale Anforderungen, wie die Anbindung verschiedener Sensoren- und Aktorensysteme, und qualitative Anforderungen im Hinblick auf die Übertragbarkeit auf andere Anwendungsfälle sowie die Wartbarkeit und Erweiterbarkeit im Allgemeinen.

3.1 Grundlegende Designentscheidungen

Für den Aufbau des Präsentationstischs bedeutet dies, dass insbesondere der Trägerrahmen nicht-destruktiv gestaltet sein muss. Angelehnt an CNC-Fräsen und 3D-Drucker wird daher ein V-Slot-System, also ein Aluminiumprofil-Trägersystem verwendet, für das zahlreiche Sensoren- und Aktorenhalterungen sowie Baugruppen zur Verfügung stehen. Außerdem können eigens entwickelte Baugruppen einfach und schnell integriert werden.
Für die Integration externer Sensoren und Aktoren muss der Präsentationstisch über eine ausreichende Zahl von außen erreichbarer USB-Ports bereitstellen, da die Hauptkommunikation vorrangig über serielle Schnittstellen erfolgen wird. Nachrangig sind auch Steuerungsmöglichkeiten über Netzwerkschnittstellen denkbar.

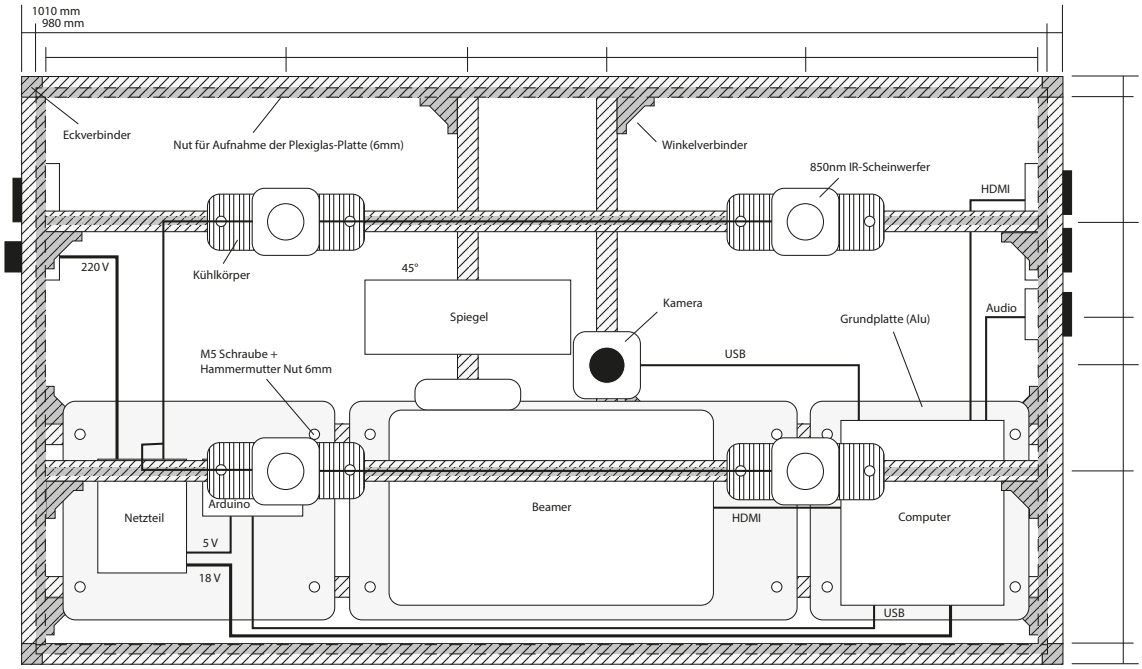


Abbildung 7: Konstruktionsplan des Präsentationstischs mit Verteilung der einzelnen Komponenten.

3.2 Sensorik

3.2.1 Kamera und Projektionsfläche

Die verwendete Kamera (Siehe Abschnitt 2.3.1, Seite 7) verwendet einen CMOS-Sensor, der sich im Vergleich zu CCD-Sensoren vor allem dadurch auszeichnet, dass dieser be-

sonders empfindlich im nahen Infrarotwellenlängenbereich ist. Das und die ausreichende Beleuchtung durch Infrarot-LEDs allein ist allerdings noch nicht hinreichend für die optimale Erfassung von auf der Projektionsfläche vorhandenen Markern und Touch-Gesten. Die Gestaltung der Projektionsfläche selbst ist maßgeblich entscheidend für die Leistungsfähigkeit, sowohl für die Erfassung von Objekten als auch für Projektion grafischer Inhalte.

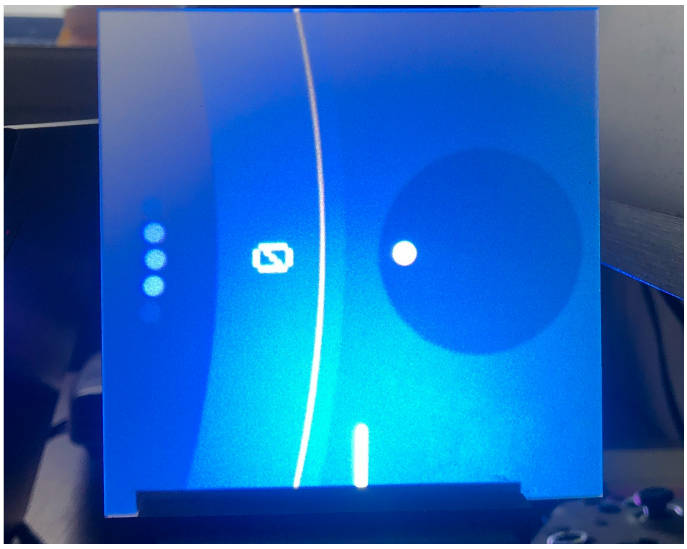


Abbildung 8: Mustertest mit diffusem Acrylglas

Aufgrund der Vorteile von Acrylglas gegenüber herkömmlichem Glas (Siehe Tabelle 3), gerade im Hinblick auf Dichte, Bruchfestigkeit und Formbarkeit, besonders im Rahmen der intensiven Nutzung in öffentlichen Einrichtungen, fiel die erste Wahl auf diffuses Acrylglas, das in einem ersten Materialtest (ca. 100cm^2) sowohl die Erfassung von Markern als auch die Projektion von Grafik ermöglichte (Siehe Abb. 8).

Tabelle 3: Vergleich der Stoffeigenschaften von Acrylglas und herkömmlichem Glas

Eigenschaft	**Acrylglas**	**Herkömmliches Glas**
Optische Klarheit	Hoch	Hoch
Dichte (g/cm^3)	Niedrig	Hoch
Bruchfestigkeit	Hoch	Gering
Wärmeleitfähigkeit (W/m-K)	Niedrig	Hoch
UV-Beständigkeit	Ja	Nein
Formbarkeit	Sehr gut	Begrenzt
Kratzfestigkeit	Geringer	Höher
Chemische Beständigkeit	Gut	Sehr gut

Mit Fertigstellung des ersten Prototyps des Projektionstischs zeigt sich, dass die Erfassung der Marker zufriedenstellend funktioniert. Die Qualität der Projektion ist jedoch unbrauchbar, was sich darin zeigt, dass bei großen Blickwinkeln auf die Projektionsfläche die projizierten Inhalte nur noch schwer erkennbar sind. Der Streuungskoeffizient scheint

bei diffusem Acrylglas viel zu gering zu sein. Durch testweises Auflegen eines Bogens Kopierpapier (A4) konnte der Streuungskoeffizient deutlich erhöht werden. Jedoch verringert sich dadurch gleichzeitig die Durchlässigkeit. Ein weiterer Nachteil ist, dass durch die 6mm dicke diffuse Acrylglasschicht sowohl die Schärfe der Projektion als auch die der Markererfassung relativ gering ist.
Um einerseits den Streuungskoeffizienten und andererseits die Bildschärfe bei gleichbleibender Dicke der Acrylglasschicht zu erhöhen, wird daher die bisher verwendete diffuse Acrylglasplatte gegen einen Verbund aus einer transparenten Acrylglasplatte und einem aufgelegten Transparentpapier ausgetauscht. Wie in Abbildung 9 (rechts) zu sehen ist, verbessert sich dadurch die Projektionsqualität erheblich.

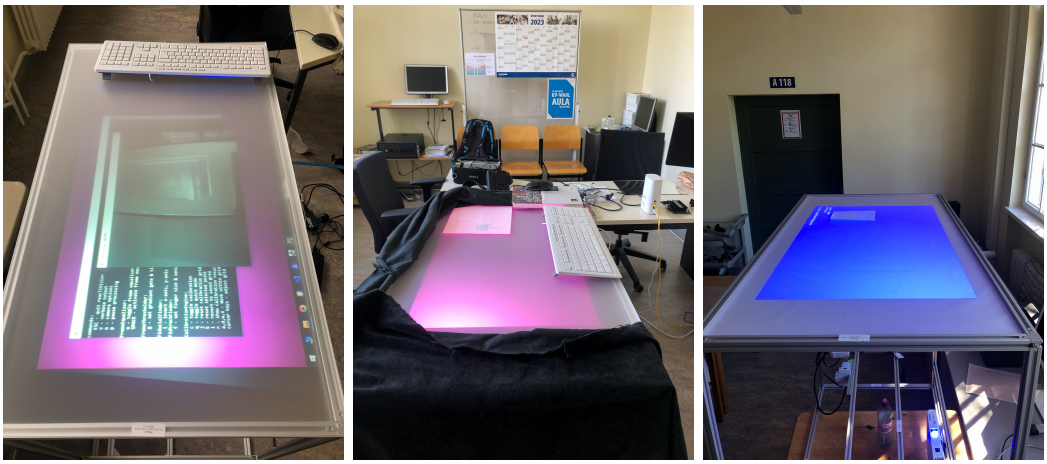


Abbildung 9: **Links:** Diffuse Acrylglasplatte. **Mitte:** Diffuse Acrylglasplatte mit aufgelegtem Kopierpapier. **Rechts:** Transparente Acrylglasplatte mit aufgelegtem Transparentpapier.

Auch die Qualität der Erfassung der Markerobjekte verbessert sich zusehends, wie in Abbildung 10 festzustellen ist, da nun die diffuse Schicht durch die Verwendung einer Transparentpapierauflage nicht mehr $6mm$, sondern nur noch $0,05mm$ - $0,1mm$ (Je nach Papierhersteller) beträgt.

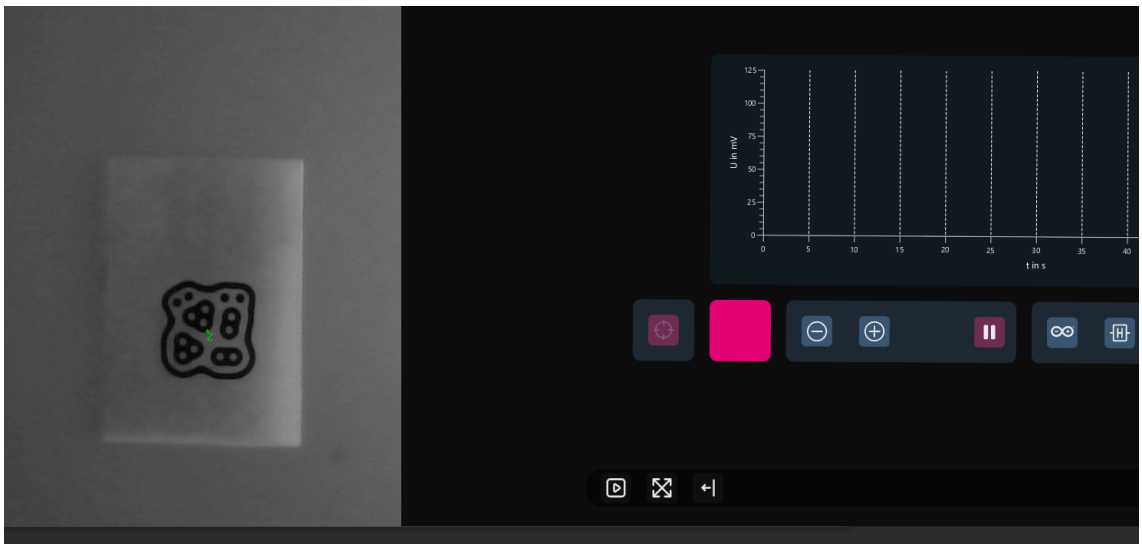


Abbildung 10: Erfassung von Markerobjekten und Moduldarstellung

3.2.2 Weitere Sensoren

Neben der Erfassung von Präsentationsobjekten durch die verwendete Kamera gibt es noch weitere Sensoren, die Informationen an die Software des Konferenzsystems übermitteln können. Solche Sensoren können entweder intern, zum Beispiel durch Anbringung an das modulare V-Slot-System, oder extern, jeweils über eine serielle Schnittstelle mit dem Konferenzsystem und seiner Software interagieren.
Beispielhaft wird ein programmierbarer Mikroprozessorbaustein „Arduino“ [1] über den universellen seriellen Bus (USB) zur Übermittlung von Messdaten aus Elektronikschaltungen integriert, um die modulare Erweiterbarkeit und Anpassbarkeit an verschiedene Anwendungsfälle, als einen Teilaspekt der Untersuchungsanforderungen, zu demonstrieren.

3.3 Entwicklung der Software

Für die Erfassung von Objekten auf dem Präsentationstisch kann die bereits verfügbare Software „ReacTIVision“ unverändert verwendet werden. Daher befasst sich diese Arbeit ausschließlich mit der Konzeptionierung und Entwicklung der Anwendersoftware zur Interpretation der von der Tracking-Software übermittelten Signaldaten. Wie auch die Hardware des Konferenzsystems wird im Folgenden untersucht, wie die Software gestaltet werden kann, um zum einen Multimediainhalte darstellen und ausgeben zu können und zum anderen, wie die Software ohne großen Aufwand an diverse Anwendungsfälle angepasst und erweitert werden kann.
Grundlage der gesamten Signalverarbeitung ist die `TUIO`-Bibliothek [8] in der Version 1.1, die unter anderem für `C++`, aber auch für `Java` zur Verfügung steht. Da die Programmiersprache Java nicht nur auf sämtliche Hardware-Ressourcen wie Kameras, Monitore oder serielle Schnittstellen zugreifen kann, sondern auch sehr leicht zu erlernen ist, wird im weiteren Verlauf die Umsetzung des Konferenzsystems mit Java untersucht.

3.3.1 Wahl des geeignetsten JDK

Das markergesteuerte Konferenzsystem macht extensiven Gebrauch von beweglichen Grafik- und Multimediainhalten. Daher muss als erstes geklärt werden, welches JDK geeignete Klassenbibliotheken enthält, die diesen Anforderungen gerecht werden. Das GUI-Toolkit Swing kommt aufgrund seinem auf die Programmierung klassischer grafischer Benutzerschnittstellen ausgelegten Schwerpunkt nicht infrage, da hier insbesondere die Multimediafähigkeit fehlt. Die Klassenbibliothek `Graphics2D` eignet sich zumindest eingeschränkt für die grafische Umsetzung. Mit Java 11 wurde 2014 der Swing-Nachfolger JavaFX eingeführt, der viele Verbesserungen und Vorteile hinsichtlich der Entwicklung moderner Multimediaanwendungen gegenüber `Graphics2D` aufweist. Tabelle 4 zeigt die wichtigsten Gemeinsamkeiten und Unterschiede zwischen den Packages `java.awt` bzw. `java.swing` und `javafx`.
Softwareentwickler Manuel Mauky von Saxonia Systems sieht die Vorteile ebenfalls bei JavaFX, indem er sagt, „dass mit JavaFX viele Dinge leichter und mit weniger Code umsetzbar sind. Neben modernen Features wie deklarativer UI-Beschreibung mittels FXML,

[6] Liberica JDK 11.0.20+8 ist die einzige JDK/JRE, die noch JavaFX enthält. In späteren Versionen wurde die Bibliothek entfernt.

Tabelle 4: Vergleich GUI-Toolkits.

Funktionalität	**Swing und Graphics2D**	**Java 11[6] mit JavaFX**
Collision Detection	✓	✓
Grafikobjekte zeichnen	✓	✓
Transformieren	✓	✓
Multimedia	eingeschränkt (wav)	✓ (wav, mp3, mp4 etc)
MediaPlayer	✗	✓
Gruppierung	✗ (Workaround über JPanel)	✓
Timeline / Keyframes	✗	✓
Transitions	✗	✓

Styling per CSS und einfachen Animationen ist [sein] persönliches Highlight das Databinding, welches u.a. auch Reactive-Programming[7] ermöglicht“ [10].

3.3.2 MarkerListener implementieren

Wie im Interface `TuioListener` (Listing 9, Seite XI) der TUIO-Bibliothek beschrieben, müssen alle Methoden zum Hinzufügen, Modifizieren und Entfernen von Objekten durch Marker, Formen und Gesten implementiert und in der `TuioClient`-Instanz registriert werden. Grundsätzlich können drei verschiedene Arten von Objekte verarbeitet werden:

1. Marker (`TuioObject`)
   `Objekte identifiziert nach Symbol`
2. Touch-Gesten (`TuioCursor`)
   `Fingerberührungen`
3. Blobs (`TuioBlob`)
   `Objekte identifiziert nach Form`
   Blobs werden nicht weiter für die Entwicklung des Konferenzsystems in Betracht gezogen und werden daher im weiteren Verlauf vernachlässigt.

Die Verwendung von JavaFX führt dazu, dass die Implementierung der `TUIO`-Bibliothek nicht so wie ursprünglich vom Entwickler Martin Kaltenbrunner vorgesehen erfolgen kann, da ihr Code auf die Verwendung von `Graphics2D` bzw. `Graphcis3D` ausgelegt ist. Während die `TUIO`-Bibliothek den herkömmlichen Java-Thread[8] verwendet, nutzt JavaFX einen eigenen Thread für die Manipulation der JavaFX-Objekte. Der Zugriff auf FX-Objekte aus nicht-FX-Threads führt zur Fehlermeldung „*Not on FX application thread;*“. Die Trennung

[7] Reactive-Programming beschreibt ein Designparadigma, das auf asynchroner Programmierlogik basiert, um Echtzeitaktualisierungen ansonsten statischer Inhalte durchzuführen.

[8] Threads sind essenziell, um parallel und effizient auf Ressourcen zugreifen und Operationen ausführen zu können. Sie erlauben es, unterschiedliche Teile einer Anwendung oder eines Prozesses zeitgleich auszuführen, ohne dass die einzelnen Teile sich gegenseitig blockieren.

beizubehalten ist jedoch insofern sinnvoll, als sich die Abfrage- und Ausgabeprozesse nicht gegenseitig blockieren, wodurch Leistungseinbußen in Hinblick auf Echtzeitreaktionen vermieden werden.
Auch um das `Tuio`-Package nicht modifizieren zu müssen - gerade im Hinblick auf künftige Updates - darf also `MarkerListener` weder FX-Grafikobjekte zur Bühne hinzufügen oder entfernen, noch Position und weitere Attribute verändern. Für jeden optisch erfassten Marker soll demnach ein `FX`-Objekt, das von `javafx.scene.Node` erbt, in einer dem Controller zugänglichen `ArrayList` gespeichert werden. Obwohl das Hinzufügen zur `ArrayList` reibungslos funktioniert, fehlt die logische Verknüpfung zum `TuioObject`-Objekt. Elemente der `ArrayList` können nicht in Abhängigkeit vom `TuioObject` modifiziert oder entfernt werden. Eine einfache Lösung stellt die Verwendung assoziativer Arrays dar, denn diese nutzen statt eines Index vom Typ `int` Objekte eines beliebigen Datentyps wie beispielsweise die `TuioObject`-Instanzen selbst. In Java sind assoziative Arrays mithilfe sogenannter `HashMaps` umsetzbar:

Listing 1: Markerobjekt instantiieren

```
22 // org.engine.MarkerListener.java
23 @Override
24 public void addTuioObject(TuioObject tobj) {
25   Controller.objectList.put( tobj, new TangibleObject(tobj) );
26 }
```

Der Auszug in Listing 1 zeigt die Methode zum Hinzufügen eines `TangibleObject` zur `HashMap` „objectList“ der abstrakten Oberklasse `Controller` (siehe auch vollständige Klasse Listing 8, Seite X). Als Index wird das `TuioObject` selbst verwendet, da so die Zuordnung zwischen realem Markerobjekt und dem grafischen Pendant effektiv umgesetzt werden kann. Die Vorteile von `HashMaps` gegenüber `ArrayLists` zeigen sich im Besonderen beim Bewegen und Entfernen von Markerobjekten. Während die `HashMap` durchlaufen wird, wird für jeden Schlüssel, repräsentiert durch ein `TuioObject`, der zugehörige Wert, repräsentiert durch ein `TangibleObject`, modifiziert. Das `TuioObject` entspricht dem realen Marker; das `TangibleObject` stellt das grafische Pendant dar, dem ein Präsentationsmodul zugewiesen werden kann. Die Methode `removeTuioObject(TuioObject␣tobj)` verdeutlicht, wie ein `TangbibleObject` anhand des Schlüssels aus der `HashMap` entfernt wird:

Listing 2: Markerobjekt entferen

```
35 // org.engine.MarkerListener.java
36 @Override
37 public void removeTuioObject(TuioObject tobj) {
38   Controller.objectList.remove(tobj);
39 }
```

Der `MarkerListener` kümmert sich also, abgesehen von initialen und finalen Modulmethoden, lediglich um die Verwaltung der:

1. Marker (`TuioObject`) und
2. Touch-Gesten (`TuioCursor`).

Wie die `TuioObject-TangibleObject`-Verknüpfung verhält sich auch die `TuioCursor-FingerTouchObject`-Verknüpfung für Touch-Gesten. Diese werden jedoch im Klassenattribut `Controller.cursorList` gespeichert.

3.3.3 Controller

Der `Controller` stellt das Zentrum der Anwendung dar und verknüpft alle beteiligten Komponenten logisch miteinander. Er steuert die Daten, die an die `View`-Schicht übergeben werden, indem modulspezifische Methoden der `TangibleObject`-Instanzen ausgeführt werden. Der `Controller` stellt außerdem die UDP-Port-Verbindung über den `TuioClient` her und ruft Präsentationsdaten von der `Model`-Schicht ab. Im Verlauf dieses Abschnitts wird die Struktur der Software dargestellt und die Interaktion mit der Klasse `MarkerListener` erläutert.

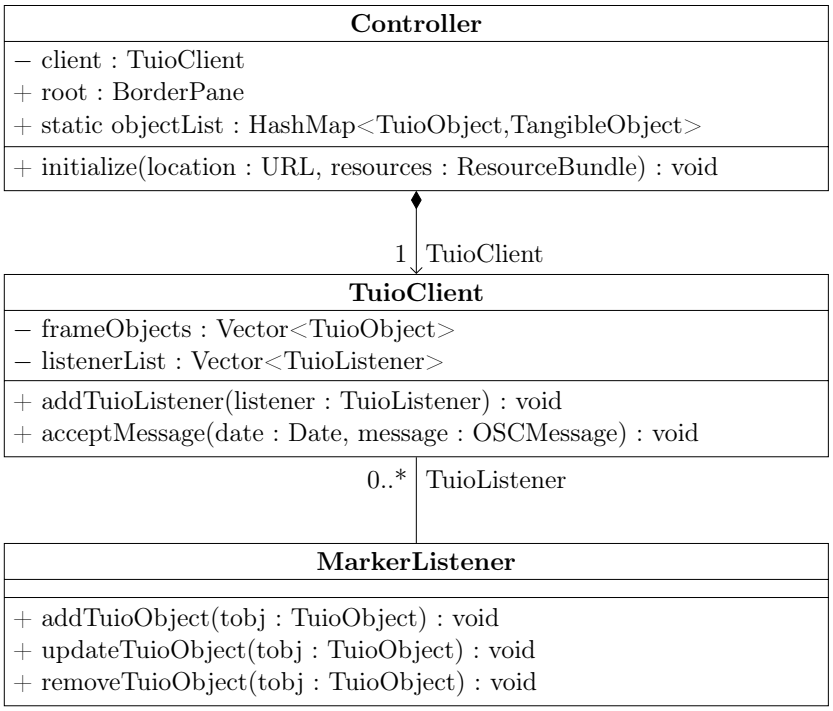


Abbildung 11: **UML-Diagramm** (Auszug): Abfrage der Sensordaten und Ausgabe exemplarisch für Makerobjekte. ① Der `Controller` sammelt `TuioObject`-Daten und aktualisiert entsprechend dieser Daten die Elemente von `root`. ② `TuioClient` stellt eine Verbindung zum konfigurierten UDP-Port her und hört diesen auf `OSCMessage`s ab. Abhängig von der Nachricht wird eine der Methoden der zugewiesenen ③ `TuioListener` ausgeführt, um `TuioObject`e zur `Controller.objectlist` hinzuzufügen, von dieser zu entfernen oder die Objekte zu modifizieren.

Die Listener-Methode `updateTuioObject()` wird mit jeder Veränderung der Marker auf der Projektionsfläche aufgerufen. Dennoch sprechen verschiedene Gründe gegen die Nutzung und Implementierung dieser Methode:

- MarkerLister befindet sich außerhalb des JavaFX-Threads. Daher kein Eingriff in FX-Elemente erlaubt
- Steuerung durch die Modelschicht widerspricht MVC-Design-Pattern

Start der Anwendung Nachdem der Controller vollständig initialisiert und der Timer gestartet wurde, wird die Methode `tick(float secondsSinceLastFrame)` (Listing 3, Z. 66) solange ausgeführt, bis der `AnimationTimer` gestoppt wird. Das bedeutet, dass jetzt die Objekte der `HashMaps` `objectList` und `cursorList` solange regelmäßig abgearbeitet werden, wie die Anwendung nicht beendet wurde. Währenddessen fügt `MarkerListener` erfasste Objekte zu den entsprechenden `HashMaps` hinzu oder entfernt diese.

Listing 3: AnimationTimer-Loop

```
64 // org.gamereact.controller.AppController.java
65   @Override
66   public void tick(float secondsSinceLastFrame) {
67     menuBar.setTranslateX(root.getWidth() / 2);
68     menuBar.setTranslateY(root.getHeight() - 50);
69
70     getKeyboardInput();
71     getTangibleInput(getAnimationDuration());
72   }
```

① Abfrage von Keyboard-Eingaben (Listing 3, Z. 70) und ② Interaktion zwischen den Elementen von `objectList` und `cursorList` (Listing 3, Z. 71).

Listing 4: Verarbeitung der Marker und Touch-Gesten (Auszug)

```
120   // org.gamereact.controller.AppController.java
121   for (Map.Entry<TuioObject, TangibleObject> object :
        objectList.entrySet()) {
122     setObjectPosition(object, animationDuration);
123   }
124   this.objectGroup.getChildren().retainAll(objectList.values());
125
126
127   for (Map.Entry<TuioCursor, FingerTouchObject> cursor :
        cursorList.entrySet()) {
128     setCursorPosition(cursor);
129     getMenuBarInput(cursor.getValue());
130   }
131   this.cursorGroup.getChildren().retainAll(cursorList.values());
```

① Für jedes Element der `HashMap` wird die Position entsprechend der Markerkoordinaten aktualisiert (Listing 4, Z. 121-123). ② Nur Elemente, die in `objectList` vorhanden sind, sollen weiter auf der Bühne verbleiben. Andere Elemente werden entfernt (Listing 4, Z. 124). ③ Das gleiche Verfahren wird auf Elemente der `curserList` angewendet (Listing 4, Z. 127-131).

3.3.4 Marker-Modulbibliothek

Gemäß den Anforderungen an die Untersuchungsaspekte soll die Frage geklärt werden, wie die Präsentationssoftware einerseits an verschiedene Anwendungsfälle anpassbar, andererseits aber auch innerhalb eines Anwendungsfalls erweiterbar gestaltet werden kann. Die Gliederung nach Modulen mit bestimmten Fähigkeiten und Aufgaben stellt einen möglichen Ansatz zur Umsetzung dieser Anforderung dar. Kundenspezifische Anforderungen an die Software können daher in Form von Modulen im Java-Package `org.gamereact.module` angelegt werden. Module müssen dabei von `org.gamereact.module.Module` oder einer ihrer spezialisierten abstrakten Unterklassen erben und alle mit der Oberklasse verknüpften

Interface-Methoden von `org.engine.ModuleInterface` implementieren. Die vier momentan verfügbaren spezialisierten und abstrakten Unterklassen sind:

1. `EmptyModule`
   Leeres Modul (wird auch immer dann erzeugt, wenn einer Symbol-ID kein anderes Modul zugeordnet wurde)

2. `ControlModule`
   Modul, das andere Module steuern kann (z. B. Lautstärke von Audio)

3. `ControllableModule`
   Modul, das gesteuert werden kann (z. B. Mediaplayer)

4. `ElectronicComponentModule`
   Modul, das digitale Elektronikschaltungen ermöglicht (z. B. Kondensator, Spule)

Darüber hinaus können beliebig viele weitere Module für diverse andere Anwendungen nach eigenen Bedürfnissen implementiert werden. Ein solches spezialisiertes Modul wurde beispielhaft implementiert, um Messungen vom Mikrocontroller „Arduino“ [1] darstellen und auswerten zu können (vgl. Anhang, Datenträger: `./video/arduino-kondensator.mp4`).

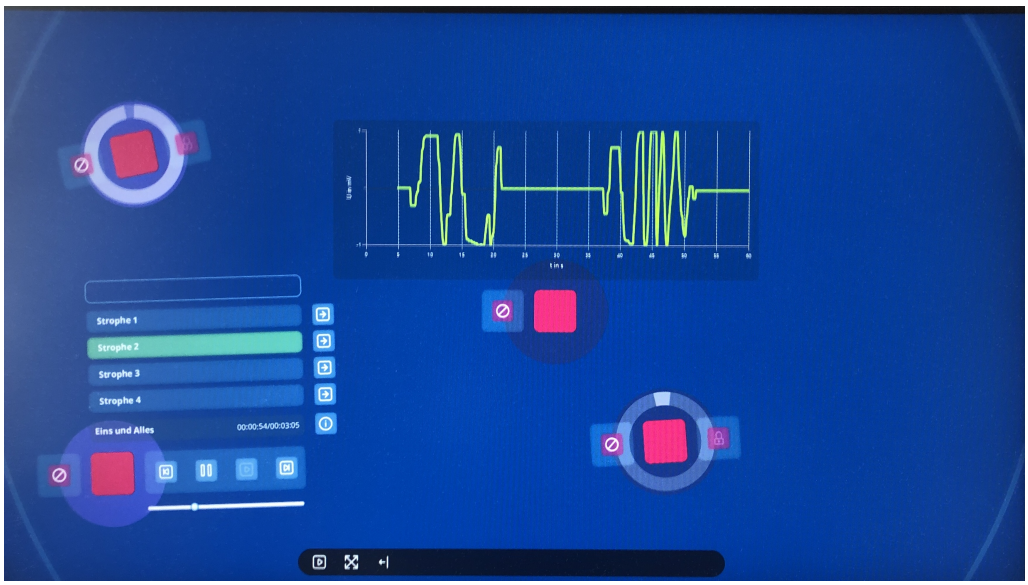


Abbildung 12: Diagramm-Modul zur Auswertung von Messdaten wie Spannungen in elektrischen Schaltungen und weitere Module

Listing 5: Arduino-Programmcode zur Übertragung von seriellen Daten

```
1  // SerialDemo.ino
2  double current = 0;
3  String message = "current";
4
5  void setup() {
6    Serial.begin(9600);
7  }
8
9  void loop() {
```

```
10      if (Serial.available()) {
11        message = "current" + String(current) + "E";
12        Serial.print(message);
13      }
14    }
```

Das Listing zeigt beispielhaft, wie mehrere Messdaten über die serielle Schnittstelle an die Präsentationssoftware übermittelt werden können. Hier soll die Messung der Stromstärke (eng.: current) übermittelt werden. Ein Datensatz besteht aus drei Teilen, die zu einer Zeichenkette zusammengefügt werden: ① Der Name der physikalischen Größe, ② Messwert und ③ der Großbuchstabe **E** als Steuerzeichen für das Ende eines Datensatzes.

Listing 6: Auswertung serieller Datensätze

```
43  // org.engine.ArduinoControl.java
44  for (byte newDatum : newData)
45  {
46    Character serialInput = (char) newDatum;
47    if(Character.isLetterOrDigit(serialInput))
48    {
49      text.append(serialInput);
50    }
51    // E kennzeichnet Ende eines Datensatzes
52    if (serialInput.equals('E'))
53    {
54      parseData(text.toString());
55      text.setLength(0);
56      break;
57    }
58  }
```

Da die Daten fortlaufend und zeichenweise übermittelt werden, ist das Steuerzeichen zwingend notwendig. Andernfalls kann nicht zuverlässig festgestellt werden, wann ein Datensatz endet. Sobald ein Datensatz komplett ist, wird mit der Methode `parseData()` die Dezimalzahl extrahiert und an das Diagramm-Modul übermittelt.

Implementierte Module müssen im zweiten Schritt in der sogenannten Modulbibliothek registriert werden, um verwendet werden zu können. Diese Bibliothek wird durch die Klasse `org.gamereact.ModuleLibrary` abgebildet (siehe Listing 10, Seite XIII).

3.3.5 Marker-Konfiguration

Die Konfiguration der Marker, also die Zuordnung von Modulen und Moduleigenschaften zu bestimmten Symbol-IDs, muss extern erfolgen, da eine Änderung der Konfiguration immer auch die Neukompilierung der Software zur Folge hätte. Die Konfiguration soll außerdem für Informatik-unerfahrene Personen leicht zu verstehen und nachvollziehbar sein und, unter anderem auch für die spätere Anbindung an ein datenbankbasiertes Webinterface, eine maschinenlesbare Syntax aufweisen.
Ein Format, das all diesen Anforderungen am besten entspricht, ist das XML[9]-Format. Präsentationsdaten wie Texte, Bilder, Musik und Videos können in Unterordnern abgelegt und

[9] XML steht für „eXtensible Markup Language“ und heißt so viel wie „erweiterbare Auszeichnungs-

in der Konfigurationsdatei zugeordnet werden. Da XML benutzerdefinierte Tags erlaubt, ist es möglich, eine eigene Konfigurationssprache zu entwickeln, die aktuell folgendermaßen aussieht:

Listing 7: XML Markerkonfiguration

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<marker>
  <object id="0" class="AUDIO_PLAYER_MODULE"
    file="goethe-eins_und_alles.mp3" title="Eins und Alles">
    <part name="Strophe 1" start="12160" end="36592" />
    <part name="Strophe 2" start="37708" end="63814" />
    <part name="Strophe 3" start="64372" end="89139" />
    <part name="Strophe 4" start="89197" end="117961" />
  </object>
  <object id="3" class="IMAGE_MODULE"  file="" title="Image">
    <image name="1" file="1.jpg" />
  </object>
</marker>
```

Die Konfiguration wird mit Instantiierung eines `TangibleObject`s abgefragt:

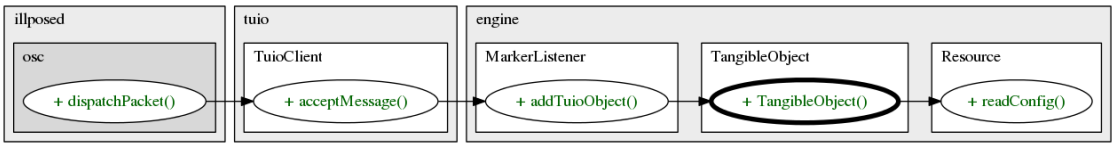


Abbildung 13: Call-Struktur vom Lesen der OSC-Nachricht bis zum Auslesen der Konfiguration (`marker.xml`).

Die beispielhafte Konfiguration aus Listing 7 enthält zwei Markerdefinitionen (`<object id='0'>..</object>` und `<object id='3'>..</object>`), die folgende Module erzeugen:

Marker-ID: 0
`AudioPlayerModule`
`goethe-eins_und_alles.mp3` mit 4 Abschnitten (Strophen).

Marker-ID: 3
`ImageModule`
Eine `.jpg`-Datei: `1.jpg`.

Jedes `<object>`-Tag enthält zudem ein `class`-Attribut (Schlüssel), das festlegt, welche Art Modul geladen und dem `TangibleObject` zugeordnet werden soll. Das `TangibleObject` stellt eine Brücke zwischen `TuioObject` (Marker) und `Module` (konkreter Inhalt) dar. Die zur Konfiguration der Marker notwendigen Klassen sind:

Resource Lädt `marker.xml`-Konfiguration. Überprüft, ob Marker konfiguriert wurde und instantiiert ggf. die `ModuleLibrary`.

ModuleLibrary Enthält eine Map mit Schlüsseln (siehe Listing 7) und Methoden zur Instantiierung der `Module`-Objekte.

sprache“. Es handelt sich dabei um ein textbasiertes Datenformat, welches in einem einfachen Texteditor bearbeitet werden kann.

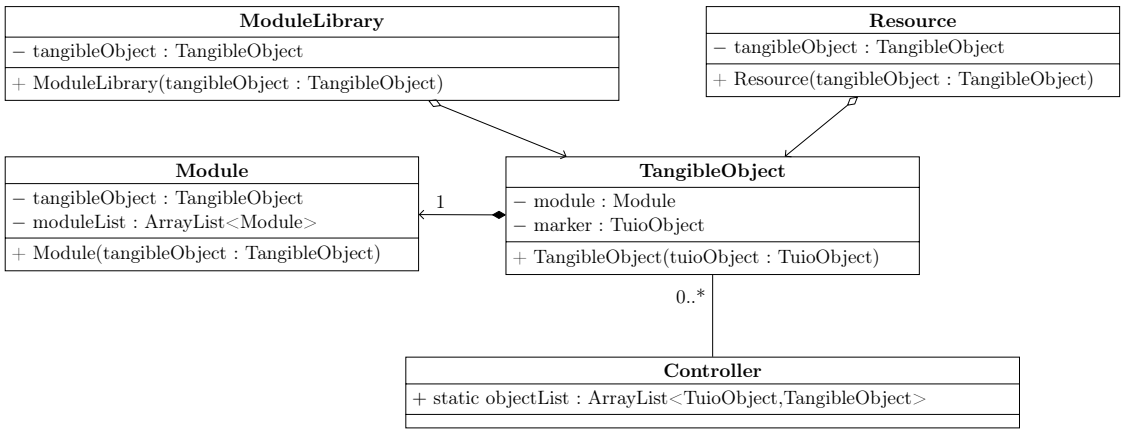


Abbildung 14: UML-Diagramm, Instantiierung von Modulen. `MarkerListener` ruft mit Auflegen eines neuen Markers auf der Projektionsfläche die Methode `addTuioObject(TuioObject tobj)`, welche eine neue Instanz von `TangibleObject` zur HashMap `Controller.objectList` hinzufügt. Welches Modul geladen wird, hängt von der Markerkonfiguration und der Klasse `ModuleLibrary` ab.

Wie sich das Konferenzsystem in Interaktion mit aufgelegten Markern, deren Rotation und Nähe zueinander sowie Touch-Gesten verhält, wird im Anhang auf dem beiliegenden Datenträger gezeigt (`./video/demo.mp4`). Hier wird deutlich, dass `Control`- und `Controllable`-Marker eine logische Beziehung eingehen, sobald ihr Abstand zueinander eine kritische Distanz unterschreitet. `Control`-Marker geben dabei in Abhängigkeit ihres Rotationswinkel ein Steuersignal an verknüpfte `Controllable`-Marker ab, die dann entsprechend reagieren. Dies zeigt, dass die in der Zielsetzung definierten Ziele der Abstraktion

- der Markerobjekte/-signale zu allgemeinen Positionsinformationen (Existenz, Nähe, 2D, Richtung) und
- der Anbindung an Erweiterungen zur anwendungsabhängigen Interpretation der Positionsinformationen

durch das erarbeitete Hardware- und Softwarekonzept unter Berücksichtigung funktionaler und qualitativer Anforderungen erreicht werden. Inwiefern maschinelles Sehen folglich als Grundlage innovativer Konferenzsysteme in Betracht gezogen werden kann, wird im folgenden Abschnitt unter Beachtung der bis hierher gewonnenen Erkenntnisse zusammenfassend diskutiert.

4 Diskussion und Fazit

Insgesamt zeigt sich, dass maschinelles Sehen als Grundlage für innovative Konferenzsysteme seine Berechtigung hat und insbesondere in Zeiten des digitalen Wandels - dazu zählt auch der Wandel weg von klassischen Eingabegeräten wie Tastatur und Maus hin zur intuitiven KI-gestützten Steuerung von Computersystemen durch Mimik und Gestik - nicht vernachlässigt werden sollte. Die Entwicklung und Implementierung ist zwar anfänglich zeit- und kostenintensiv, das Erlernen der Handhabung solcher Systeme gelingt jedoch spielend und der durch seine Nutzung entstehende Mehrwert übersteigt die dazu vergleichsweise geringen Investitionen. Ganzheitlich betrachtet meint die im Titel enthaltene „Innovation“ vor allem das intuitive Arbeiten, wie es dieses Konferenzsystem ermöglicht, indem Präsentationsinhalte nahezu jeder Art, einerseits durch haptische Bewegungsabläufe und andererseits durch Symbole, erkannt und eingesetzt werden können. Bisher etablierte, zumeist lineare Systeme, können dies nicht oder nur teilweise leisten.

4.1 Herausforderungen

Die Entwicklung der Projektionsfläche, aber auch die der Software für das Präsentationssystem war von einigen Herausforderungen geprägt, was sich zum Beispiel in anfänglichen Schwierigkeiten bei der JavaFX-basierten Implementierung der TUIO-Bibliothek zeigte. Die Entscheidung für Java und JavaFX ermöglicht zwar erweiterte Multimediafähigkeiten, doch die Implementierung des MarkerListeners konnte wegen der Inkompatibilität erst nach Entkopplung der Threads erfolgreich umgesetzt werden.

4.2 Einschränkungen

Zum Zeitpunkt der Einreichung dieser Arbeit ist der Präsentationstisch noch nicht vollständig fertiggestellt. Daher bleiben die Ergebnisse der Überprüfung von Funktionalität und Qualität in einer Realumgebung offen.

4.3 Ausblick

Für die Zukunft könnte die lokale Lösung der Markerkonfiguration durch ein datenbankbasiertes Webinterface ersetzt werden, um eine benutzerfreundliche und zentralisierte Verwaltung zu ermöglichen. Dies würde die Konfigurationsprozesse weiter optimieren und langfristig zu einer effizienteren Verwaltung des Präsentationssystems beitragen. Das Konferenzsystem würde so um das bisher noch fehlende dritte Subsystem komplettiert und die Nutzung, besonders im Sinne der umfassenden Digitalisierung, einfacher und intuitiver.

Es wird angestrebt, diese Arbeit als Grundlage für weitere Forschungen im Rahmen des sich voraussichtlich anschließenden Computer Engineering-Studiums fortzusetzen und das Konferenzsystem im Hinblick auf den Produktionseinsatz vollständig fertigzustellen.

4.4 Übertragung auf andere mögliche Anwendungsfälle

Der entwickelte Präsentationstisch ermöglicht die Erfassung und Interpretation von Objekten in Echtzeit und die modulare Softwarestruktur bietet Flexibilität für zukünftige

Anpassungen und Erweiterungen. Die gesamte Entwicklung legt somit die Grundlage für innovative Konferenzsysteme, die eine Vielzahl von Anwendungen bietet.
Im Bildungswesen spielt das Konferenzsystem eine enorm wichtige Rolle bei der digitalen Transformation. Vorträge, Präsentationen und wissenschaftliche Experimente gehören an Schulen und Universitäten zum Alltag. Dieses System hilft den Lernenden und Lehrkräften dabei, diese auf intuitive und innovative Weise umzusetzen.
Im Theater, bei Konzerten und bei vielen anderen Veranstaltungen auf einer Bühne wird nahezu immer auf Licht- und Tontechnik zurückgegriffen. Die dafür notwendigen komplizierten Steuer- und Mischpulte[10] können durch dieses Konferenzsystem ergänzt oder sogar ersetzt werden, wodurch die Handhabung zum Kinderspiel für jeden wird. Denn das System ist so intuitiv, dass es praktisch keiner Anleitung bedarf.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.
Alle sinngemäß und wörtlich übernommenen Textstellen aus fremden Quellen wurden kenntlich gemacht.

Ort, Datum Name und Unterschrift

[10] Lichtanlagen werden über DMX-Protokolle gesteuert, die über ein Modul auch durch das Konferenzsystem abgedeckt werden können.

Anhang

Datenträger

Der diesem Anhang beiliegende Datenträger enthält:

`./kompiliert/` Artefakt der Java-Anwendung (Start über `_GameReactCore.jar`)

`./quellcode/` Java-Projekt mit gesamtem Quellcode

`./software/` ReacTIVision-Engine und TUIO-Simulator

`./video/` Aufnahmen zur Demonstration der Funktionalität

Quellcodes

Listing 8: MarkerListener.java

```
1  package org.engine;
2
3  import com.tuio.*;
4  import javafx.animation.ScaleTransition;
5  import org.gamereact.module.Module;
6
7  public class MarkerListener implements TuioListener {
8
9    public MarkerListener() {}
10
11   @Override
12   public void addTuioObject(TuioObject tobj) {
13     Controller.objectList.put(tobj, new TangibleObject(tobj));
14   }
15
16   @Override
17   public void removeTuioObject(TuioObject tobj) {
18     Module module = Controller.objectList.get(tobj).getModule();
19     module.onTuioObjectRemoved(tobj);
20     Controller.connectionLineList.remove(module.getConnectionLine());
21     Controller.objectList.remove(tobj);
22   }
23
24   @Override
25   public void addTuioCursor(TuioCursor tcur) {
26     FingerTouchObject fingerTouchObject = new FingerTouchObject();
27     ScaleTransition cst = Transitions.createScaleTransition(50,
       fingerTouchObject, .5, 1);
28     Controller.cursorList.put(tcur, fingerTouchObject);
29     cst.play();
30   }
31
32   @Override
33   public void removeTuioCursor(TuioCursor tcur) {
34     Controller.cursorList.remove(tcur);
35   }
36
```

```
37   @Override
38   public void updateTuioObject(TuioObject tobj) {}
39   @Override
40   public void updateTuioCursor(TuioCursor tcur) {}
41   @Override
42   public void addTuioBlob(TuioBlob tblb) {}
43   @Override
44   public void updateTuioBlob(TuioBlob tblb) {}
45   @Override
46   public void removeTuioBlob(TuioBlob tblb) {}
47   @Override
48   public void refresh(TuioTime ftime) {}
49
50 }
```

Listing 9: TuioListener.java

```
38 /*
39 TUIO Java library
40 Copyright (c) 2005-2016 Martin Kaltenbrunner <martin@tuio.org>
41
42 This library is free software; you can redistribute it and/or
43 modify it under the terms of the GNU Lesser General Public
44 License as published by the Free Software Foundation; either
45 version 3.0 of the License, or (at your option) any later version.
46
47 This library is distributed in the hope that it will be useful,
48 but WITHOUT ANY WARRANTY; without even the implied warranty of
49 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
50 Lesser General Public License for more details.
51
52 You should have received a copy of the GNU Lesser General Public
53 License along with this library.
54 */
55
56 package com.tuio;
57
58 [...]
59
60 public interface TuioListener {
61
62   /**
63   * This callback method is invoked by the TuioClient when a new
       TuioObject is added to the session.
64   *
65   * @param  tobj  the TuioObject reference associated to the
       addTuioObject event
66   */
67   public void addTuioObject(TuioObject tobj);
68
69   /**
70   * This callback method is invoked by the TuioClient when an existing
       TuioObject is updated during the session.
71   *
72   * @param  tobj  the TuioObject reference associated to the
```

```
     updateTuioObject event
73   */
74   public void updateTuioObject(TuioObject tobj);
75
76   /**
77   * This callback method is invoked by the TuioClient when an existing
       TuioObject is removed from the session.
78   *
79   * @param  tobj  the TuioObject reference associated to the
       removeTuioObject event
80   */
81   public void removeTuioObject(TuioObject tobj);
82
83   /**
84   * This callback method is invoked by the TuioClient when a new
       TuioCursor is added to the session.
85   *
86   * @param  tcur  the TuioCursor reference associated to the
       addTuioCursor event
87   */
88   public void addTuioCursor(TuioCursor tcur);
89
90   /**
91   * This callback method is invoked by the TuioClient when an existing
       TuioCursor is updated during the session.
92   *
93   * @param  tcur  the TuioCursor reference associated to the
       updateTuioCursor event
94   */
95   public void updateTuioCursor(TuioCursor tcur);
96
97   /**
98   * This callback method is invoked by the TuioClient when an existing
       TuioCursor is removed from the session.
99   *
100  * @param  tcur  the TuioCursor reference associated to the
       removeTuioCursor event
101  */
102  public void removeTuioCursor(TuioCursor tcur);
103
104  /**
105  * This callback method is invoked by the TuioClient when a new
       TuioBlob is added to the session.
106  *
107  * @param  tblb  the TuioBlob reference associated to the addTuioBlob
       event
108  */
109  public void addTuioBlob(TuioBlob tblb);
110
111  /**
112  * This callback method is invoked by the TuioClient when an existing
       TuioBlob is updated during the session.
113  *
114  * @param  tblb  the TuioBlob reference associated to the
```

```
    updateTuioBlob event
115   */
116   public void updateTuioBlob(TuioBlob tblb);
117
118   /**
119   * This callback method is invoked by the TuioClient when an existing
        TuioBlob is removed from the session.
120   *
121   * @param  tblb  the TuioBlob reference associated to the
        removeTuioBlob event
122   */
123   public void removeTuioBlob(TuioBlob tblb);
124   /**
125   * This callback method is invoked by the TuioClient to mark the end
        of a received TUIO message bundle.
126   *
127   * @param  ftime  the TuioTime associated to the current TUIO message
        bundle
128   */
129   public void refresh(TuioTime ftime);
130 }
```

Listing 10: ModuleLibrary.java

```
1  // ModuleLibrary.java
2  package org.gamereact;
3
4
5  import javafx.util.Duration;
6
7  import org.engine.Command;
8  import org.engine.TangibleObject;
9  import org.gamereact.component.Track;
10 import org.gamereact.module.*;
11 import org.gamereact.module.Module;
12 import org.gamereact.module.electronic.*;
13 import org.w3c.dom.Element;
14 import org.w3c.dom.Node;
15 import org.w3c.dom.NodeList;
16
17 import java.util.Map;
18
19 /**
20 * holds the GameReact Module configuration
21 */
22 public class ModuleLibrary {
23
24   /**
25   * TangibleObject reference that holds both, the TuioObject and the
       corresponding Module
26   */
27   private static TangibleObject tangibleObject;
28   private static Node node;
29
30   /**
```

```
31  * Map containing the primary keys provided by the marker.xml config
      file and the module instance config
32  */
33  private static final Map<String, Command> MODULES;
34
35  static {
36
37    MODULES = Map.ofEntries(
38    Map.entry("AUDIO_PLAYER_MODULE", new Command() {
39      @Override
40      public Module create(TangibleObject tangibleObject) {
41        AudioPlayerModuleBuilder moduleBuilder = new
    AudioPlayerModuleBuilder(tangibleObject);
42        String fileName = ((Element) node).getAttribute("file");
43        String title = ((Element) node).getAttribute("title");
44        moduleBuilder
45        .setTitle(title)
46        .setFile(fileName)
47        ;
48        NodeList parts = ((Element) node).getElementsByTagName("part");
49        for (int k = 0; k < parts.getLength(); k++) {
50          Node part = parts.item(k);
51          String partName = ((Element) part).getAttribute("name");
52          int startDuration = Integer.parseInt(((Element)
    part).getAttribute("start"));
53          int endDuration = Integer.parseInt(((Element)
    part).getAttribute("end"));
54          moduleBuilder.addTrack(new Track(partName, new
    Duration(startDuration), new Duration(endDuration)));
55        }
56        return moduleBuilder.createAudioPlayerModule();
57      }
58    }),
59    Map.entry("IMAGE_MODULE", new Command() {
60      @Override
61      public Module create(TangibleObject tangibleObject) {
62        String title = ((Element) node).getAttribute("title");
63        ImageModuleBuilder imageModuleBuilder = new
    ImageModuleBuilder().setTangibleObject(tangibleObject);
64        imageModuleBuilder.setTitle(title);
65        NodeList images = ((Element)
    node).getElementsByTagName("image");
66        for (int k = 0; k < images.getLength(); k++) {
67          Node image = images.item(k);
68          String imageName = ((Element) image).getAttribute("name");
69          String imageFile = ((Element) image).getAttribute("file");
70          imageModuleBuilder.addImage(imageFile, imageName);
71        }
72        return imageModuleBuilder.createImageModule();
73      }
74    }),
75    Map.entry("VOLUME_CONTROL_MODULE", VolumeControlModule::new),
76    Map.entry("AXIS_SCROLL_CONTROL_MODULE",
    AxisScrollControlModule::new),
```

```
77      Map.entry("ROTATION_SIGNAL_OUTPUT_MODULE",
        RotationSignalOutputModule::new),
78      Map.entry("ARDUINO_CONTROL_MODULE", ArduinoControlModule::new),
79      Map.entry("LED_COMPONENT_MODULE", LEDModule::new),
80      Map.entry("TRANSISTOR_COMPONENT_MODULE", TransistorModule::new),
81      Map.entry("RESISTOR_COMPONENT_MODULE", ResistorModule::new),
82      Map.entry("CAPACITOR_COMPONENT_MODULE", CapacitorModule::new),
83      Map.entry("COIL_COMPONENT_MODULE", CoilModule::new),
84      Map.entry("INDUCTOR_COMPONENT_MODULE", InductorModule::new),
85      Map.entry("BATTERY_COMPONENT_MODULE", BatteryModule::new),
86      Map.entry("CHART_MODULE", ChartModule::new)
87      );
88    }
89
90    /**
91    * Constructor
92    * @param tanObj the TangibleObject reference that will be aggregated
          to the desired module
93    */
94    public ModuleLibrary(TangibleObject tanObj) {
95      tangibleObject = tanObj;
96    }
97
98    /**
99    * creates a new Module object by using the instance configuration of
          the MODULES map
100   * @param objectNode current object node found in marker.xml config
          file
101   * @return returns a new Module instance
102   */
103   public Module createModule(Node objectNode) {
104     node = objectNode;
105     String moduleName = ((Element) objectNode).getAttribute("class");
106     Command command = MODULES.get(moduleName);
107
108     if (command == null) {
109       throw new IllegalArgumentException("Ungueltiger Modulname: " +
        moduleName);
110     }
111
112     return command.create(tangibleObject);
113   }
114
115 }
```

Stichwortverzeichnis

Literaturverzeichnis

Bücher

[6] Martin Kaltenbrunner und Ross Bencina. *ReacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction*. TEI '07. Baton Rouge, Louisiana: Association for Computing Machinery, 2007, 69–74. ISBN: 9781595936196. DOI: 10.1145/1226969.1226983. URL: https://doi.org/10.1145/1226969.1226983.

[11] Carsten Steger, Markus Ulrich und Christian Wiedemann. *Machine vision algorithms and applications*. Weinheim, Germany: Wiley-VCH, 2018. ISBN: 978-3-527-41365-2. URL: https://d-nb.info/1140659693.

Sammlungen

[2] Julius Autz u. a. „The pitfalls of transfer learning in computer vision for agriculture". In: *42. GIL-Jahrestagung, Künstliche Intelligenz in der Agrar- und Ernährungswirtschaft*. Bonn: Gesellschaft für Informatik e.V., 2022, S. 51–56. ISBN: 978-3-88579-711-1.

[8] Martin Kaltenbrunner u. a. „TUIO - A Protocol for Table Based Tangible User Interfaces". In: *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)*. Vannes, France, 2005.

[13] Rolf P. Würtz. „Organic computing for face and object recognition". In: *Informatik 2004, Informatik verbindet, Band 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*. Bonn: Gesellschaft für Informatik e.V., 2004, S. 636–640. ISBN: 3-88579-380-6.

Websites

[1] Arduino. *Arduino Hardware*. Hrsg. von Arduino. Online erhältlich unter https://www.arduino.cc/en/hardware; abgerufen am 13. Januar 2024, 13:17 Uhr.

[3] Peter Gerwinski. *Lesser General Public License*. Hrsg. von GNU. Online erhältlich unter https://www.gnu.de/documents/lgpl-3.0.de.html; abgerufen am 05. August 2023, 15:47 Uhr.

[4] Martin Kaltenbrunner. *Reactable Live*. Hrsg. von Reactable Systems. Online erhältlich unter http://reactable.com/live; abgerufen am 24. Juli 2023, 10:45 Uhr.

[5] Martin Kaltenbrunner. *TUIO Protokoll*. Hrsg. von Martin Kaltenbrunner. Online erhältlich unter https://tuio.org/; abgerufen am 07. August 2023, 16:26 Uhr.

[7] Martin Kaltenbrunner u. a. *Reactable*. Hrsg. von Kunstuniversität Linz. Online erhältlich unter https://www.kunstuni-linz.at/galerie/arbeiten-1/medien/interface-cultures/2005/reactable; abgerufen am 24. Januar 2024, 07:40 Uhr.

[9] Dr. Karl Lamprecht. *Zeiss Archiv - Belsazar*. Hrsg. von Carl Zeiss AG. Online erhältlich unter https://www.archive.zeiss.de/objekt_start.fau?prj=zeiss&dm=Virtuelles+Museum&ref=14938; abgerufen am 07. Dezember 2023, 18:06 Uhr.

[10] Hartmut Schlosser. *Wo steht JavaFX im Vergleich zu anderen UI-Toolkits: Swing, HTML5, SWT?* Hrsg. von Software und Support Media GmbH. Online erhältlich unter https://entwickler.de/java/wo-steht-javafx-im-vergleich-zu-anderen-ui-toolkits-swing-html5-swt; abgerufen am 06. August 2023, 09:23 Uhr.

[12] Manuel Wöpke. *Wenn Besucher Künstler werden.* Hrsg. von Axel Springer SE. Online erhältlich unter https://www.welt.de/welt_print/vermischtes/article4783735/Wenn-Besucher-Kuenstler-werden.html; abgerufen am 10. Juni 2023, 17:30 Uhr.