UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Entwicklung einer Visualisierung der Eingabe bei Touchscreen-Nutzung auf Basis von Kinect-Daten

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Katrin Meng

Entwicklung eines Systems zur Kollaboration auf entfernten Touchscreeneingabegeräten

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science (B.Sc.)
im Studiengang Informatik

vorgelegt von
Arne Reepen

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	M.Sc. Gerrit Lochmann (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Juli 2014

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

. . . . . .
(Ort, Datum) (Unterschrift)

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

. . . . . .
(Ort, Datum) (Unterschrift)

Institut für Computervisualistik
AG Computergraphik
Prof. Dr. Stefan Müller
Postfach 20 16 02
56 016 Koblenz
Tel.: 0261-287-2727
Fax: 0261-287-2735
E-Mail: stefanm@uni-koblenz.de

UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

Aufgabenstellung für die Bachelorarbeit

Katrin Meng

(Mat. Nr. 209 210 613)

Thema: Entwicklung einer Visualisierung der Eingabe bei Touchscreen-Nutzung auf Basis von Kinect-Daten

Die Eingabe per Touchscreen findet bei vielen Geräten wie Smartphones, Tablets und auch Computern Verwendung. Die Arbeitsweise unterscheidet sich durch die Nutzung von Gesten grundlegend von der Bedienung mit Maus und Tastatur. Das Display wird hierbei immer durch die Eingabe-Hand verdeckt. Dies führt zu Problemen, wenn eine weitere Person dem Displayinhalt folgen möchte. Sieht man jedoch nur den Bildschirm, zum Beispiel durch Videoübertragung, wird lediglich das Resultat sichtbar und nicht die Eingabe selbst. Eine Vermittlung der richtigen Anwendung ist nur schwer möglich.

Arbeitet man zeitgleich von mehreren Geräten aus an einem Projekt, ist es ebenfalls von Vorteil den Positionen und Aktionen der Anderen gut folgen zu können.

Ansatz dieser Arbeit ist die Erstellung einer transparenten Visualisierung der Hand des Nutzers. Durch Änderung von Farbe und Opazität sollen Gesten und Berührungen des Bildschirms verdeutlicht werden. Hierzu wird das Tiefenbild einer Microsoft Kinect genutzt.

Ziel der Arbeit ist es zu evaluieren, ob Gesten und Bewegungen durch die Visualisierung der Hände eindeutig dargestellt werden ohne den Bildschirminhalt zu stören. Hierzu wird die Visualisierung in eine Testumgebung eingebunden, die für das kollaborative Arbeiten ausgelegt ist.

Insbesondere sollen die Fragen beantwortet werden ob die Darstellung grafisch ansprechend und klar zu deuten ist, ob der Bildschirminhalt weiterhin gut zu erkennen ist und ob die Hände mehrerer Anwender gut unterscheidbar sind.

Die inhaltlichen Schwerpunkte der Arbeit sind:

1. Recherche über Toucheingabe, Gesten und kollaborative Arbeit an einem Bildschirm
2. Setup der Hardware
3. Herstellung einer Visualisierung der Hand
4. Evaluation hinsichtlich der grafischen Qualität
5. Dokumentation der Ergebnisse

Koblenz, 05.12.2013

K. Meng

– Katrin Meng –

S. Müller

– Prof. Dr. Stefan Müller –

Institut für Computervisualistik
AG Computergraphik
Prof. Dr. Stefan Müller
Postfach 20 16 02
56 016 Koblenz
Tel.: 0261-287-2727
Fax: 0261-287-2735
E-Mail: stefanm@uni-koblenz.de

UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

Aufgabenstellung für die Bachelorarbeit
Arne Reepen
(Mat. Nr. 210 100 147)

Thema: Entwicklung eines Systems zur Kollaboration auf entfernten Touchscreeneingabegeräten

Die Eingabe per Touchscreen findet bei vielen Geräten wie Smartphones, Tablets und auch Computern Verwendung. Die Arbeitsweise unterscheidet sich durch die Nutzung von Gesten grundlegend von der Bedienung mit Maus und Tastatur. Das Display wird hierbei immer durch die Eingabe-Hand verdeckt. Dies führt zu Problemen, wenn eine weitere Person dem Displayinhalt folgen möchte. Sieht man jedoch nur den Bildschirm, zum Beispiel durch Videoübertragung, wird lediglich das Resultat sichtbar und nicht die Eingabe selbst. Eine Vermittlung der richtigen Anwendung ist nur schwer möglich.

Arbeitet man zeitgleich von mehreren Geräten aus an einem Projekt, ist es ebenfalls von Vorteil den Positionen und Aktionen der Anderen gut folgen zu können.

Ansatz dieser Arbeit ist die Erstellung einer Testumgebung für die Zusammenarbeit mehrerer Touchscreen-Eingabegeräte. Durch eine transparente Visualisierung der Hand sollen die Eingaben der Nutzer dargestellt werden.

Ziel der Arbeit ist es, zu evaluieren, ob die erstellte Anwendung einen Gewinn für die kollaborative Arbeit und Wissensvermittlung mit Touchscreen-Geräten bietet.

Insbesondere sollen die Fragen beantwortet werden, ob es leichter ist dem Bildschirminhalt zu folgen, ob Wissen besser zu übermitteln ist und ob eine effektive Zusammenarbeit möglich ist.

Die inhaltlichen Schwerpunkte der Arbeit sind:

1. Recherche über Toucheingabe, Gesten und kollaborative Arbeit an einem Bildschirm
2. Setup der Hardware
3. Herstellung eines System für die Zusammenarbeit mehrerer Personen
4. Evaluation hinsichtlich des Mehrwertes
5. Dokumentation der Ergebnisse

Koblenz, 05.12.2013

A. Reepen

– Arne Reepen –

S. Müller

– Prof. Dr. Stefan Müller –

Zusammenfassung

In den letzten Jahren ist eine steigende Verbreitung von Touchscreen-Geräten zu verzeichnen. Ihre Bedienung unterscheidet sich grundlegend von der mit Maus und Tastatur. Durch die Eingabe mit Gesten oder mehreren Fingern kann es schwierig sein den Aktionen eines Anderen zu folgen. Probleme entstehen durch die Verdeckung des Bildschirms mit der Eingabehand. Sieht man nur den Bildschirminhalt, zum Beispiel bei einer Videoübertragung, gehen Informationen über die Eingabe verloren.

In dieser Arbeit wird ein System entwickelt, das die kollaborative Arbeit an voneinander entfernten Touchscreen-Geräten verbessern soll. Dazu wird aus den Tiefendaten eines Kinect Sensors eine grafische Repräsentation der Eingabehand erstellt. Durch Einblendung dieser Visualisierung soll es einem Anwender erleichtert werden den Eingaben eines Anwenders zu folgen. Bedienkonzepte, wie zum Beispiel Gesten, sollen dadurch besser vermittelt werden. Außerdem soll so die Möglichkeit geschaffen werden, Informationen über eine gemeinsame Problematik effizienter auszutauschen. Deshalb wurde ein Testsystem mit zwei Arbeitsplätzen entwickelt. Darin übernimmt ein Anwender die Rolle des Erklärenden und führt einen zweiten Anwender, den Ausführenden, durch verschiedene Testszenarien. Hierbei stehen ihm bei einem Teil der Aufgaben die Visualisierung der Hand zur Verfügung, während er in anderen Aufgaben nur verbal mit seinem Gegenüber kommunizieren kann.

Im Rahmen einer Evaluation wird das System auf seine Effizienz zur Bedienung von Touchscreen-Systemen überprüft. Des Weiteren wird untersucht, inwieweit die grafische Qualität den gestellten Anforderungen genügt, um einen Mehrwert für die Anwendung zu bieten.

Abstract

Within the last years the number of touchscreen devices increased. Their handling is fundamentally different from mouse and keyboard. When using gestures or multiple fingers for input, it can be difficult to follow the actions of another person. Problems occur, when blocking the display with the hand used for input. If one is just able to see the content of the screen, for example in a video transmission, information about the actual input is lost.

This thesis aims to create a system that enhances collaborative work on remote touchscreen devices. To achieve this, the depth data of a Kinect sensor is used to create a graphical representation of the hand used for input. Displaying this visualization shall simplify it for a user to follow the actions made by another user. This is used to clarify operating concepts like gestures. Furthermore this should be a way to efficiently exchange information about shared problems. To achieve this, a testing environment with two workplaces was created. One user acts a an instructor, while he guides another one trough various test scenarios. In some of those tests the instructor has the chance to use the visualization of his hand, while in other he is limited to use only his voice.

An evaluation will check the system on its efficiency, while using touchscreen-devices. Also it will be evaluated if the graphical fidelity is sufficient to result in an increased value for the system.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Seitdem die ersten Computer gebaut werden, ist die Mensch-Maschine-Interaktion ein wichtiger Bestandteil der Forschung und Entwicklung. Selbst der beste Rechner ist für den Menschen wertlos, wenn er diesen nicht zielgerichtet bedienen kann. Um die Funktionalität der Maschine effektiv nutzen zu können, muss eine geeignete Schnittstelle bereitgestellt werden. Für diese Schnittstellen stehen neben der reinen Funktionalität auch Dinge wie Ergonomie und Benutzerfreundlichkeit (engl. Usability) im Vordergrund. Hier spielen Ein- und Ausgabegeräte eine große Rolle. Über die Zeit haben sich diese immer weiter entwickelt, um die Bedienung von Computern effizienter und einfacher zu gestalten. [Bol09]
Angefangen über die reine Eingabe über eine Kommandozeile (*Command Line Interface, CLI*) entwickelten sich grafische Bedienoberflächen (*Graphical User Interface, GUI*), die über die Maus gesteuert werden. Als nächster großer Schritt und Vereinfachung der Kommunikation von Mensch und Computer gelten die *Natural User Interfaces (NUIs)*, die über einen oder mehrere Sinne des Benutzers direkt angesprochen werden. Beispiele hierfür sind die Bedienung über Berührung (engl. touch) bei Touchscreens, Spracherkennung, Erkennung von Gesten der Hände oder des gesamten Körpers. [KASA08] Schwerpunkt in dieser Arbeit ist die Benutzung von Geräten mit Touchscreen.
Heutzutage sind Geräte, die über Berührung gesteuert werden, weit verbreitet. Das erste Multi-Touch-Display wurde bereits 1982 von Nimish Meta von der Toronto University entwickelt. [Bux07] Doch erst mit Erscheinen des iPhone im Jahr 2007 wurde die Multi-Touch-Technologie für die Allgemeinheit zugänglich. [Sch10]
Durch die großen Fortschritte in der Entwicklung von Mikroprozessoren, Speicher und Kameras, können viele Geräte preisgünstiger produziert werden und die Verbreitung von Smartphones und Tablets nimmt weiter zu. Somit gewinnt die Eingabe über Touchscreens mehr Bedeutung und ist für immer mehr Menschen relevant. Auch bei nicht mobilen Endgeräten, wie zum Beispiel Fahrkartenautomaten und Bankautomaten, findet man immer mehr auf Berührung basierende Bedienung. Durch die direkte Benutzung der Hand, ohne den Umweg über zusätzliche Eingabegeräte, entsteht eine natürlichere und effizientere Bedienung der Geräte. Im Gegensatz zur herkömmlichen Maus, die nur 2 Freiheitsgrade bietet, ermöglicht die Nutzung der Hand 27 Freiheitsgrade. [ES03] Dadurch entstehen komplexere Möglichkeiten von Eingabekonzepten. Das macht die Eingabe über Touchscreens sehr flexibel und in vielen Anwendungen einsetzbar.
Betrachtet man bisherige GUIs, ist eine wichtige Regel die Sichtbarkeit. Über Elemente wie Menüs und Buttons sind alle möglichen Interaktionen für den Nutzer einzusehen. Das ermöglicht es dem Nutzer ein unbekanntes System ohne zusätzliche Hilfe zu erkunden und alle Funktionen eigenständig zu entdecken. Diese Sichtbarkeit aller verfügbaren Funktionen ist bei Schnittstellen, die auf Bedienung durch Berührung ausgelegt sind, nicht zwangsläufig gegeben.

Es stellt sich die Frage, ob Natural User Interfaces, insbesondere Bedienung über Berührungen und Gesten wirklich „natürlich“ und intuitiv anwendbar sind. Zumeist sind die Eingabemechanismen und verwendete Gesten durch Konventionen festgelegt. [MWW10] Dadurch sind sie stark abhängig von Vorkenntnissen und Wissen des Nutzers. Im Zusammenhang mit der fehlenden Sichtbarkeit aller Funktionen sind diese, für ungeübte Nutzer, schwerer selbst zu erforschen. Es besteht Bedarf an Tutorials und Anleitungen. [Nor10] Die Vermittlung der richtigen Bedienung gestaltet sich jedoch schwieriger als mit Maus und Tastatur. In Multi-Touch-Geräten sind Ein-und Ausgabegerät vereint. Durch die Eingabehand wird dadurch immer auch der Inhalt der Ausgabe verdeckt. Für einen Beobachter wird es schwerer dem Geschehen zu folgen. Sieht man nur den Bildschirminhalt, zum Beispiel bei Videoübertragungen, sieht man zwar das Resultat der Eingabe, jedoch nicht wie dieses erreicht wird. Es fehlt ein Zeigewerkzeug, wie es bisher die Maus ermöglicht. Dadurch ist es schwierig die richtige Anwendung und nötige Gesten zu vermitteln.
Ein weiterer Anwendungsfall, bei dem ein Zeigewerkzeug hilfreich ist, ist Kollaboration. Durch zunehmende Globalisierung gibt es, gerade in der Arbeitswelt, immer mehr internationale Teams, die räumlich getrennt an gemeinsamen Projekten und Aufgaben arbeiten. Die Effizienz bei entfernten, kollaborativen Aufgaben ist stark von den Kommunikationsmöglichkeiten der Teilnehmer abhängig. In [VGF99] wird gezeigt, dass sich die Qualität der Kommunikation mit der zusätzlichen Nutzung von Videos im Gegensatz zu rein verbalen Erklärungen deutlich verbessert. Erklärungen von Sachverhalten gewinnen dadurch an Effizienz und ermöglichen die schnellere Lösung von Aufgaben.
Wir benutzen unsere Hände, um zu verdeutlichen auf welche Dinge wir Bezug nehmen, seien sie real oder virtuell. Diese Interaktionen geben wichtige Hinweise bei der gemeinsamen Lösung von Aufgaben und insgesamt bei der Kommunikation. Bei den Touchscreen-Anwendungen werde Gesten genutzt, um bestimmte Aktionen auszuführen. Diese Informationen fehlen bei Telefonkonferenzen und geteilten Bildschirminhalten ohne Zeigewerkzeug. [FSY+04] [VGF99]

1.2 Zielsetzung

Ziel dieser Arbeit ist es herauszufinden, ob sich die kollaborative Arbeit an Touchscreen-Geräten verbessern lässt. Speziell die Probleme der Verdeckung von Bildschirminhalten, das Fehlen eines Zeigewerkzeugs und das Vermitteln von Bedienkonzepten sollen dabei im Vordergrund stehen. Dafür soll ein System zur kollaborativen Arbeit an zwei entfernten Geräten erstellt werden. In diesem soll eine Visualisierung der Eingabehand des einen Benutzers für den Benutzer am anderen Gerät sichtbar werden. Es soll eine Testanwendung erstellt werden, die unterschiedliche Anwendungsfälle simuliert. Dabei sollen die folgenden Hypothesen im Vergleich zur Benutzung ohne die eingeblendete Hand überprüft werden.

H1 Die Einblendung der Hand trägt zur effizienteren Ausführung von Touchscreen-Anwendungen bei.

- **H1.1** Wissen kann effektiver zwischen den Nutzern ausgetauscht werden.
- **H1.2** Bedienkonzepte lassen sich besser vermitteln.
- **H1.3** Die Aufmerksamkeit des Gegenübers lässt sich besser auf bestimmte Bildschirminhalte lenken.

H2 Die grafische Qualität der Visualisierung der Hand erfüllt die nötigen Anforderungen, um einen Vorteil in der Anwendung zu bieten.

- **H2.1** Der Bildschirminhalt ist jederzeit gut erkennbar.
- **H2.2** Position und Ausrichtung der *Hand* werden deutlich.
- **H2.3** Es wird ersichtlich welche Eingaben von der *Hand* gemacht werden.
- **H2.4** Die Hände mehrerer Anwender können gut unterschieden werden.

1.3 Verwandte Arbeiten

Viele Arbeiten haben untersucht, inwieweit weitere Informationen zur gemeinsamen Bearbeitung von Aufgaben oder zur Erklärung bereitgestellt werden können, um eine Hilfestellung zu bieten. Meist wurde dies durch das Überblenden der eigentlichen Aufgabe mit den zusätzlichen Bildinhalten umgesetzt. Sei es in einer virtuellen Umgebung oder als Erweiterung der realen Welt. Im Folgenden werden einige dieser Arbeiten kurz vorgestellt.

BeThere: 3D Mobile Collaboration with Spatial Input
Ein Prototyp für Kollaboration im dreidimensionalen Raum. Der Benutzer hat Zugriff auf die wirkliche Umgebung des Anderen. Diese wird per Kinect aufgenommen und in ein 3D-Objekt konvertiert, dafür wird *KinectFusion*(siehe [Netb]) verwendet. Das 3D-Model wird an den entfernten Benutzer gesendet. Die Position und Orientierung von diesem wird getrackt. Somit kann er sich unabhängig von dem anderen Anwender in der virtuellen Umgebung umschauen. Er hat die Möglichkeit in dieser Umgebung zu navigieren und für den Anwender ein statisches Model einer Hand anzeigen, die auf den entsprechenden Punkt in der realen Welt zeigt. [SJF+13]

Gestures Over Video Streams to Support Remote Collaboration on Physical Tasks
Ein Videosystem, um das kollaborative Arbeiten an physikalischen Objekten zu vereinfachen. Es werden zwei Möglichkeiten der Hinweisgebung zur Verfügung gestellt. Einmal per Cursor-ähnlichem Zeige-Werkzeug und per freier Stifteingabe, um Zeichnungen zu übermitteln. Dies ermöglicht das Lenken der Aufmerksamkeit durch Zeigen und das Zeichnen von Gesten. Es wurde gezeigt, dass reines

Zeigen nicht ausreicht, um eine Verbesserung zu erreichen. Das Zeichnen, durch das sowohl Zeigen als auch Gesten erzeugt werden können, eignete sich gut, um Kommunikation und damit die Effizienz zu verbessern. [FSY+04]

VideoDraw: A Video Interface for Collaborative Drawing
System zur Simulation eines gemeinsamen, virtuellen „Skizzenbuchs“. Über Videoübertragung werden nicht nur die Zeichnungen des Anderen, sondern auch seine Hände übertragen. Diese werden über die eigene Zeichenfläche geblendet. Somit kann man die Entstehung der Zeichnungen des Anderen mitverfolgen, beeinflussen und diskutieren. Auch hier wurde gezeigt, dass die Hände viel und produktiv zur Verdeutlichung und Unterstützung benutzt wurden. Zum Beispiel, um schneller auf Dinge aufmerksam zu machen und Regionen zu kennzeichnen. [TM90]

C-Slate: A Multi-Touch and Object Recognition System for Remote Collaboration using Horizontal Surfaces
Ein bildbasiertes System zur Kollaboration und Unterstützung von Video- und Audio-Konferenzen. Es wird eine Stereo-Kamera über einem nicht berührungsempfindlichen Tablet verwendet. Durch die Stereo-Aufnahmen werden Tiefenwerte berechnet. Objekte des Vordergrunds werden segmentiert. Das System kann die Hand von anderen Objekten unterscheiden und durch ihre Pose bestimmte Aktionen auslösen. Auch andere Objekte werden erkannt und als Eingabe benutzt. Die Hand des Anwenders wird übertragen und für den Anderen sichtbar. Hierbei wird die Transparenz verwendet, um einen Tiefeneindruck für den entfernten Benutzer zu erzeugen. Verdeckung kann so gezielt durch die Höhe der Objekte vom Benutzer beeinflusst werden. [IAC+07]

2 Grundlagen

2.1 Kommunikation zur Informationsvermittlung

Zwischenmenschliche Kommunikation ist effizienter, je mehr Gemeinsamkeiten die beiden Teilnehmer haben. Dies können gemeinsames Wissen, Überzeugungen, Ziele, Zugehörigkeit zu der selben Gruppe oder ähnliches sein. Diese Gemeinsamkeiten entwickeln sich auch durch und während der Ausführung gemeinsamer Aufgaben, aufgrund von *linguistischer* und *räumlicher* Co-Präsenz. Es muss eine Grundlage zur erfolgreichen Kommunikation geschaffen werden. Zum Beispiel müssen Objekte mit ähnlichen Ausdrücken benannt werden, um diese richtig zu identifizieren. [FK92] [FSY+04] Um dies zu erleichtern, benutzen Menschen Gesten, um ihre Aussagen zu unterstützen und so für den Anderen zu verdeutlichen. Es gibt zeigende Gesten, die dazu verwendet werden auf Objekte oder Regionen hinzuweisen, und repräsentative Gesten, die durch Handform und Handbewegungen ausgedrückt werden. [FSY+04]

Abbildung 1: Aufbau des Kinect-Sensors [uWB]

2.2 Microsoft Kinect

Kinect wurde zusammen von *Microsoft* und der Firma *PrimeSense* entwickelt und am 4.November 2010 als Eingabegerät für die Xbox 360 veröffentlicht. So wurde es möglich, Spiele nur durch seine Stimme und Körperbewegungen zu steuern. Durch die gute Verfügbarkeit und den niedrigen Preis fand Kinect jedoch in kurzer Zeit in vielen weiteren Bereichen Verwendung. Im Februar 2012 wurde eine weitere Version, *Kinect for Windows*, und ein offizielles *Kinect Software Development Kit(SDK)* veröffentlicht. Dies erleichtert es weiteren Entwicklern den Kinect Sensor für ihre Projekte zu nutzen. Die Kombination aus Farb- und Tiefendaten eröffnet in vielen Anwendungsfällen große Möglichkeiten. Beispielsweise in der Informatik, Medizin und Robotik. Microsoft spricht von dem Kinect-Effekt. [Zha12] Die Verkaufszahlen liegen im Februar 2013 bei 24 Millionen verkauften Exemplaren. [Cen]
Zu den technischen Daten des Kinect Sensors, wie in Abbildung 1 zu sehen: Es sind ein Tiefensensor, eine Farbkamera, ein Mikrofon-Array und ein Neigungs-Motor verbaut. Die Farbkamera liefert 10 Frames pro Sekunde mit RGB-Bildern in einer Auflösung von 1280*1024 Pixeln, 30 Frames bei einer Auflösung von 640*480. Die maximale Auflösung der Tiefenbilder entspricht 640*480 und ist ebenfalls mit 30 Frames pro Sekunde abrufbar. Kinect hat ein Öffnungswinkel von 43° in vertikaler und 57° in horizontaler Richtung. [Neta] Die Reichweite des Tiefensensors liegt bei 0,8m bis 4m, in diesem Bereich liefert der Sensor brauchbare Werte. Kinect for Windows verfügt zusätzlich über einen Near Mode, in dem die Reichweite bereits bei 0,4m beginnt. [fWB]
Die Erfassung der Tiefendaten erfolgt über den Tiefensensor und die von PrimeSense entwickelte Technologie *Structured Light*. Der Tiefensensor besteht aus einem Infrarot-Projektor und einem CMOS-Sensor, der als Infrarot-Kamera dient. Zur Berechnung der Tiefen wird ein bekanntes Muster aus Punkten durch den Projektor ausgestrahlt. Dieses ist für das menschliche Auge unsichtbar. Mit der Infrarot-Kamera wird das, durch vorhandenen Objekte verzerrte, Muster aufgenommen. Dieses wird mit dem ursprünglichen Muster verglichen und aus diesen

Unterschieden die Tiefen berechnet. In das Tiefenbild werden die Entfernungen zum nächstgelegenen Objekt in Millimetern geschrieben. [Zha12]
Kinect liefert jedoch nicht immer verlässliche Tiefenwerte. Zu Problemen bei der Datenerfassung kommt es zum Beispiel im Freien oder bei starker Sonneneinstrahlung, da der Infrarotteil des Sonnenlichts das projizierte Muster überstrahlt. Auch bei stark reflektierenden Oberflächen kommt es zu unbestimmten Werten. Außerdem entstehen Lücken im Tiefenbild, die durch die versetzte Position von Projektor und Kamera entstehen. Punkte, die von der Kamera aufgenommen werden können aus Sicht des Projektors verdeckt sein und so kein Muster aufweisen. Diese Punkte bleiben im Tiefenbild folglich ohne Information. [KE12]

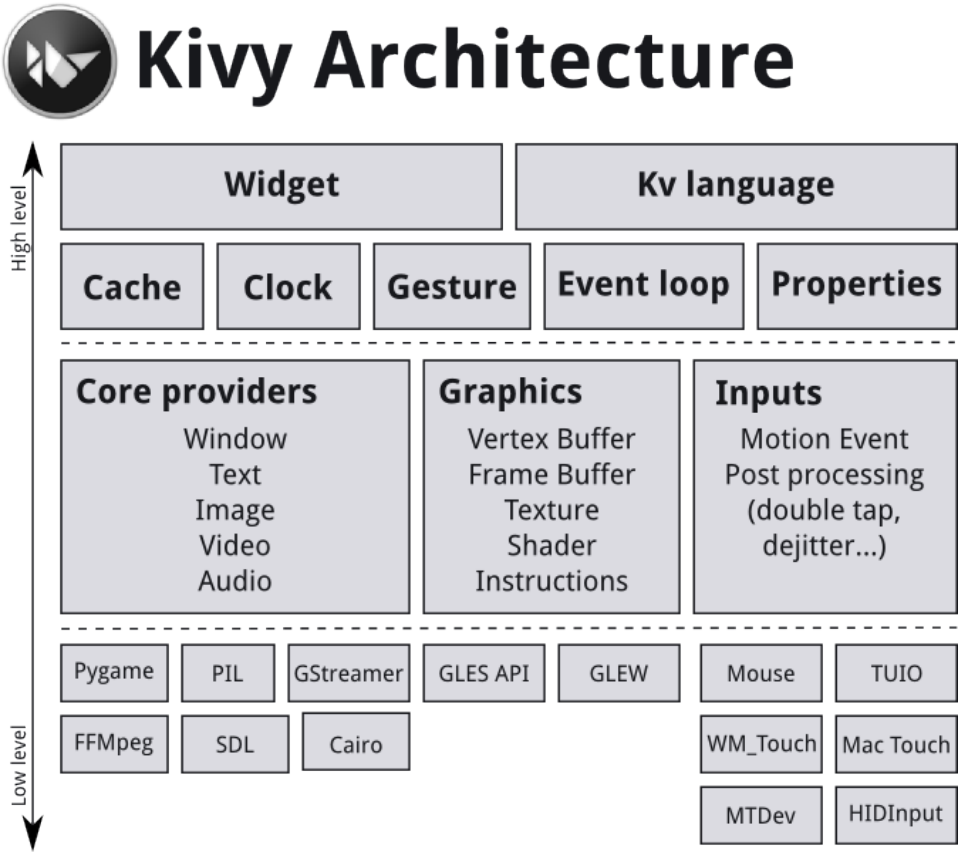

Abbildung 2: Kivy Architektur [Gui14]

2.3 Kivy

2.3.1 Geschichte

Kivy ist der konzeptuelle Nachfolger von *PyMT* und wurde Anfang 2011 in seiner ersten Version vorgestellt. Wie schon zuvor PyMT, ist Kivy ein Open Source Python Framework für die Entwicklung von Multi-Touch-Anwendungen. Obwohl es die gleichen Ideen verfolgt, die schon in PyMT angestrebt wurden, bietet es grundlegende Änderungen in der Codebasis. Diese führen zu deutlichen Geschwindigkeits-Vorteilen, verhindern allerdings auch die Rückwärtskompatibilität zu PyMT Projekten. Aufgrund dessen wurde mit Kivy ein neues Projekt der gleichen Entwickler ins Leben gerufen. Im Jahr 2012 hat das Entwicklerteam 5000 $ von der Python Software Foundation erhalten, um Kivy auf Python 3.3 zu portieren. Kivy Projekte können für Windows, Linux, OS X sowie Android und iOS erstellt werden. Aktuell ist die stabile Version 1.8.0 vom Januar 2014 verfügbar. [Gui]

2.3.2 Aufbau und Architektur

Die Kivy Architektur basiert auf Modularität und Abstraktion (siehe Abbildung 2). Grundlegende Funktionen, wie die Verwaltung von Fenstern oder das Anzeigen eines Bildes, werden gekapselt und unter den sogenannten *Core Providern* zusammengefasst. Diese sprechen betriebssystemspezifische APIs an und agieren als Kommunikationsschicht zwischen dem Betriebssystem und Kivy. Dies ermöglicht plattformunabhängigen Code und macht das System offen für Erweiterungen. Für viele Funktionen nutzt Kivy bestehende Python Module, um die eigene Codebasis so kompakt wie möglich zu halten und die Portierung zwischen Plattformen zu vereinfachen. Neben den *Core Providern* gibt es *Input Provider*, diese bilden die Schnittstelle für zahlreiche Eingabesysteme wie Maus, Tastatur, Touchscreens und Trackpads. Dies ermöglicht einfaches Hinzufügen von neuen Eingabegeräten oder die Unterstützung bekannter Geräte auf anderen Plattformen. [Gui14]
Kivy verwendet eine eigene Grafik API, die auf OpenGL basiert. Diese ist aus Gründen der Performanz in C programmiert. Der Anwender hat die Möglichkeit reine OpenGL Kommandos zu nutzen, dies wird jedoch nicht empfohlen, da die Kivy eigenen Kommandos besser optimiert sind. Sie vereinfachen die Benutzung deutlich, da nicht mit Vertex Kommandos gearbeitet werden muss, sondern für viele Grafikelemente wie Rechtecke und Kreise eigene Kommandos vorliegen. [Gui14]
Funktionalitäten werden in Kivy über *Widgets* realisiert. Diese sind das zentrale Interface Element von Kivy, müssen jedoch nicht zwangsläufig über eine grafische Repräsentation verfügen. Kivy bietet bereits eine Vielzahl von Widgets, wie Buttons, Texteingabefelder oder Filebrowser, an. Auch andere Python Module, wie zum Beispiel Audio- und Video Player des GStreamer Moduls, werden als Widgets umgesetzt. Um Widgets anzuordnen, werden *Layouts* verwendet. Diese können auch ineinander verschachtelt werden, um die Funktionen verschiedener Layout Typen, wie *GridLayout* oder *FloatLayout*, miteinander zu kombinieren. Al-

le Widgets werden in einem *Widget Baum* verwaltet. Dieser Baum besitzt einen Wurzelknoten mit beliebig vielen Widgets als direkte oder indirekte Kinder. Bei Interaktionen mit dem Programm werden *Events* ausgelöst, die von den Widgets verarbeitet werden. Wenn ein Event eingeht, wird dieses von der Wurzel über den Baum weitergeleitet. An jedem Knoten wird entschieden, ob das Event verarbeitet und verbraucht oder weitergeleitet wird. [Gui14]

Die Abbildung 3 zeigt eine sehr einfache Kivy Applikation und das Codebeispiel 1 den zugehörigen Programmcode.
Die Kivy App *MyApp* gibt in ihrer build()-Funktion *MyWidget* zurück. In der Klasse MyWidget wird ein rotes Rechteck, sowie ein Button mit der Aufschrift „Start“ erzeugt.

```
1  from kivy.app import App
2  from kivy.uix.widget import Widget
3  from kivy.uix.button import Button
4  from kivy.graphics import Color, Rectangle
5
6  class MyWidget(Widget):
7      def __init__(self, **kwargs):
8          super(MyWidget, self).__init__(**kwargs)
9          with self.canvas:
10             Color(1.0, 0.0, 0.0, 1.0)
11             Rectangle(pos= (200,200), size= (400,300))
12             Button(pos= (350,100), size= (100,50), text = "Start")
13
14 class MyApp(App):
15     def build(self):
16         return MyWidget()
17
18
19 if __name__ == '__main__':
20     MyApp().run()
```

Codebeispiel 1: Code für einfache Kivy Applikation

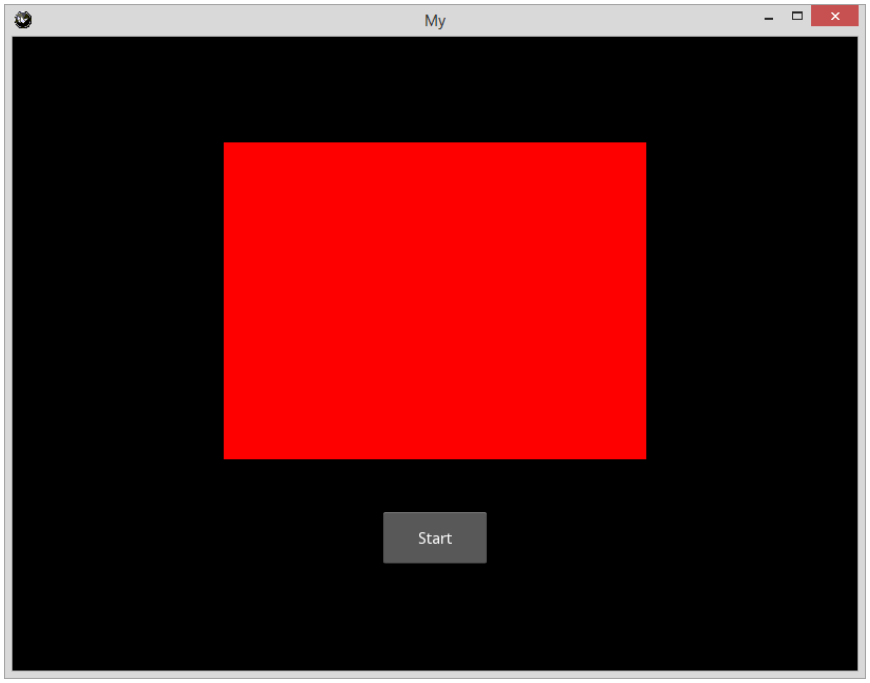

Abbildung 3: einfache Kivy Applikation

2.4 Gestenerkennung

Die Umsetzung der Gestenerkennung in Kivy folgt der Implementierung von Oleg Dopertchouk, Recognition of Handwritten Gestures. [Dop04]
Grundsätzlich wird zunächst eine Gestendatenbank aufgebaut, die alle zu erkennenden Gesten enthält. Das Erstellen beziehungsweise Erkennen einer Geste geschieht wie folgt:
Gesten werden aus Sequenzen von 2D Punkten, sogenannten *Strokes* aufgebaut. Um das Absetzen des Fingers zu ermöglichen, kann eine Geste aus mehreren Strokes bestehen. Ein Stroke wird über einen dreistufigen Prozess normalisiert. Zunächst wird die Eingabe auf ein einheitliches Maß skaliert. Anschließend werden die Punkte der Eingabe in einen gleichmäßigen Abstand gesetzt, um die Erkennung unabhängig von der Geschwindigkeit der Bewegung zu machen. Abschließend werden die Punkte um einen gemeinsamen Ursprung zentriert. Jeder Stroke wird als 2·N-dimensionaler Vektor interpretiert. Die Ähnlichkeit von Eingabe und gespeicherter Geste in der Datenbank wird über das Skalarprodukt der beiden Vektoren bestimmt. Sind Eingabe und Vergleichswert vollständig identisch ergibt sich ein Übereinstimmungswert von 1.0. Dieser Wert nimmt ab je unterschiedlicher beide Vektoren sind. Soll eine Geste zu einer Eingabe gefunden werden, so werden alle Gesten in der Datenbank überprüft und diejenige mit dem höchsten Übereinstimmungswert wird als Ausgabe zurückgegeben. Liegen alle Übereinstimmungswerte unterhalb eines Schwellwertes wird keine Geste zurückgegeben. [Dop04]

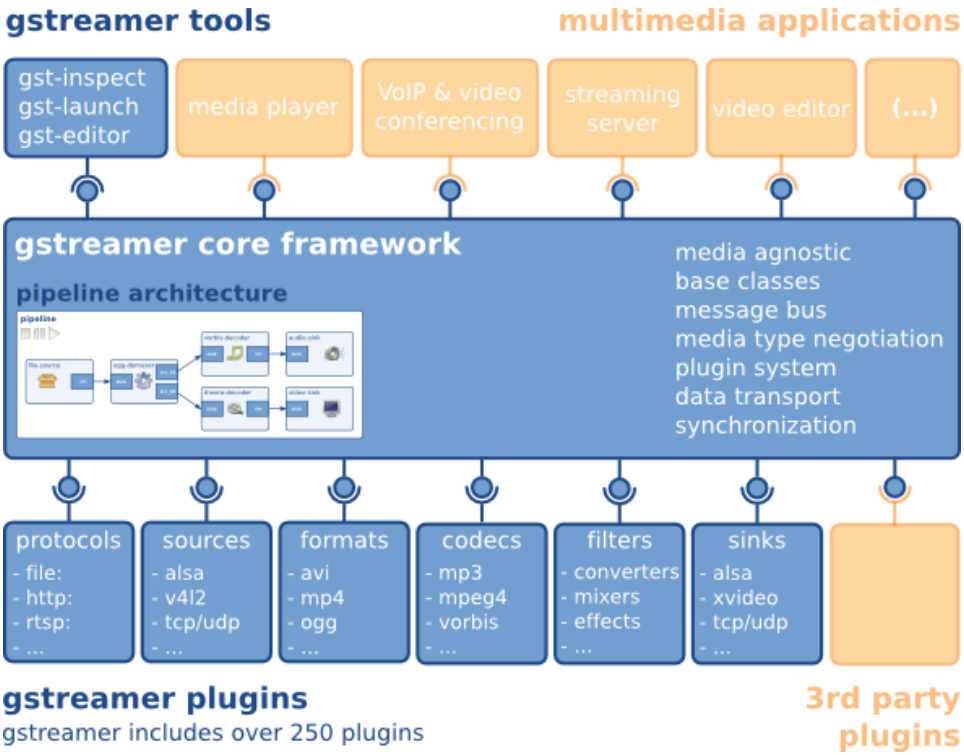

Abbildung 4: GStreamer Architektur [Tay]

2.5 GStreamer

GStreamer ist ein freies Framework zur Erstellung von Multimedia-Anwendungen. Es können beliebige Streaming-Anwendungen geschrieben werden, in denen sowohl Audio- als auch Videodaten verwaltet werden können. Die Architektur von GStreamer ist in Abbildung 4 zu sehen.
GStreamer basiert auf einer Plugin-Struktur, dadurch ist es einfach mit eigenen Elementen und Funktionen erweiterbar. Es gibt drei Hauptarten von Elementen mit spezifischen Funktionen. Zur Daten-Eingabe sogenannte *Source*-Elemente, zur Weiterverarbeitung der Daten beispielsweise Filter-, Codier- und Decodier-Elemente und sogenannte *Sink*-Elemente, die für die Ausgabe zuständig sind. Die verschiedenen Elemente lassen sich zu sogenannten *Pipelines* anordnen und verbinden. Durch sie wird der Datenfluss bestimmt und kontrolliert. Es wird bereits eine große Anzahl an fertigen Elementen bereitgestellt, bei Bedarf kann jedoch jederzeit ein eigenes Plugin erstellt und eingebunden werden.
Elemente haben *Pads*, durch die die Daten fließen. Sie sind mit Stecker und Buchse vergleichbar und es kann über *Caps* spezifiziert werden welche Daten angenommen, beziehungsweise weitergeleitet werden können. In Abbildung 5 sieht man eine einfache Pipeline mit ihren Elementen.
Ein Hauptaugenmerk von GStreamer liegt auf einer hohen Leistungsfähigkeit bei

niedriger Latenz. Bei dem Datenfluss durch die Pipeline entsteht nur ein minimaler Overhead, da die Weiterleitung bei typischen Pipelines nur aus einer Pointer Dereferenzierung besteht. Außerdem können direkte Zugriffe auf den Zielspeicher durchgeführt werden. Plugins werden erst zur Laufzeit geladen. Durch spezialisierte Plugins können Hardware-beschleunigte Funktionalitäten eingebaut werden. [Tay]

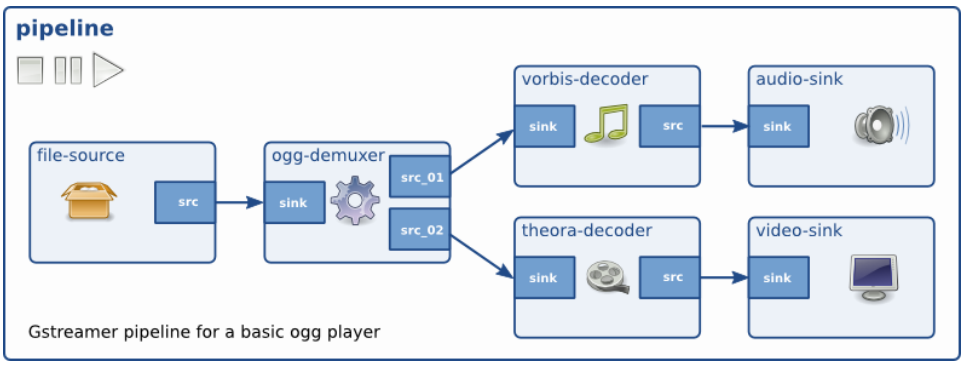

Abbildung 5: Pipeline für einen einfachen ogg-Player [Tay]

2.6 Screen Space Ambient Occlusion

Unter *Screen Space Ambient Occlusion (SSAO)* versteht man ein Beleuchtungsmodell, das genutzt wird, um indirekte/globale Beleuchtung zu simulieren. Es gilt nur für diffuse Oberflächen und nimmt ein einheitliches Umgebungslicht an. Einzelne Lichtquellen werden nicht berücksichtigt. Es is allgemein definiert als das Integral über die Verdeckung, die von den Punkten einer Hemisphäre Omega mit einem gegebenen Radius R mit dem aktuellen Oberflächenpunkte P als Mittelpunkt und orientiert an seiner Normalen n, beigesteuert wird (siehe Abbildung 6).

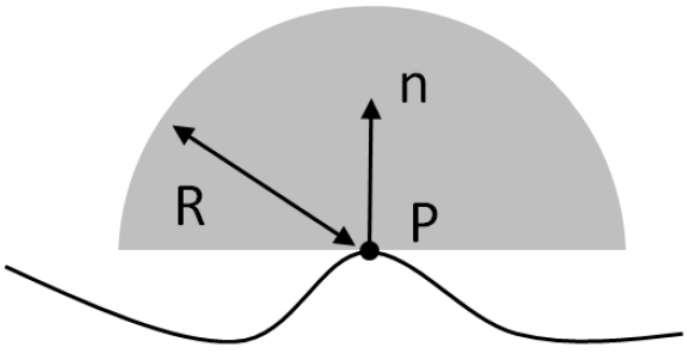

Abbildung 6: Hemisphäre für den betrachteten Punkt auf der Oberfläche [BS08]

Alle Punkte innerhalb dieser Hemisphäre werden als mögliche Okkluder betrachtet, die zur Verdeckung des betrachteten Punktes beitragen. Punkte außerhalb der Hemisphäre werden ignoriert. [BS08]
Screen Space Ambient Occlusion ist eine Vereinfachung des AO-Verfahrens, das die Integration über eine Kugeloberfläche umgeht. Es wertet die Nachbarschaft lediglich punktuell aus und nähert den AO-Wert somit an. Durch die schnellere Berechnung ist SSAO echtzeitfähig und funktioniert auch bei dynamischen Szenen. Die Berechnung ist unabhängig von der Komplexität der Szene, da im zweidimensionalen Screen Space gerechnet wird. Es ist keine Vorverarbeitung nötig und die Berechnungen können komplett auf der GPU ausgeführt werden. Probleme entstehen durch fehlendes Wissen über nicht sichtbare Geometrie. Dies wird als fehlende Verschattung von verdeckter Geometrie, Fehlern am Bildschirmrand, und bei transparenten Oberflächen deutlich. [Kaj09]

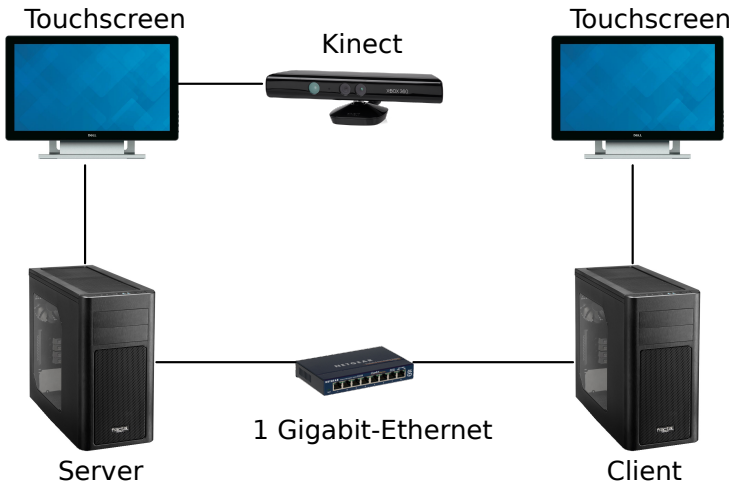

Abbildung 7: Aufbau des Testsystems

3 Realisierung

3.1 Konzeption

Es soll ein System simuliert werden, mit dem mehrere Benutzer an voneinander entfernten Geräten miteinander arbeiten können. Mit dem System sollen die im Kapitel 1.2 formulierten Hypothesen evaluiert werden. Dazu werden zwei Computer benötigt, die über ein Netzwerk miteinander verbunden sind. Diese müssen über eine Kamera zur Aufzeichnung der Eingabehand und Touchscreen-Monitore verfügen. Es sollen Situationen simuliert werden, in denen ein Benutzer über Informationen verfügt, die der andere Benutzer benötigt. [MIL99] Auf Basis dessen

Abbildung 8: Befestigung der Kinect am Monitor

bietet sich eine Aufteilung in Erklärenden und Ausführenden an. Für die Evaluierung der Hypothesen reicht es aus die Übertragung der Hand von nur einem der Arbeitsplätze zu verwenden. Daher ist die Verwendung von nur einem Kinect Sensor, am Arbeitsplatz des Erklärenden, ausreichend. Daraus hat sich das Testsystem wie in Abbildung 7 ergeben. Außerdem muss ein Weg gefunden werden die Kinect zu fixieren. Dabei muss der Bildschirm vollständig zu sehen sein. Außerdem ist zu berücksichtigen, dass der Kinect Sensor erst ab etwa 40cm brauchbare Tiefenwerte liefert. Auch die Verdeckung durch den Körper des Anwenders ist zu berücksichtigen. Kopf oder Rücken können die freie Sicht auf die Hand verhindern, sodass keine Tiefenwerte für diese geliefert werden können. Diese Anforderungen haben den Aufbau in Abbildung 8 ergeben.

3.1.1 Testanwendung

Um die Hypothesen H1.1 - H1.3 evaluieren zu können, müssen Tests zu verschiedenen Szenarien und Aufgabentypen erstellt werden. Aus der zuvor erstellten Rollenaufteilung ergibt sich die Anforderung von unterschiedlichen Testanwendungen für Erklärenden und Ausführenden. Dabei sollen neben einfachen Aufgaben mit grundlegenden Bedienkonzepten von Touchscreen-Eingaben auch komplexere Anwendungen getestet werden. Zum einen *Natural User Interface* Elemente, bei

denen Objekte direkt manipuliert werden. Zum anderen herkömmliche *Graphical User Interface* Bestandteile wie Menüs, Schieberegler und Schaltflächen. [KASA08] Außerdem weitere Aufgaben, in denen Informationen vermittelt werden müssen, die sich nicht auf die Bedienung beziehen, sondern rein auf Informationen die der Bildschirminhalt bereitstellt. Die Bedienung soll mittels verschiedener Multi-Touch-Gesten möglich sein.

3.1.2 Handvisualisierung

Um eine Visualisierung der Hand zu erstellen, soll die Eingabehand aufgenommen und freigestellt werden. Es gibt verschiedene Möglichkeiten Hände in Bildern zu finden. So ist es möglich in Farbbildern nach Pixeln zu suchen, die einen bestimmten Farbton haben. [CAHS06] Oder diese über das Herausrechnen der Bewegung vom statischen Hintergrund zu trennen. [Pic04] In dem hier entwickelten System besteht der Hintergrund nur aus dem Monitor. Dieser bildet eine ebene Fläche und ist somit klar durch die Entfernung zur Kamera von der Hand unterscheidbar. Daher bietet sich ein Tiefenbild zur Weiterverarbeitung an. In dem Testsystem soll daher als bildgebendes Element *Kinect for Windows* verwendet werden. Zur Bearbeitung der erhaltenen Tiefentextur sollen GLSL Shader verwendet werden. Dadurch wird eine schnellere Berechnung gewährleistet, da die Rechenleistung der GPU genutzt wird.

3.2 Anforderungsliste

Aus den Hypothesen und den Überlegungen der Konzeption hat sich folgende Anforderungsliste ergeben.

Testanwendung allgemein:

1. Es muss eine Testanwendung für den Ausführenden geben.
2. Es muss eine Testanwendung für den Erklärenden geben.
3. Die Testanwendung muss verschiedene Tests bereitstellen.
4. Die Tests müssen in festgelegter Reihenfolge durchlaufen werden.
5. Es müssen Multi-Touch-Gesten zur Bedienung genutzt werden können.
6. Die Benutzeroberfläche soll schlicht und gut erkennbar sein.
7. Alle Texte müssen gut lesbar sein.

Testanwendung Ausführender:

8. Ein Test beginnt erst, wenn der Erklärende bestätigt, dass er bereit ist.
9. Ein Test beginnt erst, wenn der Ausführende bestätigt, dass er bereit ist.

10. Es muss die Zeit gemessen werden, die für einen Test benötigt wird.
11. Alle Testzeiten müssen in einer Textdatei gespeichert werden.
12. Der Ausführende soll Hinweise auf erfüllte Teilaufgaben erhalten.
13. Der Ausführende darf nicht zum nächsten Test gelangen, ohne das die Aufgabe abgeschlossen ist.
14. Der Ausführende muss zum nächsten Test gelangen wenn die Aufgabe abgeschlossen ist.
15. Alle Tests müssen abschließbar sein.
16. Es muss die Hand des Erklärenden eingeblendet werden können.
    (siehe Darstellung der *Hand*)

Testanwendung Erklärender:

17. Der Erklärende muss zu jedem Zeitpunkt die aktuelle Aufgabenstellung einsehen können.
18. Der Erklärende soll über einen Button direkt zum nächsten Test gelangen, auch ohne die Aufgabe selbst abzuschließen.
19. Der Erklärende soll innerhalb des Tests weitere Hinweise zum Lösen der Aufgabe erhalten.
20. Die Touchkoordinaten des Erklärenden müssen übertragen werden.

Kinect/Streaming:

21. Das Kinect-Tiefenbild muss über das Netzwerk übertragen werden.
22. Die Übertragung muss stabil sein.
23. Die Übertragung soll eine geringe Latenz aufweisen.

Darstellung der *Hand*:

24. Die Position der dargestellten Hand muss mit der der realen Eingabehand übereinstimmen.
25. Es müssen die Touchkoordinaten des Erklärenden visualisiert werden.
26. Die Position und Ausrichtung der *Hand* muss erkennbar sein.
27. Die *Hand* soll keine Bildschirminhalte verdecken.
28. *Hände* unterschiedlicher Anwender sollen unterschiedlich dargestellt werden können.
29. Die *Hand* soll klar vom Hintergrund unterscheidbar sein.

3.3 Verwendete Technologien

Die Anwendung wurde auf Windows 8.1 64-Bit Systemen entwickelt, getestet und evaluiert. Als Entwicklungsumgebung kam Microsoft Visual Studio 2013 Ultimate in der Version 12.0.21005.1 REL zum Einsatz, zusammen mit Python Tools for Visual Studio 2.0.11016.00, um weitere Funktionen für die Entwicklung in Python zu erhalten.
Als primäre Programmiersprache wurde Python 2.7 verwendet, welches auch die Grundlage für das Kivy Framework bildet. Kivy 1.7.2 w32 bildet das grundlegende Framework für die entwickelte Anwendung, da es eine solide Unterstützung für Touchscreen-Eingabegeräte bietet und bereits GStreamer 0.10 enthält, welches für das Streaming des Kinect-Bildes benutzt wird. Die Übertragung der Touchkoordinaten wird über das Twisted Framework 14.0.0 win32 abgewickelt.
Um Zugriff auf den Tiefensensor der Microsoft Kinect zu erhalten, wurde das OpenNI 2 SDK 2.2.0.33 x86 verwendet.

3.4 Geräte

Das gesamte System umfasst 2 Computer mit Intel i5 3,4 GHz Quadcore Prozessoren, 8GB DDR3 RAM und AMD HD6950 Grafikkarte. Bei den verwendeten Monitoren handelt es sich um Dell P2314T, kapazitive Touchscreen-Monitore mit 23 Zoll und FullHD Auflösung von 1920x1080 Pixeln. Am ersten Computer, dem Arbeitsplatz des Erklärenden, ist die Microsoft Kinect for Windows angeschlossen. Dieser Computer streamt die Daten des Kinect Tiefensensors an den zweiten PC, an dem der Ausführende arbeitet. Die Computer der Arbeitsplätze sind über eine 1 Gigabit-Ethernet Verbindung verbunden (siehe Abbildung 7). Für die Befestigung des Kinect Sensors wurde eine Halterung gebaut. Diese ist in Abbildung 8 zu sehen. Sie zentriert den Sensor in 60cm Entfernung von der Bildschirmoberfläche.

3.5 Aufbau der Anwendung

3.5.1 Die Testanwendung

In Abbildung 9 ist ein Überblick über den Aufbau der Testanwendung zu sehen. Das Basiselement des Kivy Programms ist die EvalTouchApp, diese bildet den Einstiegspunkt in den Programmablauf. Die Klasse TouchTest ist das Basis Widget, in dem die gesamte Anwendung gezeigt wird. Das Element *HandKomposition* wird im Abschnitt 3.5.3 genauer beschrieben.

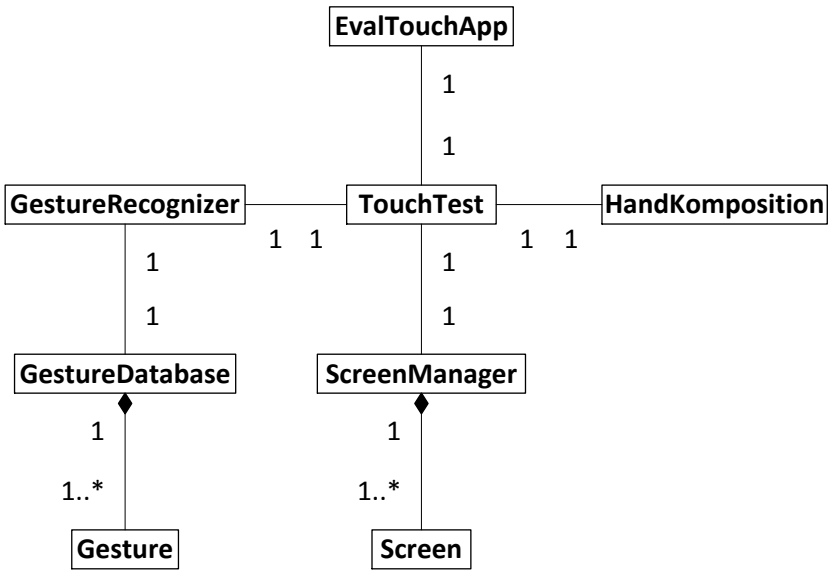

Abbildung 9: Überblick über den Aufbau der Testanwendung

Widget: TouchTest
Dieses Widget bildet das Eltern Widget für die weiteren Hauptbestandteile der Anwendung, den *Screenmanager*, die *Gestenerkennung* und die *HandKomposition*. Hier werden die Touch Events verarbeitet, die auf allen Screens denselben Effekt zeigen. Es werden zusammengehörige Strokes aufgezeichnet und an den GestureRecognizer weitergeleitet.

Screenmanager
Nach Punkt 3 der Anforderungsliste sollen verschiedene Tests bereitgestellt werden. Hierfür wurde ein *ScreenManager* aus dem *kivy.uix.screenmanager Modul* eingesetzt. Dieser ermöglicht es für jeden Test einen eigenen *Screen* zu erstellen. Ein Screen stellt verschiedene Events bereit, die sowohl beim Betreten als auch beim Verlassen des Screens ausgelöst werden. Diese werden benutzt, um dem Benutzer zu Beginn jedes Tests ein Pop-up-Fenster mit den nötigen Information zu zeigen. An diese Pop-up-Fenster ist ebenfalls die Zeitmessung gekoppelt, um Anforderung 8. und 9. zu entsprechen. Dem Erklärenden steht außerdem der „Anweisungen“-Button in der unteren linken Bildschirmecke zur Verfürgung. Dieser zeigt ihm zu jederzeit noch einmal das Pop-up-Fenster mit den Anweisungen. (siehe Punkt 17. der Anforderungsliste) Die Screens und ihr Inhalt werden in 3.5.2 vorgestellt.

Gestenerkennung
Die in Punkt 5 der Anforderungsliste geforderte Gesten-Unterstützung wird über das *kivy.gesture* Modul umgesetzt.
Um die Erkennung von Multi-Touch-Gesten zu ermöglichen, wurde der Algorithmus erweitert. Es werden die Startzeiten zweier *TouchEvents* verglichen. Wenn diese innerhalb von 200ms gestartet wurden, wird die nachfolgende Geste als Multi-Touch-Geste assoziiert und nicht als zwei unabhängige Gesten. Dies ermöglicht beispielsweise einen „2-Finger-Swipe“ in eine beliebige Richtung zu erkennen. Aber auch kombinierte Gesten, wie das Halten eines Objektes, während mit der anderen Hand ein Haken gezogen wird. Viele der hier verwendeten Gesten finden sich in [MWW10], andere sind selbst erstellt, um die Probanden vor unbekannte Aufgaben zu stellen.
Wird eine Geste gezeichnet, veranschaulicht eine farbige Spur auf dem Bildschirm die bisher zurückgelegte Strecke. So ist es einfacher zu erkennen warum eine Geste gegebenenfalls nicht erkannt wurde. Beispielsweise weil, ein Kreis nicht ausreichend rund gezeichnet wurde. Die Spur ist auf maximal 500 2D Punkte beschränkt, um bei der Nutzung von mehreren Fingern keine Verlangsamung der Gestenerkennung zu verursachen. Einige der in der Testanwendung verwendeten Gesten sind in Abbildung 10 zu sehen. Die abgebildeten Spuren sind Screenshots der angesprochen Visualisierung.

Abbildung 10: Einige der unterstützten Gesten, von links: 2-Finger-Swipe, Haken, Kreis und Cross

3.5.2 Die Tests

Die Testanwendung umfasst 13 verschiedene Tests. Diese gliedern sich grob in drei Aufgabentypen.

Aufgaben Typ 1: einfache Anwendungen
Der erste Aufgabentyp umfasst einfache Aufgaben, die mit grundlegendem Verständnis von Touch-Eingabemöglichkeiten abgeschlossen werden können.
Diese Gruppe umfasst die nachfolgenden Tests:

Test Ecken

Dieser erste Test ist einfach gestaltet, um den Probanden an die Benutzung des Touchscreens heranzuführen.
Die Aufgabe besteht darin, weiße Quadrate in den Ecken des Displays anzutippen. Nachdem das erste Rechteck angetippt wurde, verschwindet es und das nächste erscheint. Dies geschieht vier mal bis jede Ecke einmal angetippt wurde. Der Aufbau ist in Abbildung 11 zu sehen.

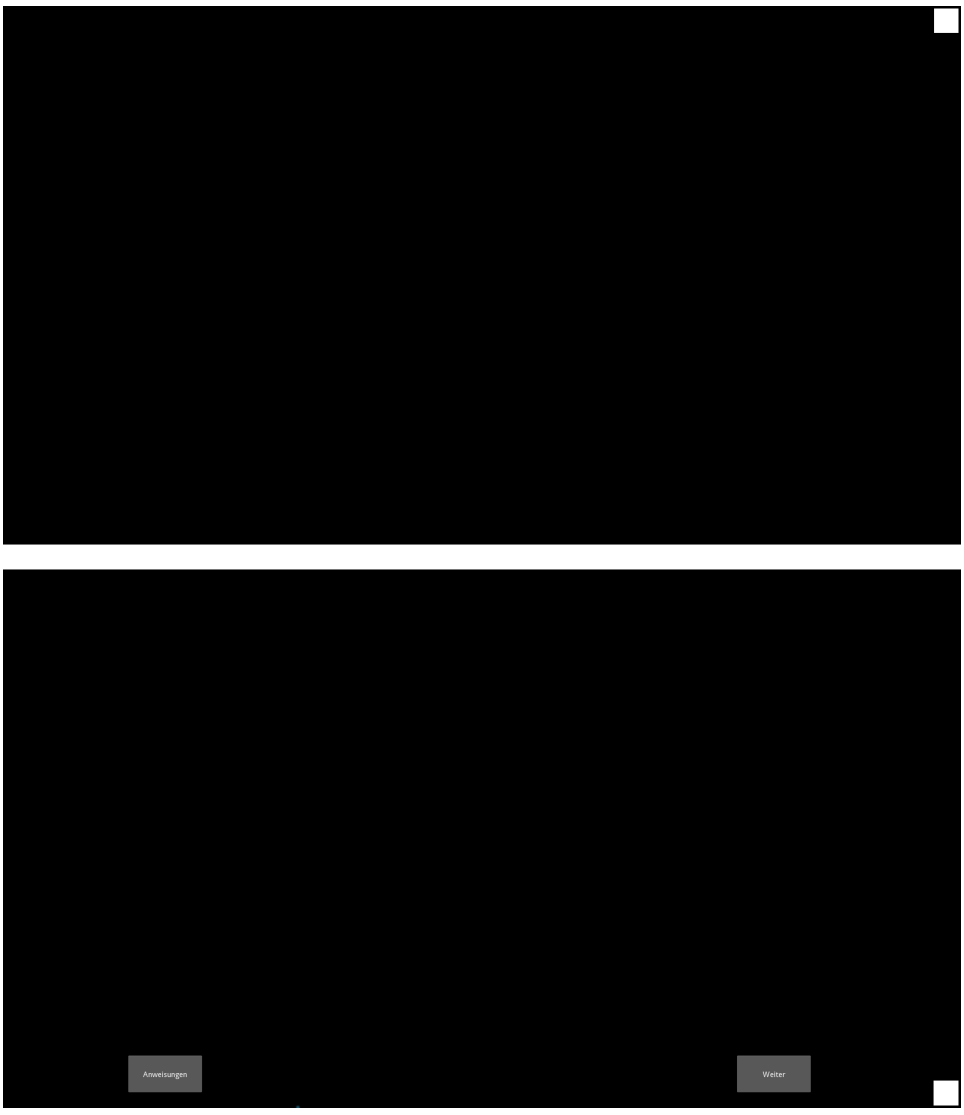

Abbildung 11: Test Ecken, oben Ausführender, unten Erklärender

Test Transformation

Das weiße Quadrat muss zunächst in den blauen Rahmen verschoben werden. Dann mit zwei Fingern um $90°$ im Uhrzeigersinn rotiert werden und anschließend mit zwei Finger auf die Größe des Rahmens skaliert werden. Die hier erforderlichen Gesten zur Manipulation des Testobjektes sind von verschiedenen Touch-Eingabegeräten bekannt. Ziel ist es hier zu überprüfen, ob bei der Verwendung von bereits bekannten Gesten ein Mehrwert durch die Einblendung erreicht werden kann, oder ob eine mündliche Beschreibung zu gleichen Ergebnissen führt. Der Aufbau ist in Abbildung 12 zu sehen.

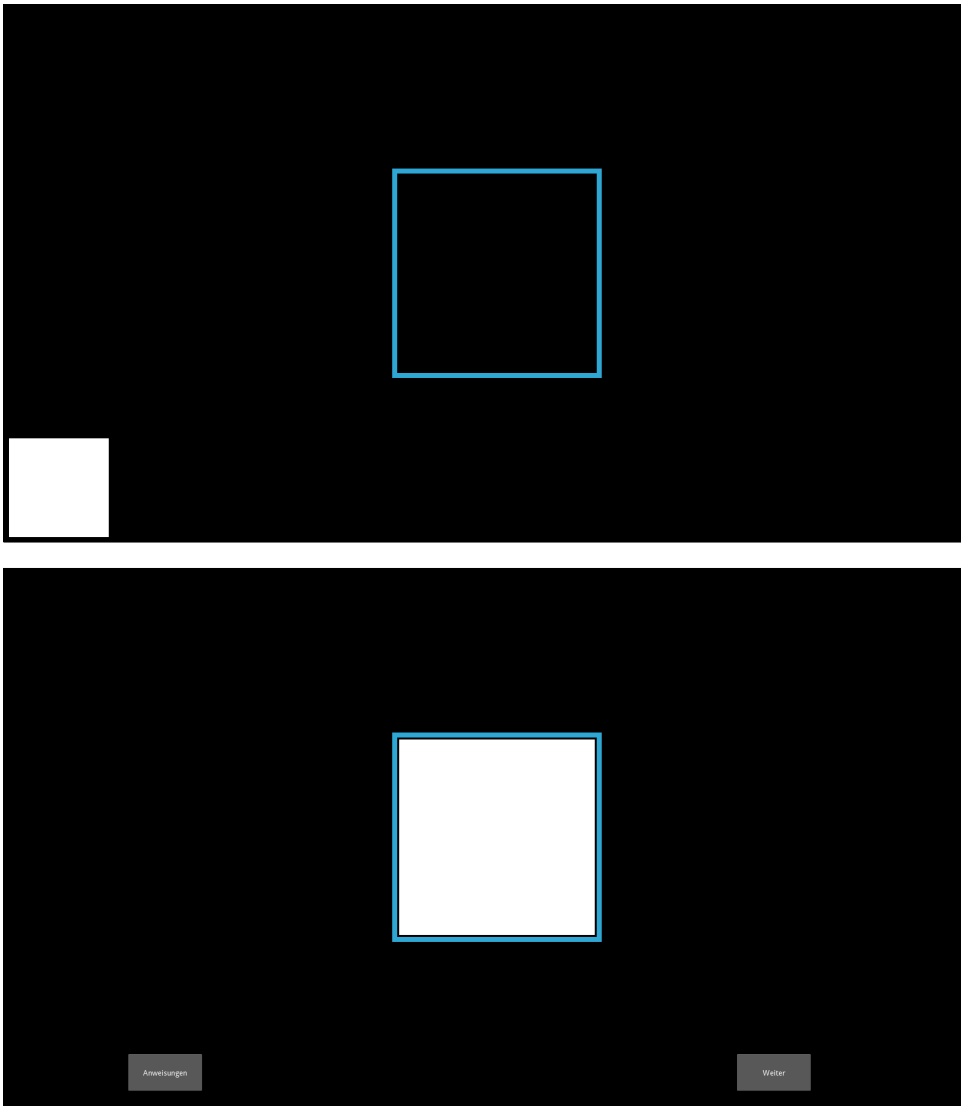

Abbildung 12: Test Transformation, oben Ausführender, unten Erklärender

Test Entsperrmuster

Dieser Test basiert auf einem Modell zur Entsperrung von Smartphones, die bei vielen Geräten Verwendung findet. Dabei müssen neun Punkte in einer festen Reihenfolge verbunden werden, um das Gerät zu entsperren, beziehungsweise diesen Test erfolgreich zu beenden. Obwohl das Konzept der Bedienung bekannt ist, kann die Aufgabe nicht ohne die Informationen des Erklärenden gelöst werden. Dem Erklärenden wird die korrekte Reihenfolge dauerhaft angezeigt, während der Ausführende nur die neun Quadrate sieht. Der Aufbau ist in Abbildung 13 zu sehen.

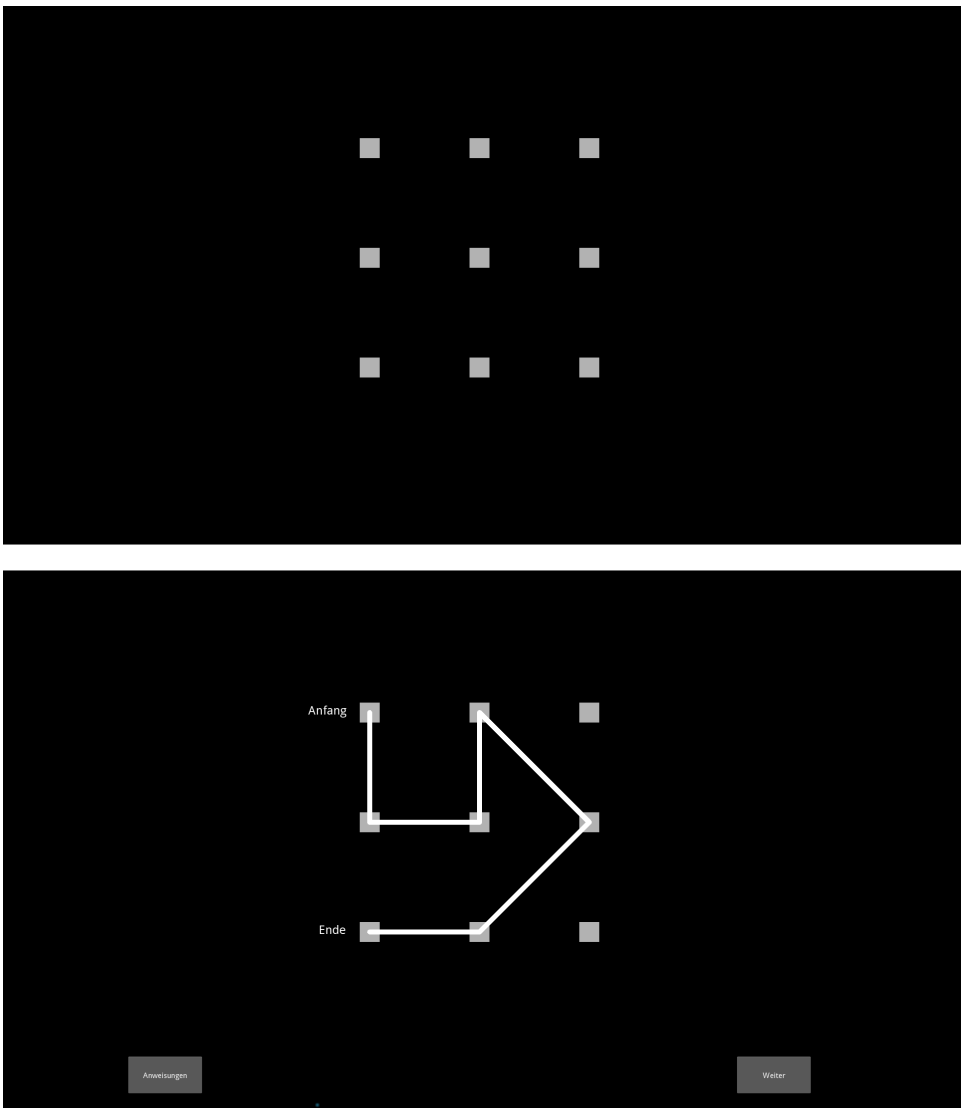

Abbildung 13: Test Entsperrmuster, oben Ausführender, unten Erklärender

Test Formen

In diesem Test müssen geometrische Formen in ihre entsprechenden Umrandungen verschoben werden. Bevor die Objekte verschoben werden können müssen sie ausgewählt werden. Dies geschieht durch das Zeichnen eines Kreises um das Objekt. Nachdem ein Objekt ausgewählt wurde kann es angetippt und verschoben werden. Die Geste zur Auswahl eines Objektes unterscheiden sich von einfachem Antippen. Sie muss daher vom Erklärenden beschrieben werden. Der Aufbau ist in Abbildung 14 zu sehen.

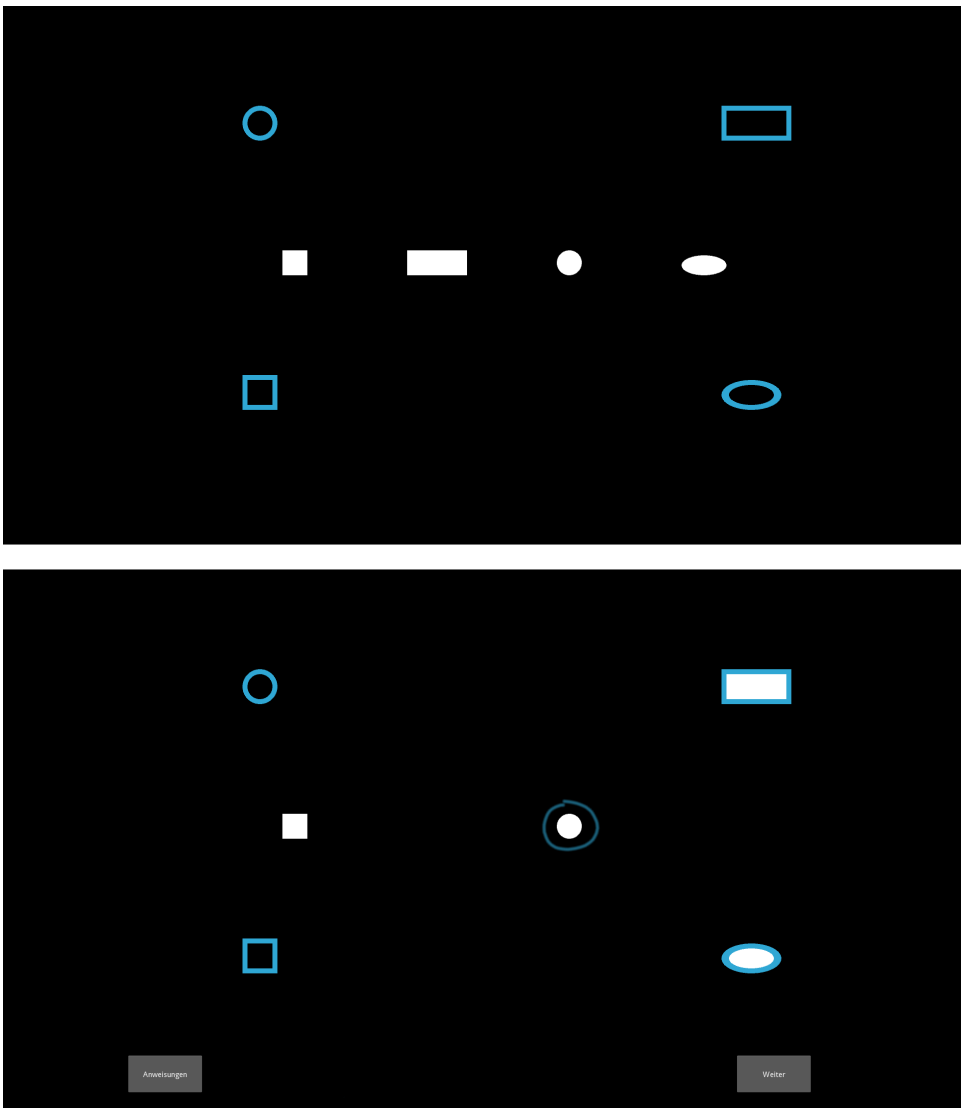

Abbildung 14: Test Formen, oben Ausführender, unten Erklärender

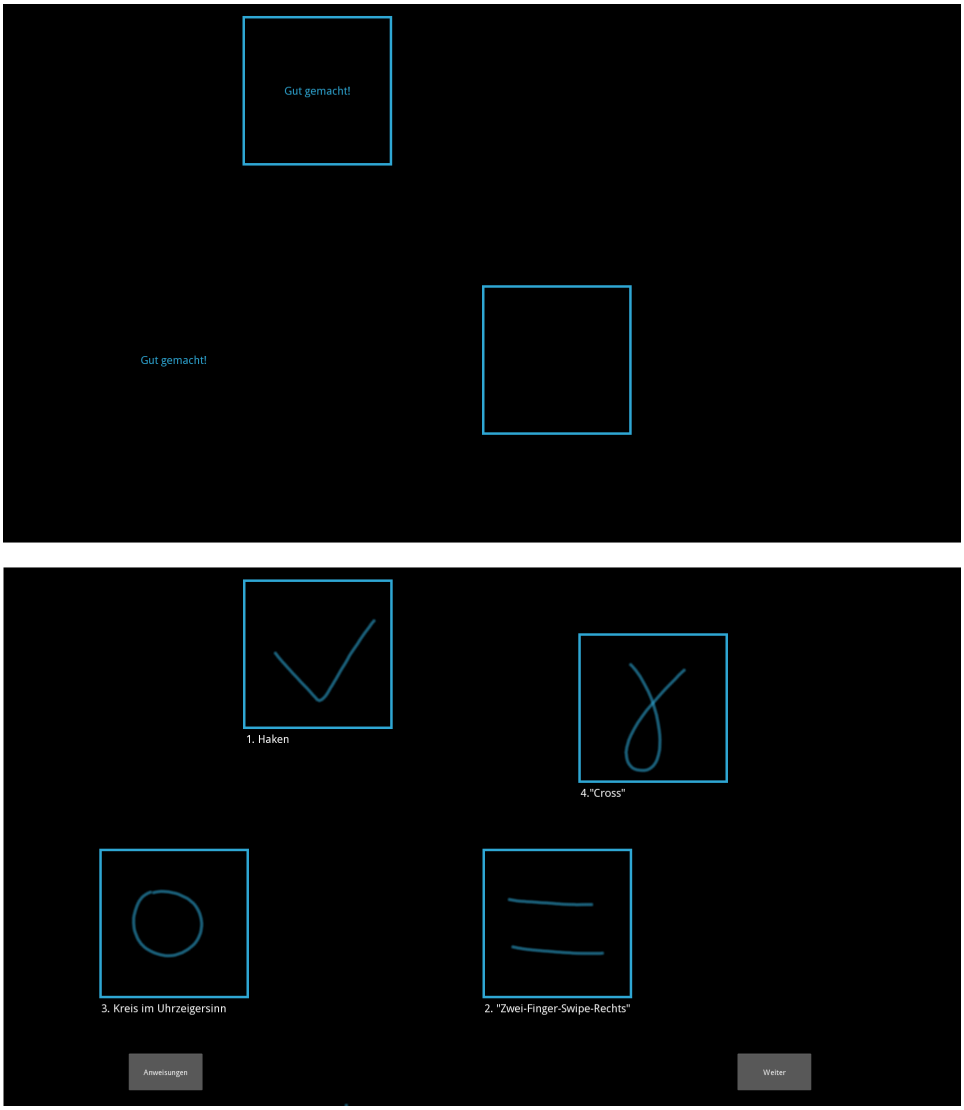

Abbildung 15: Test Gesten, oben Ausführender, unten Erklärender

Test Gesten

Zu Beginn sieht der Ausführende zwei blaue Rahmen. Dem Erklärenden werden noch zwei weitere Rahmen angezeigt, sowie die Geste, die in diesem Bereich ausgeführt werden muss.
Die Aufgabe besteht darin, vier verschiedene Gesten in den markierten Bereichen auszuführen. Wurde eine Geste im korrekten Bereich ausgeführt, wird dieser mit 'Gut gemacht!' gekennzeichnet. Dieser Test beinhaltet zwei Schwierigkeiten. Zum einen muss der jeweilige Verlauf der Gesten erklärt werden. Bei der Ausführung des Kreises ist beispielsweise zu beachten, dass dieser im Uhrzeigersinn verlaufen muss und der Start bei '12-Uhr' liegt. Zum anderen sind nur zwei der vier Ziel-

bereiche für den Ausführenden sichtbar. Der Erklärende muss eine effektive Möglichkeit finden seinem Gegenüber die Position dieser Zielbereiche zu vermitteln. Der Aufbau ist in Abbildung 15 zu sehen.

Aufgaben Typ 2: komplexere Anwendungen
Die zweite Gruppe von Aufgaben erfordert es komplexere Abläufe und Bedienkonzepte zu vermitteln. Zwei der Aufgaben sind an reale Anwendungen angelehnt während die anderen 2 keine bekannten Vorlagen haben.

Test Regler
Dieser Test besteht aus drei Reglern, die unterschiedliche Einflüsse aufeinander haben. Der erste Regler begrenzt den zweiten Regler in seinem Maximalwert. Der zweite Regler kann nie einen höheren Wert annehmen als der erste. Ziel ist es, durch geschicktes Kombinieren der Regler einen bestimmten Wert im Ergebnisfeld einzutragen. Das Verhalten der Regler folgt dabei keinem, das man direkt aus einer alltäglichen Anwendung kennt. Stattdessen ist der Test wie ein Rätsel gestaltet, um zu überprüfen, ob auch nicht direkt anschauliche Problemstellungen von der Einblendung der Hand des Erklärenden profitieren können. Der Erklärende hat dabei die Aufgabe die genaue Funktionsweise der Regler zu vermitteln. Er soll jedoch nicht einfach die korrekte Stellung der Regler verraten, die nötig ist um den Test abzuschließen. Der Aufbau ist in Abbildung 16 zu sehen.

Test Texteditor
Dieser Test zeigt einen einfachen Texteditor. In diesem muss der Proband zunächst über das Menü ein neues Dokument anlegen. Dann kann durch Doppeltippen der rechte Text markiert werden. Mit einem Finger wird das Zieltextfeld gehalten und mit zwei Fingern vom rechten in das linke Textfeld gewischt. Dies kopiert den markierten Text. Abschließend muss das Dokument über das Menü gespeichert werden. Die grundlegende Funktionsweise eines Texteditors ist geläufig und die Menüaufteilung folgt bekannten Mustern. Der Test simuliert hier die Vermittlung eines einfachen Ablaufs zur Bedienung eines Computerprogramms. Neben den bekannten Funktionsweisen eines Textverarbeitungsprogrammes enthält der Test einige Besonderheiten, wie sie auf einem Touchscreen-Gerät zu finden sein könnten. Der Aufbau ist in Abbildung 17 zu sehen.

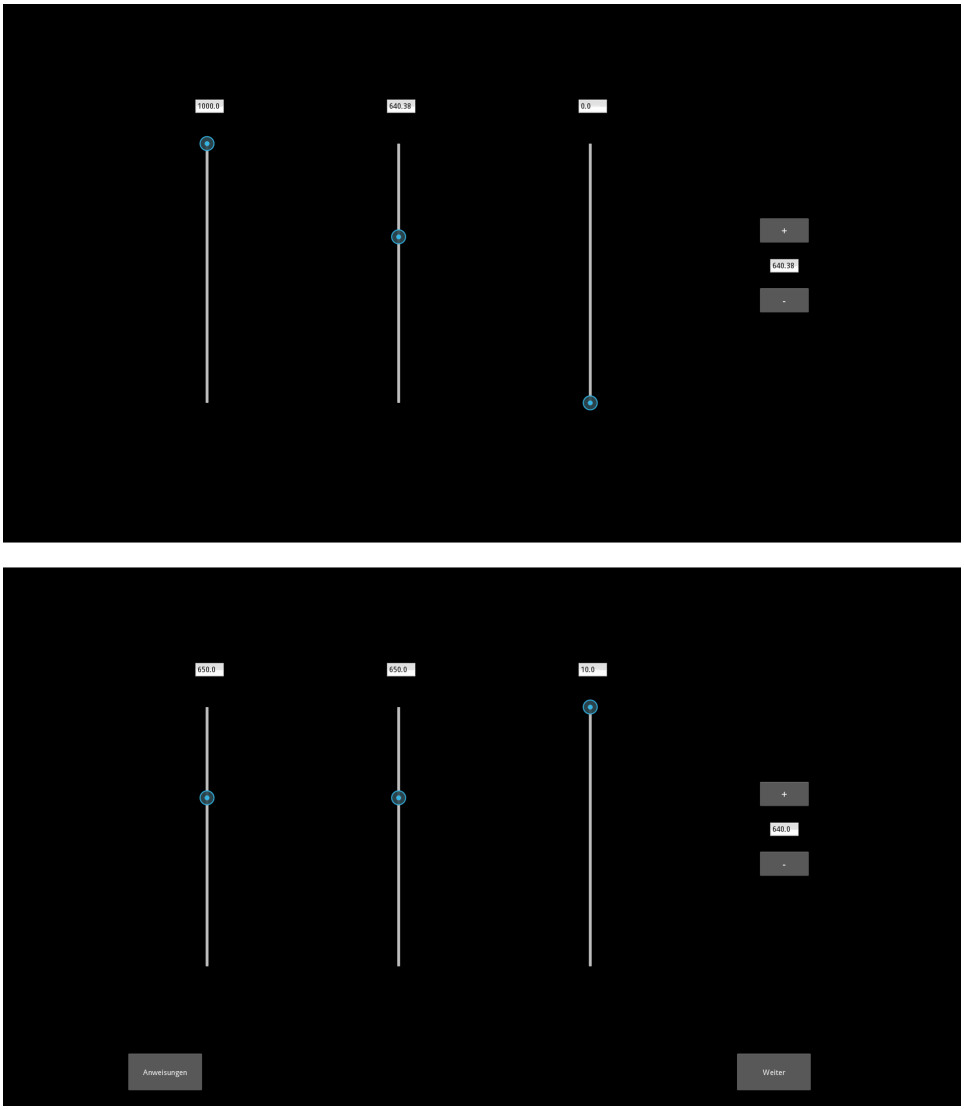

Abbildung 16: Test Regler, oben Ausführender, unten Erklärender

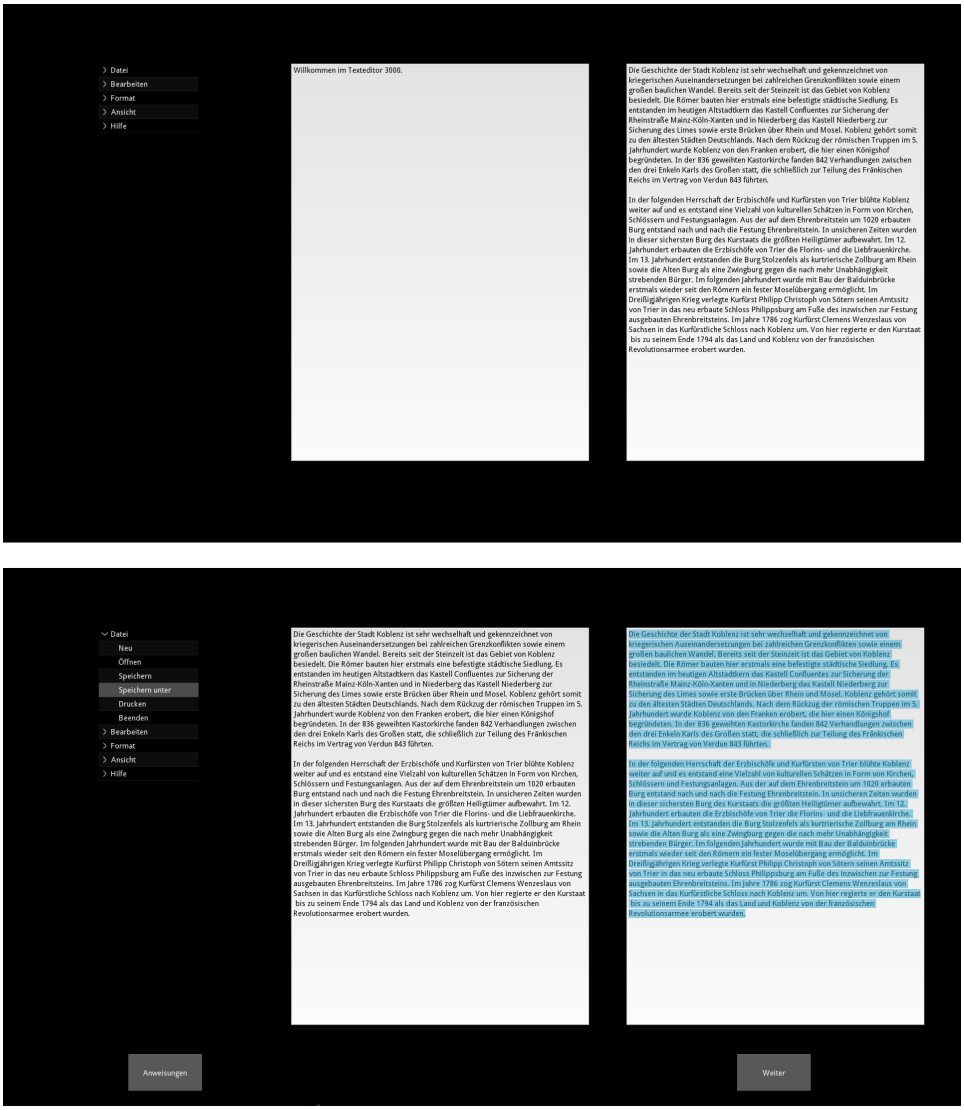

Abbildung 17: Test Texteditor, oben Ausführender, unten Erklärender

Test Farben

In diesem Test muss eine Farbmisch-Anwendung bedient werden. Es stehen Bedienelemente für RGBA und HSV Farbmodelle, sowie ein Farbrad zur Verfügung. Der Proband muss die drei weißen Quadrate auf der rechten Seite in bestimmten Farben einfärben. Grundsätzlich beschränkt sich die Bedienung auf bekannte Mechaniken, wie Regler. Einfaches Tippen auf die Zielfelder, weist diesen die aktuell eingestellte Farbe zu. Die Herausforderung besteht darin, die genaue Funktionsweise der einzelnen Regler zu vermitteln. Die richtige Lösung kann kaum durch Ausprobieren gefunden werden, da die gesuchten Farben nur bei präziser Stellung der einzelnen Regler zueinander erhalten werden. Der Aufbau ist in Abbildung 18 zu sehen.

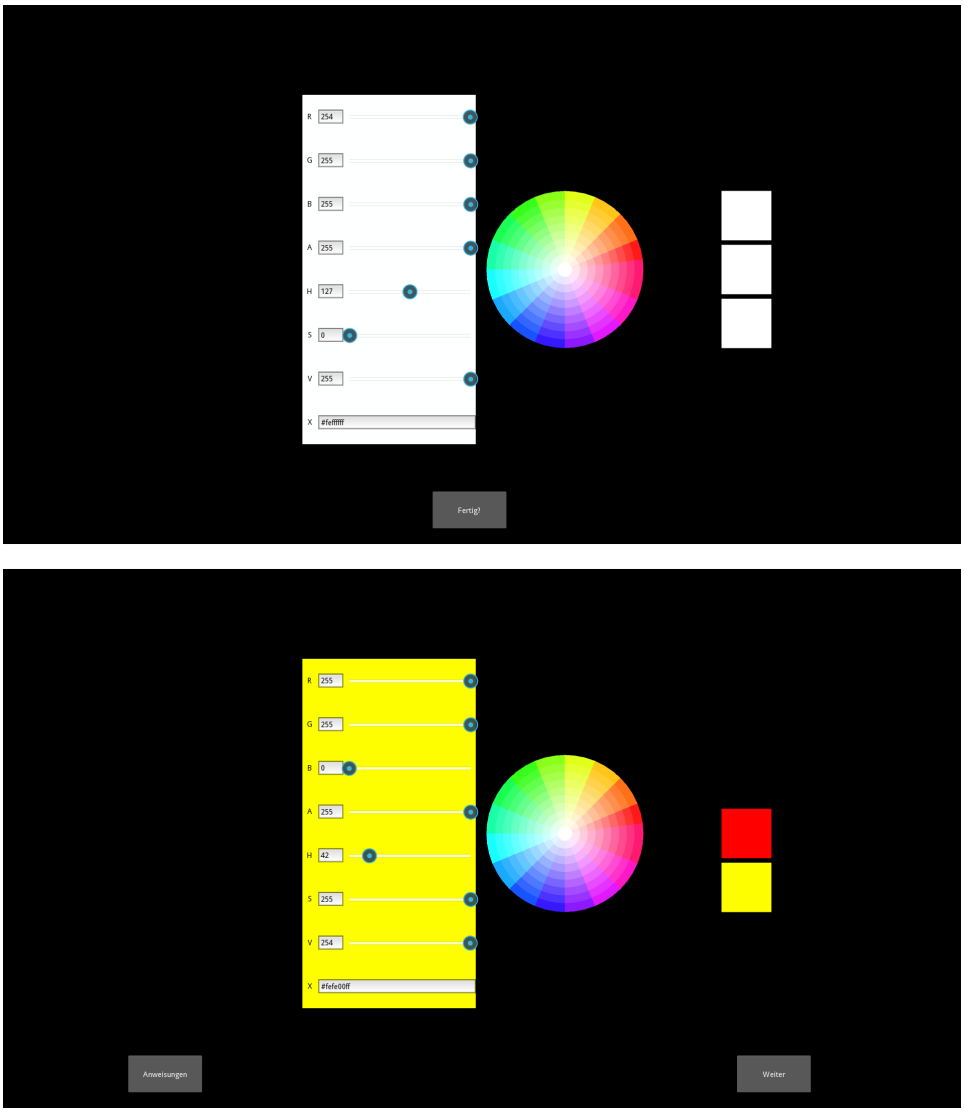

Abbildung 18: Test Farben, oben Ausführender, unten Erklärender

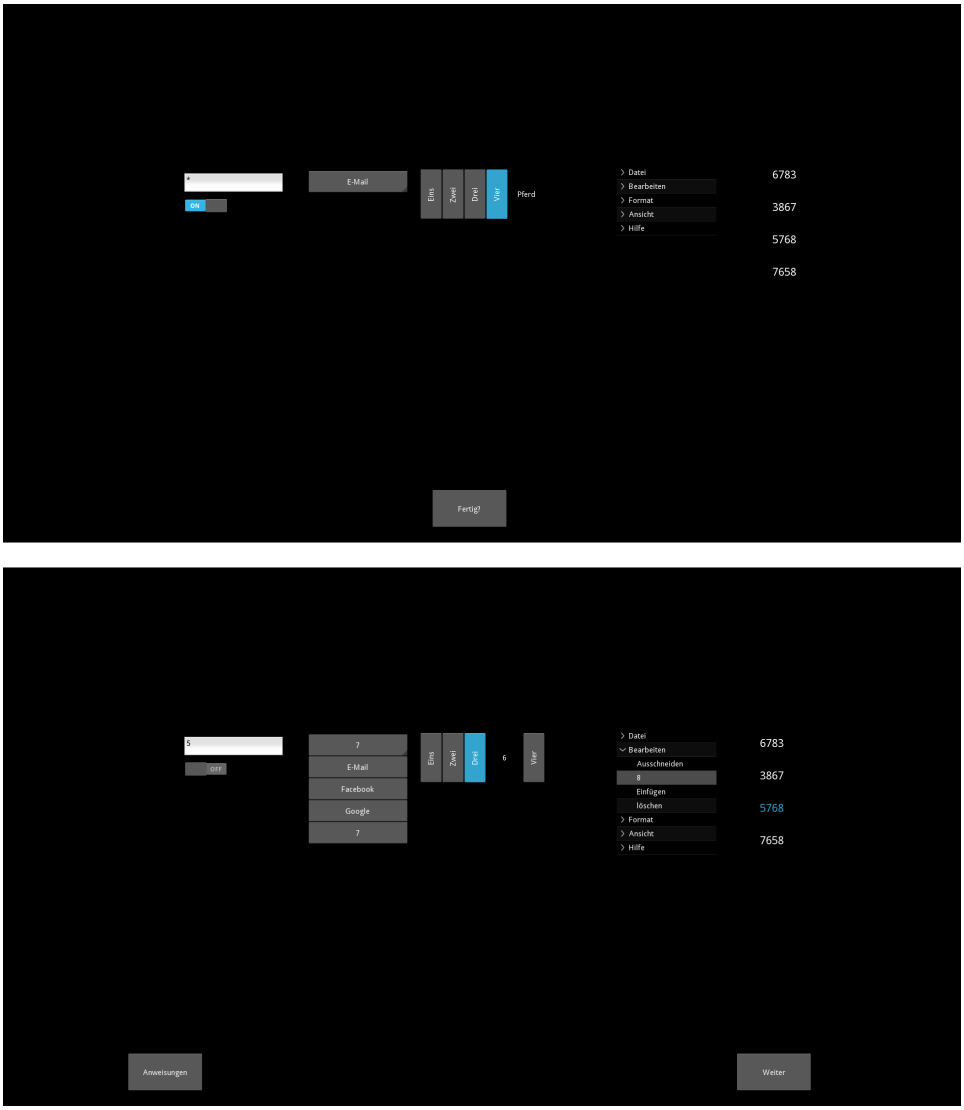

Abbildung 19: Test Zahlenfolge, oben Ausführender, unten Erklärender

Test Zahlenfolge

In diesem Test sind verschiedene Menüstrukturen und Bedienelemente vereint worden. Ziel ist es in den Menüs eine Zahlenfolge zu finden und diese anschließend in der Auswahl rechts zu markieren. Alle vier dieser Menüelemente sind mit kleinen Veränderungen versehen, um sie für den Test etwas komplexer zu machen. Das erste Textfeld ist zunächst als Passwortfeld eingestellt, der Text ist daher nicht lesbar. Ein Schalter wechselt die Ansicht zu einem normalen Textfeld. Das zweite Menü lässt sich nur öffnen, indem der Benutzer darin mit zwei Fingern nach unten wischt. Es öffnet sich ein Auswahlmenü. Der dritte Block ist ein Menü, das sich durch einfaches Tippen umschalten lässt, allerdings wird nur der Inhalt der

aktuellen Zelle angezeigt. Beim vierten Element können die Unterkategorien nur durch einen kleinen Pfeil aufgeklappt werden. Hat der Proband alle vier Zahlen gefunden muss er die richtige Zahlenfolge mit einer Geste auswählen. Eine Hand hält die Zahlenfolge, während die andere einen Bestätigungshaken zeichnet. Der Aufbau ist in Abbildung 19 zu sehen.

Aufgaben Typ 3: Bildschirminhalte
Die dritte Gruppe bezieht sich nicht auf die Bedienung von Benutzeroberflächen, stattdessen es sollen Informationen über dargestellte Bildschirminhalte vermittelt werden.

Test Text
Den Probanden liegt hier ein langer Text vor, in dem zwei Wörter gefunden und markiert werden müssen. Es sind keine Zeilennummern angegeben. Daher muss der Erklärende andere Wege zur Beschreibung der Position finden. Das zwei Personen sich über eine bestimmte Stelle in einem Text unterhalten wollen ist eine bekannte Situation, die man sich bei einer entfernten Kollaboration vorstellen kann. Neben der Orientierung über Zeilennummern oder Absätze bleibt noch eine inhaltliche Orientierung, wenn beiden Personen der Text bekannt ist. Das direkte Zeigen auf eine Textstelle sollte hier ein schnelleres und präziseres Zusammenarbeiten ermöglichen. Der Aufbau ist in Abbildung 20 zu sehen.

Test Karte
In diesem Test muss der Erklärende dem Ausführenden einen Weg auf einer Karte vermitteln. Beiden liegt dabei eine Karte vor, auf der nur für den Erklärenden der Weg bereits vorgegeben ist. Einer zweiten Person eine Wegbeschreibung nur mit Hilfe von Straßennamen und markanten Objekten, wie beispielsweise einem Kreisverkehr, zu erklären kann zu Problemen führen. So ist es zum Beispiel wichtig zu wissen, ob Richtungen aus Sicht des Fahrers oder nach Orientierung der Karte gegeben werden. Wenn dem Erklärenden seine Hand als Werkzeug zur Verfügung steht, hat er die Möglichkeit den genauen Weg auf der Karte anzuzeigen. Der Ausführende könnte den Weg direkt nachverfolgen und einzeichnen. Der Aufbau ist in Abbildung 21 zu sehen.

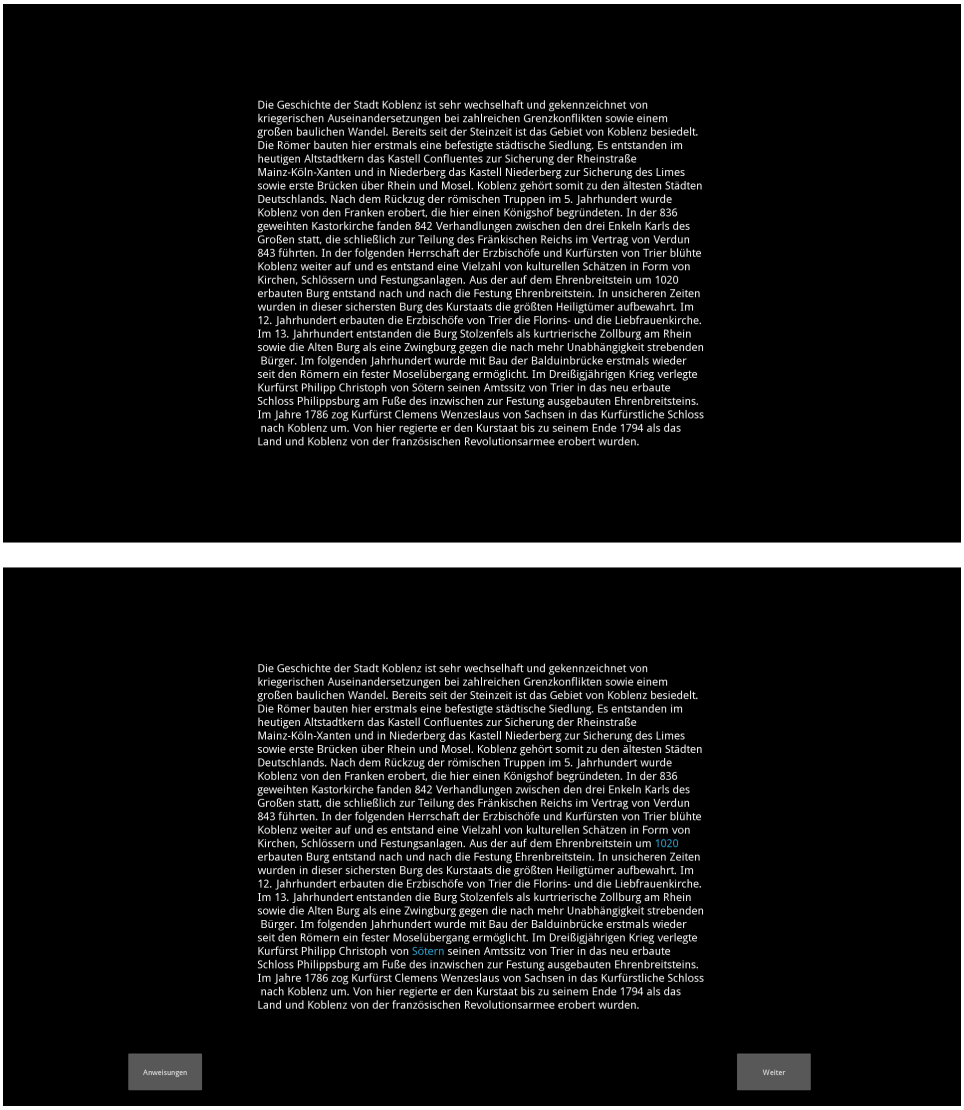

Abbildung 20: Test Text, oben Ausführender, unten Erklärender

Abbildung 21: Test Karte, oben Ausführender, unten Erklärender

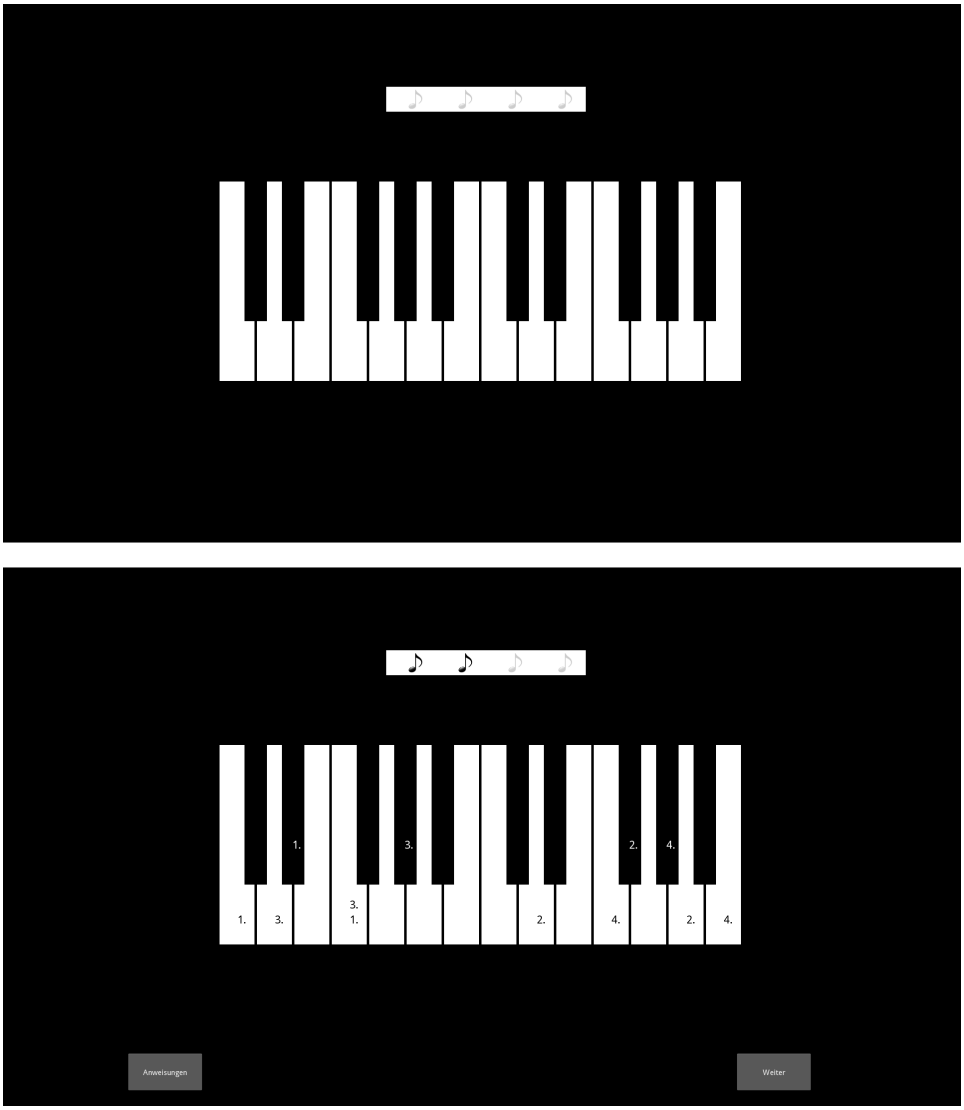

Abbildung 22: Test Klaviatur, oben Ausführender, unten Erklärender

Test Klaviatur

Dem Probanden liegt in diesem Test eine Klaviatur mit 24 Tasten vor. Sie besteht aus 14 Untertasten und 10 Obertasten. Um dem Probanden eine Rückmeldung über die gedrückte Taste zu geben, verfärbt sich diese. Im oberen Bereich befinden sich vier, zunächst ausgegraute, Notensymbole. Diese geben Rückmeldung über das erfolgreiche Abschließen einer Teilaufgabe. Ziel dieses Tests ist, es dem Probanden eine praxisnahe Aufgabe zu stellen. Der Proband hat die Aufgabe vier verschiedene Griffe auf der Klaviatur zu spielen, von denen immer zwei gleichzeitig gehalten werden müssen.

Mit der *Hand* kann der Erklärende seine Hände effektiv nutzen, in dem er die Griffe

selbst spielt und seinem Gegenüber so die Griffe anzeigt. Auf rein verbalem Wege besteht die Möglichkeit die Tasten abzuzählen, sowie die Unterscheidung in Unter- und Obertasten zu nutzen. Der Aufbau ist in Abbildung 22 zu sehen.

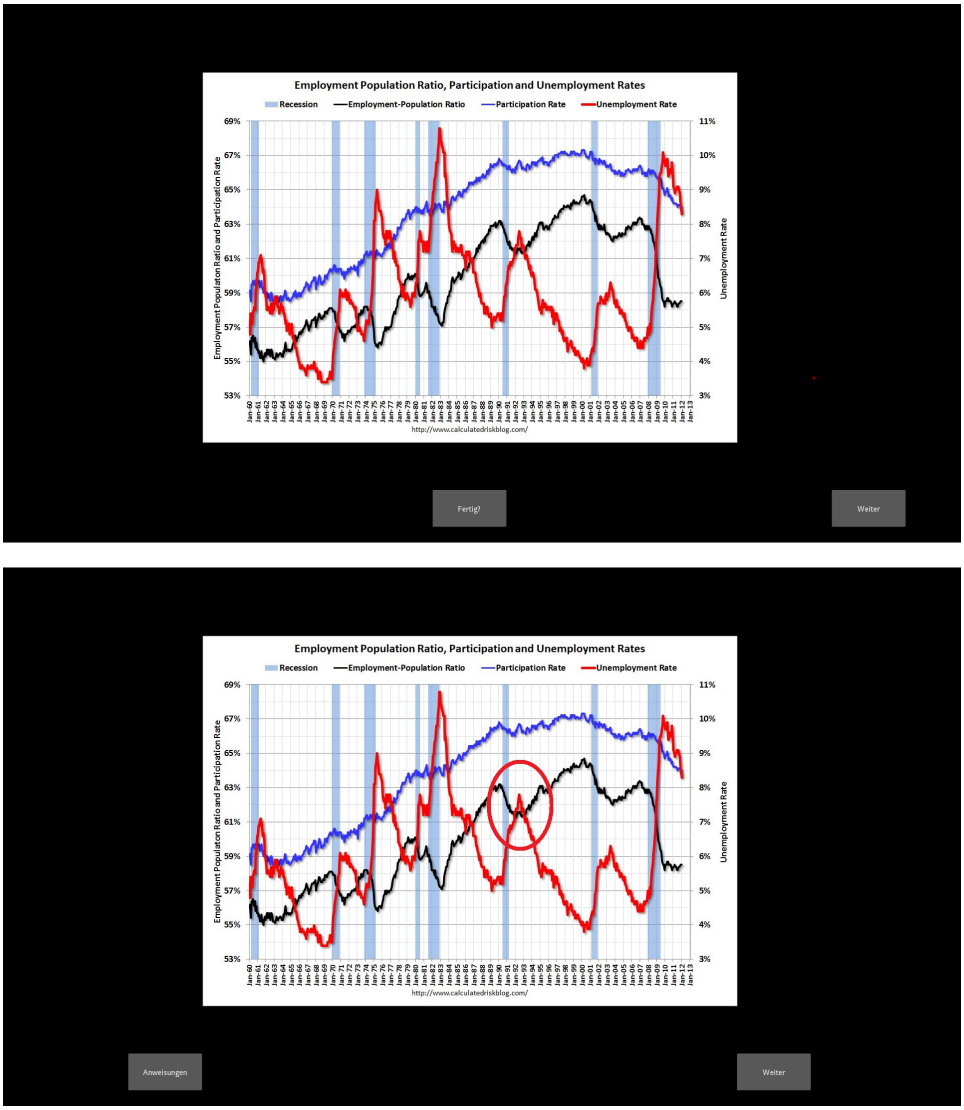

Abbildung 23: Test Graph, oben Ausführender, unten Erklärender

Test Graph

Dieser Test simuliert die gemeinsame Bearbeitung eines Graphen. Der Erklärende muss sein Gegenüber auf eine markante Stelle im Verlauf eines Graphen aufmerksam machen. Um zu bestätigen, dass der Ausführende die korrekte Stelle gefunden hat, muss er diese mit einem Kreis markieren. In der Theorie erhält der Erklärende hier die Möglichkeit auf die gesuchte Stelle zu zeigen, wodurch der Ausführende

direkt sieht was die gesuchte Position im Graphen ist. Während ohne *Hand* die angegeben Achsen, Farben der Kurven und der jeweilige Verlauf der Graphen genutzt werden müssen, um den Ausführenden auf die gesuchte Stelle aufmerksam zu machen. Der Aufbau ist in Abbildung 23 zu sehen.

3.5.3 Visualisierung der Hand

Streaming
Das Streaming der Kinect Daten wird über das in Kivy integrierte GStreamer-Framework realisiert. Für das Senden über das Netzwerk werden zwei Pipelines benötigt. Eine, die als Sender-Pipeline fungiert und eine, die die Daten empfängt und zur Weiterverarbeitung bereitstellt.
Um auf die Tiefendaten des Sensors zugreifen zu können, wird OpenNI genutzt. Der Sensor wird gestartet und aus einem Stream der Tiefendaten das aktuelle Frame abgegriffen. Daraus wird der Buffer ausgelesen und in den GStreamer internen Buffertyp umgewandelt. Um diesen Buffer direkt in eine Pipeline geben zu können, wird dieser Ablauf in ein GStreamer Plugin *kinectdepthsrc* gekapselt. Dieses steht danach als eigenes Element zur Verfügung und kann in die Sender-Pipeline eingebaut werden. Die fertige Pipeline ist wie folgt aufgebaut:

```
'kinectdepthsrc'
'! ffmpegcolorspace '
'! video/x-raw-gray,
    width=640, height=480,
    bpp=(int)16, depth=(int)16 '
'! ffmpegcolorspace '
'! rtpvrawpay'
'! udpsink host=REMOTE_HOST, port=5000'
```

Codebeispiel 2: Die Sender Pipeline

Für die Übertragung wird das *Real-Time Transport Protocol (RTP)* verwendet. Das Protokoll baut auf dem *User Datagram Protocol (UDP)* auf, das einen verbindungslosen und schnellen Transport über das Internet ermöglicht. Die Hauptanwendung von RTP liegt in der Übertragung von echtzeitsensitiven Daten, wie Video-Streams. Ein zusätzlicher Codec wird in diesem Fall nicht verwendet, da nur gering aufgelöste Bilder verschickt werden und die Bandbreite für eine stabile Übertragung ausreichend ist. In der Testanwendung des Ausführenden wird die entsprechende Empfänger-Pipeline initialisiert und ebenfalls gestartet.

Sie besteht aus diesen Elementen:

```
'udpsrc port=5000 '
'caps = "application/x-rtp,
    media=(string)video,
    clock-rate=(int)90000,
    encoding-name=(string)RAW,
    sampling=(string)YCbCr-4:4:4,
    depth=(string)16,
    width=(string)640,
    height=(string)480" '
'! rtpvrawdepay '
'! ffmpegcolorspace '
'! video/x-raw-rgb,
    width=640,height=480,
    bpp=(int)16,depth=(int)16 '
'! ffmpegcolorspace '
'! appsink name=out emit-signals=true sync=false'
```

Codebeispiel 3: Die Empfänger Pipeline

Die Klasse, in der die Pipeline läuft, besitzt ein Attribut *databuf*, in das der aktuelle Buffer gespeichert wird, wenn er fertig übertragen ist. Diese Buffer werden in einer *Queue* gespeichert. Über die *pop*-Methode können die so gespeicherten Buffer wieder aus der Queue entnommen werden. So stehen die Buffer für das Kivy-Widget zum Abruf bereit und können dort weiterverarbeitet werden. Der Zugriff auf die reinen Bufferdaten wird durch das Element *appsink* ermöglicht.

Grafische Bearbeitung

Der Code zur Visualisierung der Eingabehand läuft auf dem Computer des Ausführenden. Hier werden die Daten aus der GStreamer-Empfängerpipeline angenommen. Die bereitgestellten Buffer werden in einer Kivy-Textur gespeichert und so in den passenden Datentyp konvertiert. Da in den Buffern die Daten des unbearbeitetem Kinect-Tiefenbildes stehen, müssen darauf die weiteren Bearbeitungsschritte angewendet werden. Als erstes muss der richtige Ausschnitt des Tiefenbildes gewählt werden. In Abbildung 24 sieht man das gesamte Eingabebild. Um Anforderung 24 erfüllen zu können, muss der gewählte Texturausschnitt der Größe des Eingabefensters entsprechen. Da beide Testanwendungen im Vollbildmodus auf identischen Monitoren laufen, wird die gesamte Bildschirmregion ausgeschnitten (siehe Abbildung 25). So entspricht die Position der Eingabehand am Bildschirm des Erklärenden, der Position der angezeigten Hand auf dem Bildschirm des Ausführenden. Die reinen Kinect Daten werden auf den Bildschirmbereich zugeschnitten und in mehreren Shader-Durchläufen bearbeitet. Durch den Zuschnitt erhält man eine Textur mit einer geringen Auflösung von 360x175 Pixeln.

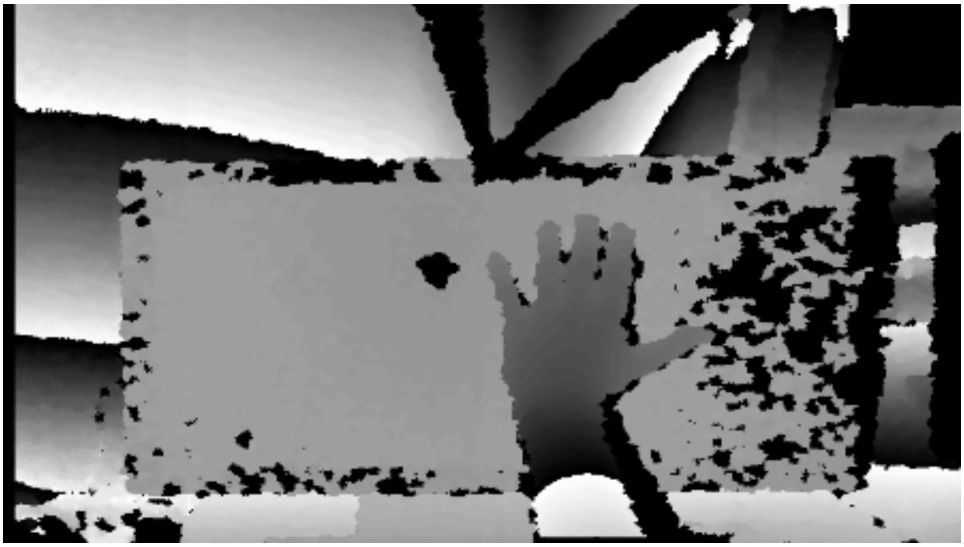

Abbildung 24: Ausgangsbild, unbearbeitet Kinect Daten

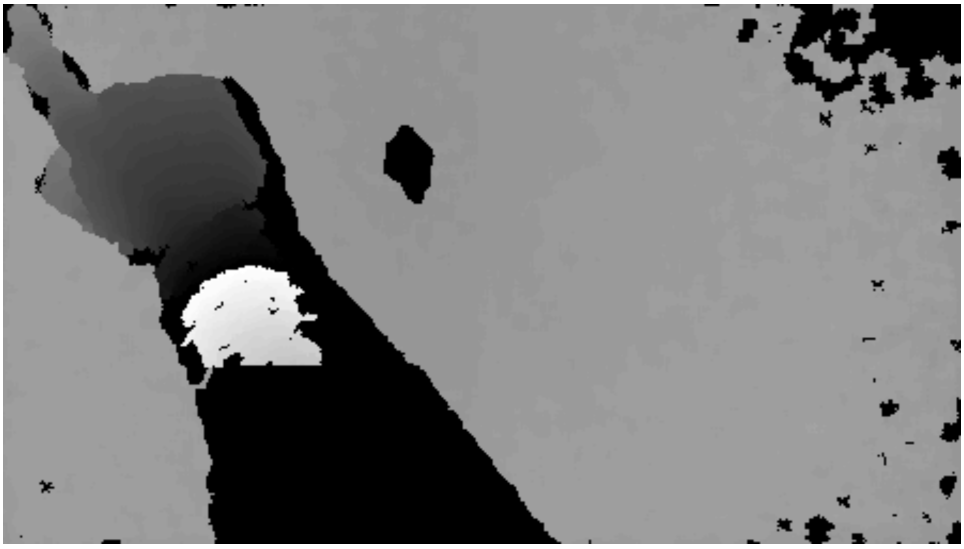

Abbildung 25: Zugeschnittenes Ausgangsbild

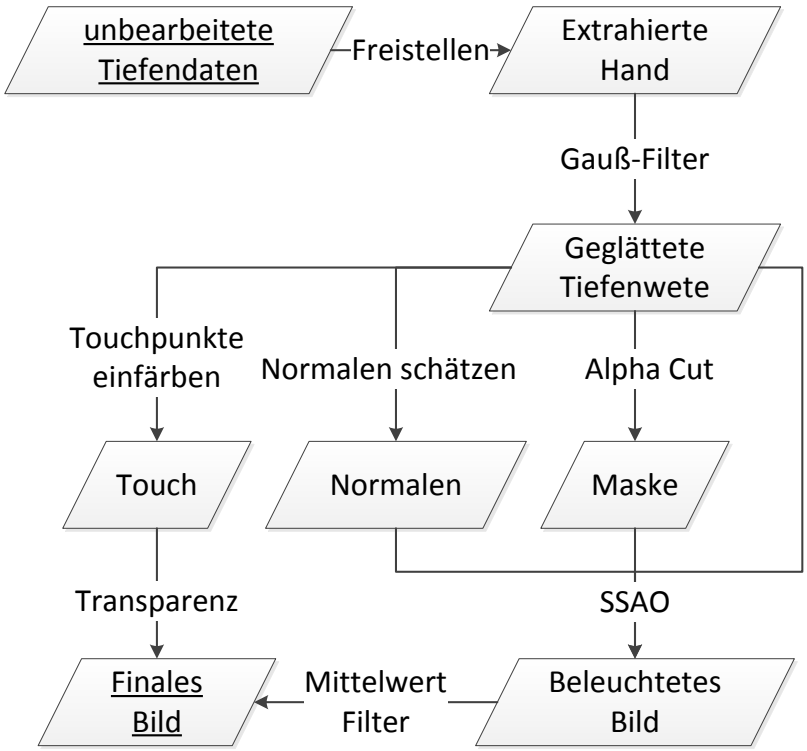

Abbildung 26: Ablauf der Bearbeitungsschritte der Visualisierung

Die finale Visualisierung der Hand wird in einem extra Widget über das Hintergrundprogramm gelegt. Der Hintergrund ist hierbei transparent. Die Darstellung der Hand ist teilweise transparent, um die Hintergrundinformationen nicht komplett zu verdecken. Zusätzlich wird eine indirekte Beleuchtung durch einen Shader, der Screen Space Ambient Occlusion berechnet, angenähert. Dadurch soll ein plastischerer Eindruck der *Hand* und somit eine deutlichere Einschätzung der Handbewegung und -haltung erreicht werden. Um anzuzeigen wann der Bildschirm berührt wird, wird die Hand ausgehend von dem berührten Punkt blau eingefärbt. Somit werden die Touch-Eingaben des Erklärenden visualisiert. Ein Überblick der Shader-Durchläufe ist in Abbildung 26 zu sehen.

Vorverarbeitung
Um nur die Hand abzubilden, muss diese als ersten Schritt von dem Hintergrund extrahiert werden. Das wird durch einen Schnitt der Tiefenwerte an einem Grenzwert erreicht. Da die Entfernung des Bildschirms zum Sensor nicht überall gleich ist, wird der Grenzwert abhängig von der Position gewählt. So kann für verschiedene Zonen ein möglichst genauer Trennwert gewählt werden.

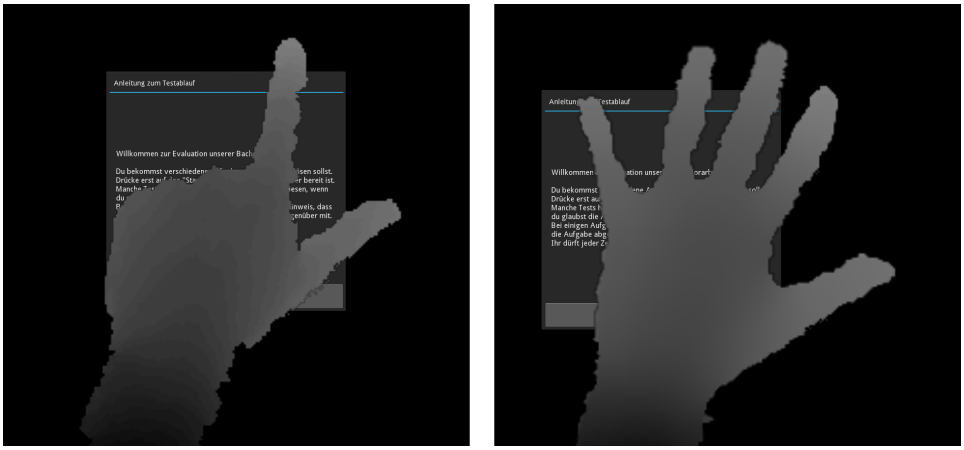

Abbildung 27: Die freigestellte Hand: links unbearbeitet, rechts geglättet

Alle Tiefenwerte die kleiner als der Grenzwert, bekommen einen alpha-Wert von 1, alle anderen werden zu Hintergrundpixeln mit dem alpha-Wert 0. Das Ergebnis der Freistellung sieht man in Abbildung 27. Aufgrund der Unsicherheiten in den Kinect Daten werden die Ränder der *Hand* unsauber dargestellt. Durch die geringe Auflösung und der Abbildung von mehreren Tiefenwerten auf einen Grauwert erhält man Stufenartefakte. Um diese zu entfernen, wird das Bild geglättet. Dazu wird ein Gauß-Filter implementiert. Zur Steigerung der Performanz wurden hierfür zwei Shader verwendet. Einen zur Glättung in horizontaler und einen in vertikaler Richtung. Es werden nur die Werte der Handpixel, gekennzeichnet durch den alpha-Wert 1, geglättet. Dies verhindert, die Veränderung der Tiefenwerte durch die schwarze Umgebung.
Als nächstes soll der Umriss der *Hand* geglättet werden. Dazu wird der Gauß-Filter auf alle Werte des Bildes angewendet. Durch die unterschiedlichen alpha-Werte entstehen neue Werte, die zwischen 0 und 1 liegen. In Abbildung 28 ist das Ausgabebild zu sehen und die fertige Maske, die aus diesem entsteht. Hierfür wird ein alpha-Wert gewählt, an dem die Pixel zwischen Hintergrund und *Hand* getrennt werden. Alle Pixel mit einem höheren alpha-Wert werden weiß gefärbt, alle mit einem niedrigeren Wert schwarz. Alle Pixel werden voll opak.
Als Grundfarbe wird weiß gewählt, um einen Kontrast zu der Hintergrundfarbe zu schaffen. Hier besteht die Möglichkeit, andere Farben auszuwählen. Dadurch kann die Anforderung 28 erfüllt werden, die Hände mehrerer Anwender zu unterscheiden. In der Praxis wären dafür mehrere Kinect Sensoren nötig. An allen Arbeitsplätzen werden die Hände aufgezeichnet und versendet. Es gibt je eine Pipeline und ein Kivy Widget, in dem die Hand dargestellt wird. Fügt man diese einfach dem Widget-Baum der Hauptanwendung zu, werden sie durch Kivy automatisch übereinander geblendet. In Abbildung 29 ist ein vorstellbares Ergebnis zu sehen. Die Hände des einen Anwenders werden in blau, die des anderen in rot dargestellt. Dies erfüllt Punkt 28 der Anforderungsliste.

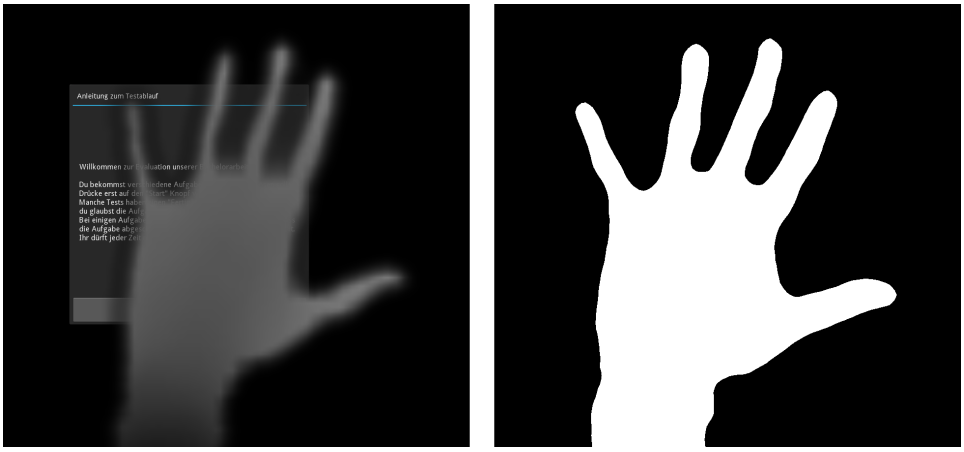

Abbildung 28: Geglättete Ausgangswerte und fertige Maske der Hand

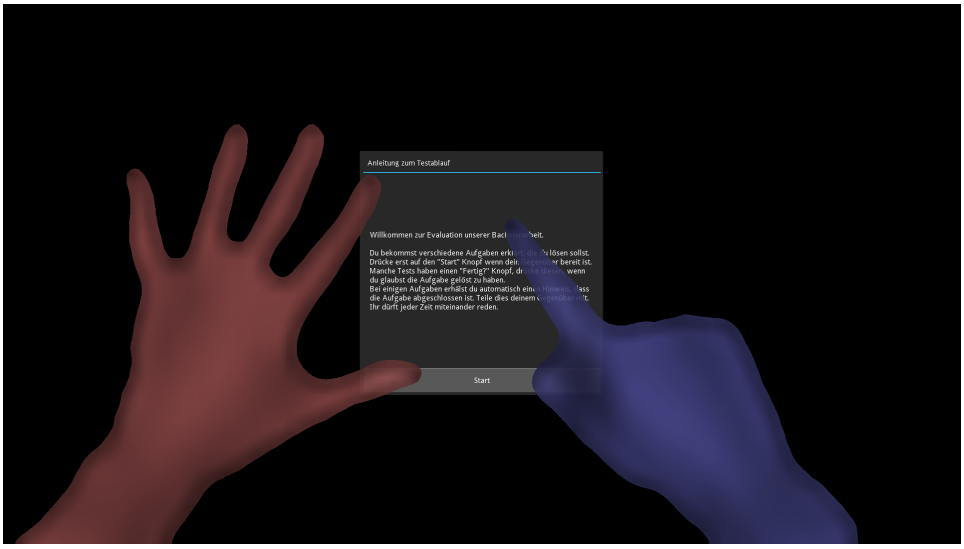

Abbildung 29: Unterschiedliche Farben für unterschiedliche Anwender, normale Transparenz

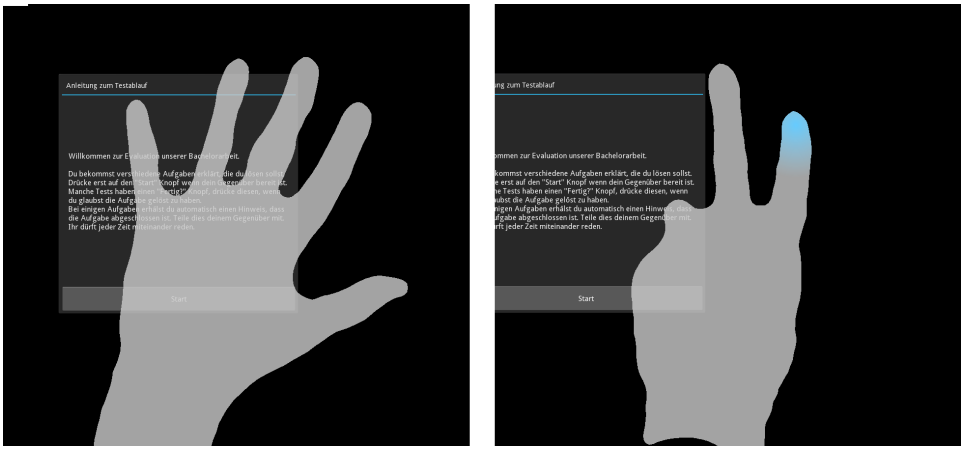

Abbildung 30: Tranzparenz, Kennzeichnung bei Bildschirmberührung

Touchpunkte

Wie in Anforderung 25 gefordert, soll die Eingabe der Hand deutlich werden. Um das zu gewährleisten, muss unter anderem erkennbar sein wann die Hand den Bildschirm berührt. Die hierfür benötigten Touchpunkte werden von der Anwendung des Erklärenden über das Twisted-Modul übertragen. In der Anwendung des Ausführenden werden die empfangenen Koordinaten gespeichert. Sie dienen gemeinsam mit der bisherigen Ausgabetextur und der Maske als Eingabe des *touch*-Shaders. In diesem werden die Pixel der *Hand* abhängig von der Entfernung von aktuellen Touchpunkten eingefärbt (siehe Codebeispiel 4).

```
1 float diff = length(vec2(x*1.7, y)
2              - vec2(tex_coord0.x*1.7,tex_coord0.y))*10;
3
4 float op = 1 - clamp(diff, 0.0, 1.0);
5 col.rgb = op * vec3(0.4, 0.8, 1.0) + (1.0 - op) * col.rgb;
```

Codebeispiel 4: Auszug aus Shader zu Visualisierung der Touchpunkte

So entsteht ein kreisförmiger Farbverlauf um jeden Touchpunkt. Mit Hilfe der Maske werden nur die Handpixel gefärbt (30). Alle Hintergrundpixel werden per *discard* ignoriert. Aufgrund der Begrenzung durch die verwendeten Monitore, können maximal zehn verschiedene Touchpunkte angezeigt werden.

Beleuchtung

Um die in Anforderung 26 geforderte Bedingung zu erfüllen, soll eine indirekte Beleuchtung angenähert werden. Hierzu wird ein SSAO-Verfahren angewendet. Dieses Verfahren eignet sich gut, da es auf bereits gerenderten Bildern arbeitet. So kann man es direkt auf der Textur anwenden, ohne vorher Geometrie zu erstellen. Für die Berechnung benötigt man die Tiefenwerte und die Normalen. Die Tiefenkarte liefert der Kinect Sensor. Da keine Geometrie vorliegt, müssen die Normalen aus den Tiefendaten angenähert werden.

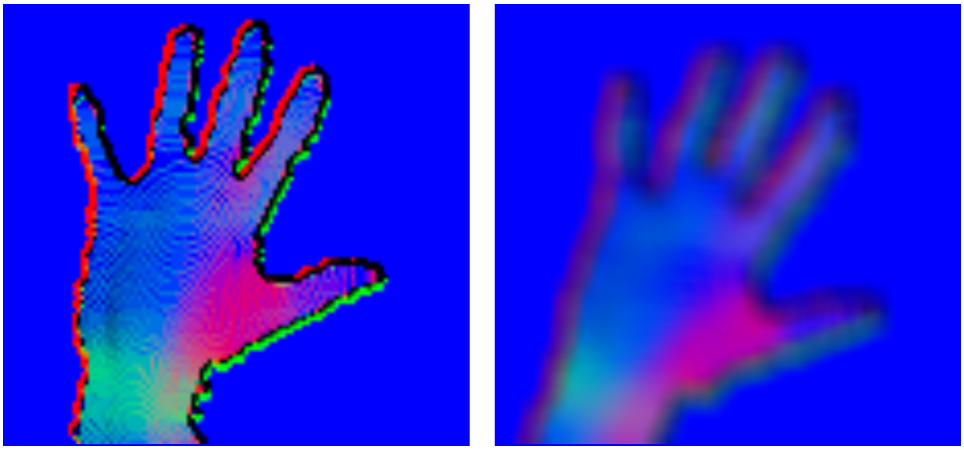

Abbildung 31: Angenäherte Normalen

Normalenberechnung: Für die Annäherung von Normalen werden in dem zuständigen Shader für jeden Pixel seine Nachbarn mit den zugehörigen Tiefenwerten betrachtet. Daraus werden Tangenten an den Punkt berechnet. Es entstehen je ein Vektor mit dem Gradienten der Tiefe in x-Richtung und y-Richtung Das Kreuzprodukt dieser Tangentenvektoren liefert eine Normale für den betrachteten Punkt. Zusätzlich wird ein Vorzeichenwechsel der z-Achse vorgenommen, um sicherzustellen das die Normalen immer in Blickrichtung zeigen (siehe Codebeispiel 5).

```
1 vec3 normal = cross(tangent1, tangent2);
2 normal.z = -normal.z;
3 normal = normalize(normal);
```

Codebeispiel 5: Normalenannährerung durch Kreuzprodukt

Durch verbliebene Stufenartefakte im Ausgangsbild entstehen falsche Annahmen der Normalen, die ebenfalls zu Stufen im Ausgabebild führen (siehe Abbildung 31). Dem wird mit einer nachfolgenden einfachen Gauß-Glättung entgegen gewirkt. Die Normalen werden in einer weiteren Textur gespeichert und zusammen mit dem Tiefenbild als Eingabe für den SSAO-Shader verwendet.

SSAO: Um den Verdeckungsfaktor für jeden Punkt ausrechnen zu können, wird die umgebende Hemisphäre zufällig abgetastet. Die Menge der Abtastpunkte bestimmt über Qualität und Rechenzeit. Da für jeden abgetasteten Punkt ein Texturzugriff erfolgen muss, wird die benötigte Rechenzeit schnell zu hoch, um in Echtzeit ausgeführt werden zu können. Bei wenigen Abtastpunkten entstehen Störmuster im Ausgabebild, die jedoch mit einem Mittelwertfilter entfernt werden können. Der hier implementierte Shader arbeitet mit 16 Abtastpunkten. Normalerweise werden die Samples auf einer Halbkugel gewählt und mit den rückproijzierten Punkten in Weltkoordinaten verglichen. Da hier keine Geometrie zugrunde liegt, entfällt dieser Schritt. Die Samples werden innerhalb eines Kreises um den betrachteten Punkt gewählt und mithilfe der Tiefenwerte direkt verglichen.

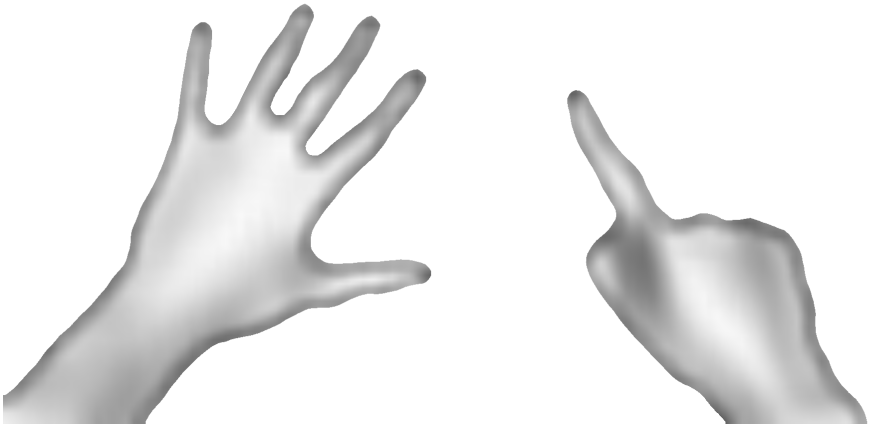

Abbildung 32: SSAO-Anteil der Pixel

Für jeden dieser Punkte wird so ein Verdeckungsfaktor ausgerechnet. Dazu wird das Skalarprodukt zwischen der Normalen des betrachteten Punktes und dem Verbindungsvektor zu dem Abtastpunkt berechnet. Ist dieses positiv ist der Samplepunkt im vorderen Halbraum und trägt zur Verdeckung bei. Ist es negativ wird der betrachtete Punkt nicht verdeckt, da der Samplepunkt weiter hinten liegt. Der Verdeckungsfaktor wird ausgerechnet durch die Multiplikation des Skalarproduktes, oder Null bei negativem Ergebnis, mit einem Gewichtungsterm.

```
1 float ao = max(0.0, dot(n,v)) * (1.0 / (1.0 + d)) * 1.5;
```

Codebeispiel 6: Berechnung des Verdeckungsfaktors

Der Verdeckungsfaktor von allen Samples wird aufaddiert und gemittelt. In Abbildung 32 wird die endgültigen SSAO-Textur gezeigt. Ein Pixel ohne Verdeckung ist weiß und mit großer Verdeckung schwarz dargestellt. Die Textur wird zum Schluss über die finale Textur der *Hand* geblendet.
Im Vergleich zu der Abbildung der Hand ohne den SSAO-Anteil wird deutlich wie die angenäherte Beleuchtung zum plastischen Eindruck beiträgt. In Abbildung 30 wirkt die Darstellung flach und man kann die Handhaltung nur anhand der Silhouette erahnen. In Abbildung 32 erkennt man durch die Schattierung, die Form der *Hand*. Dadurch können auch übereinander liegende Objekte besser differenziert werden.

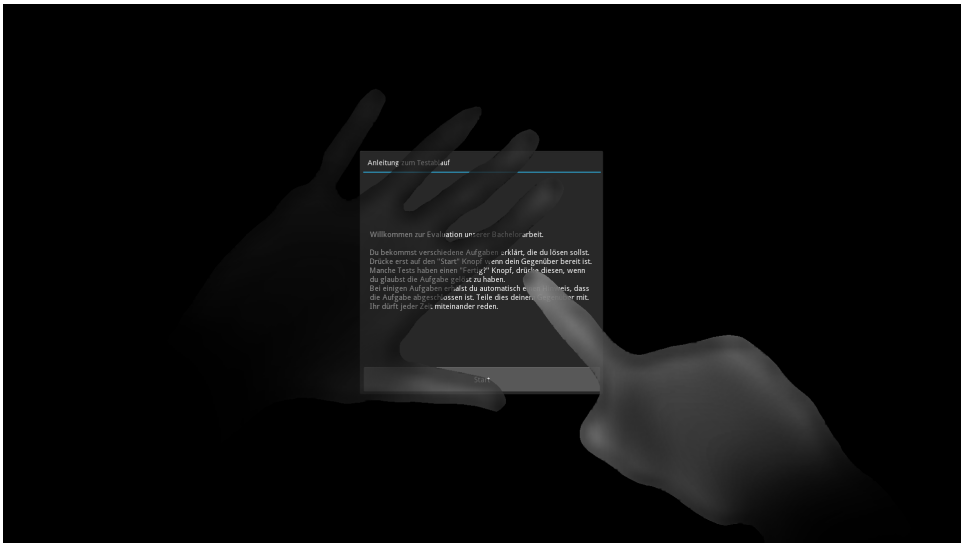

Abbildung 33: Endergebnis höhenabhängige Transparenz

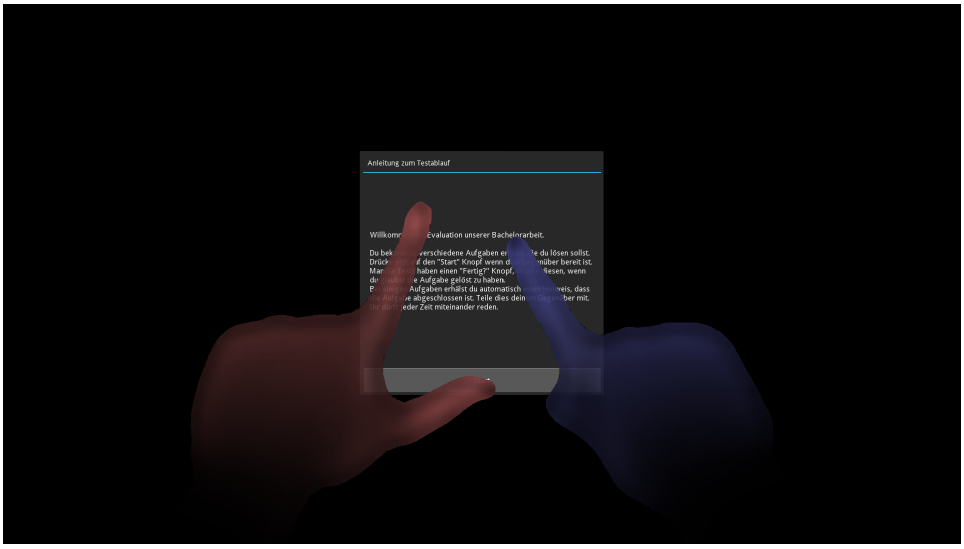

Abbildung 34: Unterschiedliche Farben bei höhenabhängiger Transparenz

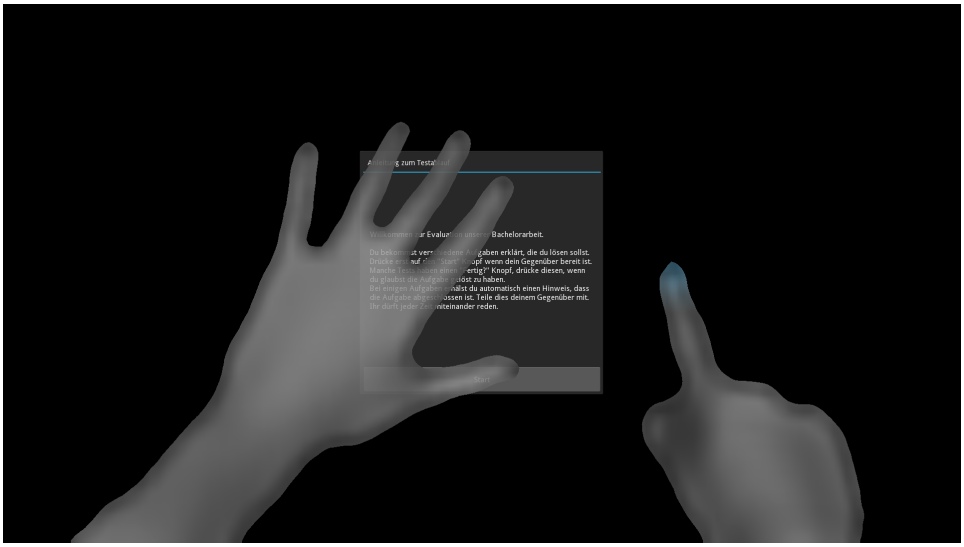

Abbildung 35: Endergebnis einheitliche Transparenz

Finale Komposition

Um Anforderung 27 erfüllen zu können, muss die Verdeckung des Bildschirmes durch die *Hand* verringert werden. Hierzu werden alle Pixel der *Hand* transparent dargestellt. Die Transparenz kann entweder gleichmäßig für die ganze *Hand* gelten, oder abhängig von ihrer Höhe berechnet werden (siehe Abbildung 33). Bei Abhängigkeit zur Höhe kann zwar die Anforderung 26 besser erfüllt werden, da zusätzlich ein Hinweis für die Lage der Hand gegeben wird. Allerdings kann es so dazu kommen, dass die *Hand* mit dem Hintergrund verschwimmt und nicht mehr genau vom Hintergrund zu trennen ist (siehe Abbildung 34). Das würde Anforderung 29 widersprechen. Diese Textur wird mit der Textur der SSAO-Anteile überblendet. Abschließend wird die Auflösung auf die des Bildschirms angepasst. Das bewirkt eine letzte Glättung des Bildes durch den internen Box-Filter. Das finale Ergebnis mit konstanter Transparenz ist in Abbildung 35 zu sehen.

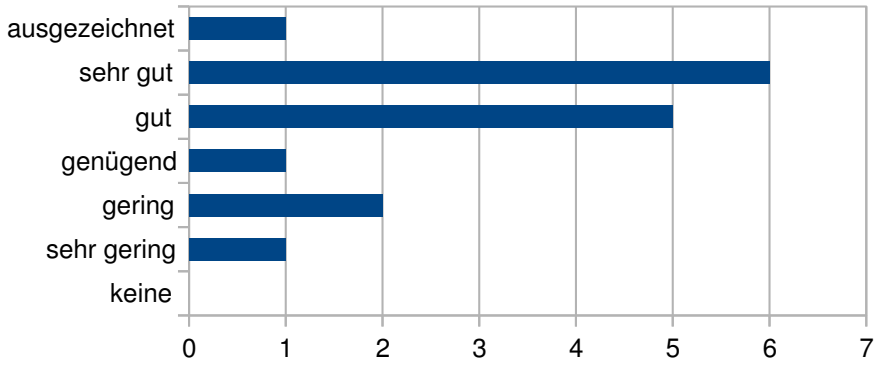

Abbildung 36: „Wie schätzt du deine Erfahrungen mit Touchscreen-Eingabegeräten ein?“

4 Evaluation

4.1 Durchführung

4.1.1 Ablauf und Probanden

Insgesamt werden die Probanden durch 13 verschiedene Tests geführt, in denen sie jeweils eine Aufgabe zu lösen haben. Dabei erhalten sie Anweisungen durch den Erklärenden, dem die gleichen Tests vorliegen. Die 13 Tests gliedern sich in zwei Teile, von denen einer mit der Einblendung der Hand des Erklärers durchgeführt wird und der andere ohne diese. So wird gewährleistet, dass jeder der Probanden eine Aussage über die Nutzung der *Hand* treffen kann. Im ersten Durchlauf übernimmt einer der Entwickler die Rolle des Erklärenden, während der erste Proband die Aufgaben durchführt. Anschließend beantwortet der Proband den ersten Fragebogen zu seiner Rolle als Ausführender. Schließlich übernimmt er selbst die Rolle des Erklärenden und führt einen weiteren Probanden durch die Tests. Anschließend beantwortet er einen weiteren Fragebogen zu seiner Rolle als Erklärender. Durch diese Aufteilung erhalten alle Probanden einen Einblick in die Rolle des Ausführenden, sowie des Erklärenden und können entsprechend Fragen aus beiden Perspektiven beantworten.
Die Tests wurden mit insgesamt 11 weiblichen und 7 männlichen Teilnehmern im Alter von 20 bis 54 Jahren durchgeführt. Das durchschnittliche Alter der Probanden liegt bei 26 Jahren. 15(75%) der Probanden benutzen regelmäßig ein Smartphone, 3(15%) ein Tablet und nur einer einen Computer oder Laptop mit Touchscreen. Einer der Probanden benutzt keine Touchscreen-Geräte. In Abbildung 36 ist die Selbsteinschätzung der Probanden zu Ihrer Erfahrung mit Touchscreen-Geräten zu sehen. Der Medianwert liegt bei „gut“. 12(78%) der Probanden gaben an, selbst schon einmal eine vergleichbare Problemstellung mit einer entfernten Person bearbeitet zu haben.

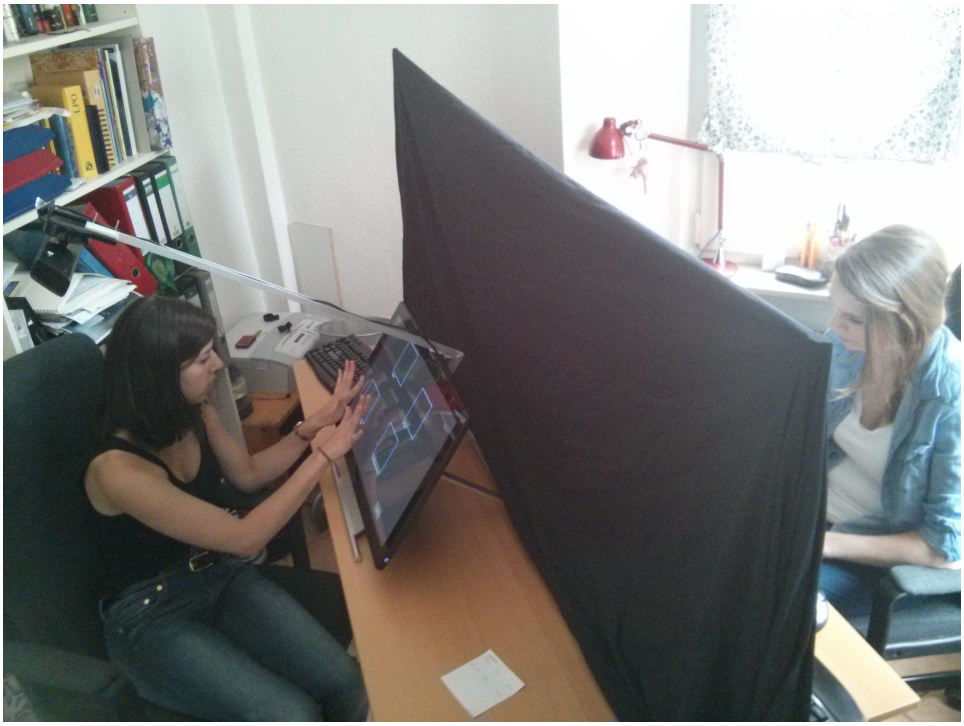

Abbildung 37: Zwei Probandinnen während des Versuchs

4.1.2 Versuchsaufbau

Wie in Abbildung 37 zu sehen, wurden beide Probanden eines Durchlaufs so voneinander getrennt, dass sie zwar miteinander sprechen, sich jedoch nicht sehen können. Dies soll räumlich getrennte Zusammenarbeit simulieren. Dadurch entspricht die Situation im wesentlichen einer, in der nur per Telefon oder ähnlichem kommuniziert werden kann. So wird den Probanden die Möglichkeit genommen ihre Hände zur Erklärung einzusetzen, ausgenommen in den dafür vorgesehenen Tests und unter Verwendung des Kinect-Systems.

4.2 Ergebnisse

4.2.1 Effizientere Ausführung

Alle Probanden konnten die Testreihe erfolgreich abschließen. Die Medianwerte der Testzeiten finden sich in Abbildung 38.
Die Aufgaben vom Typ 1 verwenden geläufige Gesten, dennoch konnten sich ihre Testzeiten im Durchschnitt um 18% verbessern. Die Tests *Ecken*, *Gesten* und *Entsperrmuster* profitieren besonders deutlich von der Verwendung der *Hand*. Ihre Testzeiten verbessern sich um bis zu 43%. Bei den Tests *Gesten* und *Entsperrmuster* hat der Erklärende Zugriff auf Wissen, das der Ausführende nicht haben kann und auch nicht durch Ausprobieren erlangen kann. Deshalb ist der Ausführende

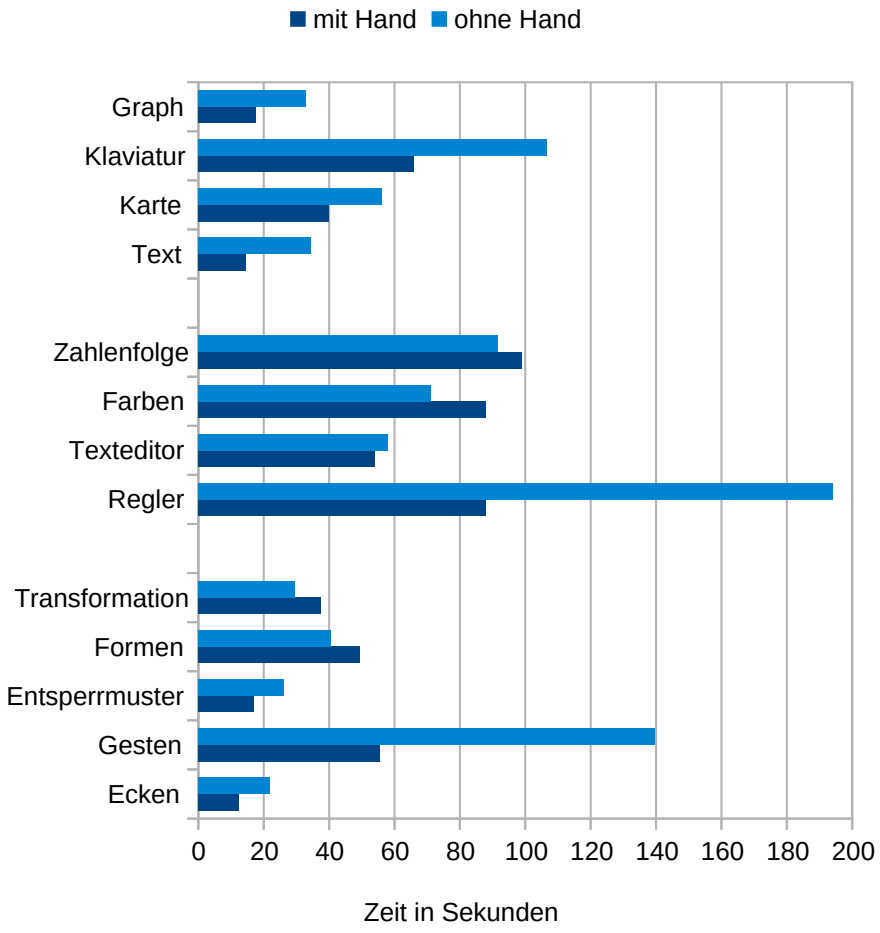

Abbildung 38: Median der Testzeiten aller Probanden

besonders auf gute Erklärungen angewiesen. Bessere Informationsvermittlung bewirkt daher einen großen Vorteil. Dies zeigt deutlich die Richtigkeit von Subhypothese H1.1. Besonders der Test *Gesten* profitiert stark von der Einblendung der Hand. Der Median der Testzeiten liegt ohne *Hand* bei 139 Sekunden, mit *Hand* nur bei 55 Sekunden (siehe Abbildung 38). Dies entspricht einer Verbesserung von ca. 60%. Die Beobachtung der Testpersonen hat gezeigt, dass dies hauptsächlich durch die fehlende Kennzeichnung der Zielbereiche zustande kommt (siehe Abbildung 15). Steht die *Hand* zur Verfügung kann sehr effizient auf die geforderten Bereiche Aufmerksam gemacht werden (siehe Abbildung 39). In den Tests *Transformationen* und *Formen* hingegen bringt es keinen Vorteil die *Hand* zu sehen. Hier kann auch durch Ausprobieren schnell das Ziel erreicht werden. Bei *Formen* wiederholt sich zudem das Bedienkonzept, einmal verstanden kann die Aufgabe

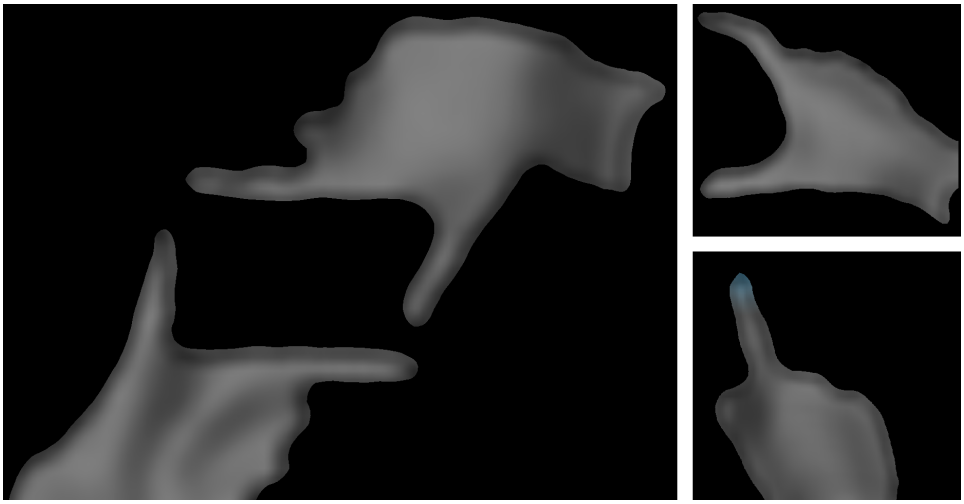

Abbildung 39: Einige Möglichkeiten auf Bildschirmbereiche Aufmerksam zu machen.

schnell zu Ende gelöst werden. Diese Ergebnisse unterstützen Hypothese H1.
Die komplexeren Aufgaben vom Typ 2, mit Ausnahme des Tests *Regler*, profitieren zeitlich nicht durch Einblendung der Hand. Dies lässt sich dadurch erklären, dass die Erklärenden die Aufgaben selbst erst einmal ausgeführt haben. Teilweise sind sie sich nicht sicher wie die Aufgabe genau zu lösen ist und müssen öfter die Aufgabenstellung überprüfen. Durch diese Unterbrechungen kann die *Hand* nicht effektiv genutzt werden. Der Test *Regler* zeigt, anders als der Rest dieser Gruppe, deutliche Unterschiede in beiden Testzeiten. Seine Ausführungzeit verringert sich von durchschnittlich 194 Sekunden auf 88 Sekunden. Das entspricht einer Verbesserung um 55%. Dieser Test weist mit 40 die höchste Standardabweichung von allen durchgeführten Test auf. Die deutliche Verbesserung ist möglicherweise damit zu erklären, dass sich nicht alle Probanden genau an die Aufgabenstellung gehalten haben. Es war gefordert die Funktionsweise der Regler zu erklären. Nicht aber direkt die korrekte Stellung zu verraten, um die Aufgabe ab zu schließen. Eine Bestätigung für Subhypothese H1.2 lässt sich daher nicht ableiten.
Bei den Aufgaben zur Informationsvermittlung war die Bedienung mit angezeigter Hand immer schneller. Diese Tests lassen sich durch einfaches Zeigen auf einen Bildschirmbereich sehr schnell lösen, während ohne die *Hand* die Positionen beschrieben werden müssen. Im Schnitt sind die Testzeiten dieser Gruppe um ca. 49% schneller, wenn die *Hand* verwendet wird. Besonders deutlich verbessert sich die Ausführung der Klaviatur, da diese insgesamt umfangreicher ist als die anderen Tests dieses Typs.

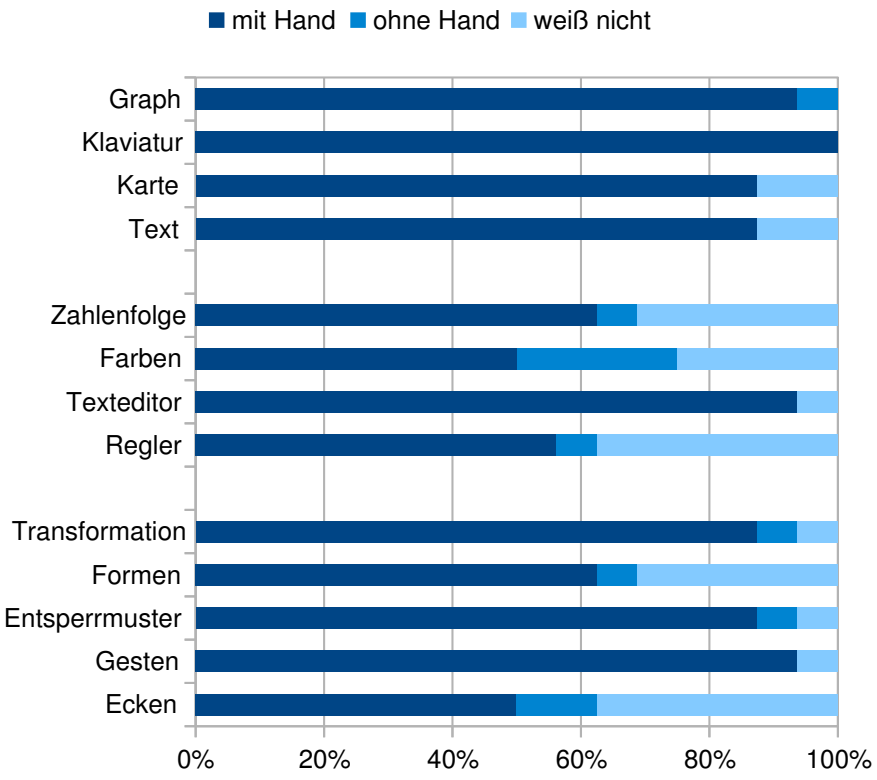

Abbildung 40: „Wenn du eine solche Aufgabe wieder lösen müsstest, würdest du sie lieber ... durchführen."

Nach Abschluss der Tests würden 76% der Probanden die Aufgaben von Typ 1, 66% der Probanden die Aufgaben von Typ 2 und 92% der Probanden die Aufgaben von Typ 3 in Zukunft lieber mit *Hand* lösen (siehe Abbildung 40). Dies zeigt, dass Anwender den Nutzen der *Hand* bei Aufgaben vom Typ 3 als besonders hoch einschätzen. Bei diesem Aufgabentyp liegt der Schwerpunkt auf dem Lenken der Aufmerksamkeit des Gegenübers auf einen bestimmten Teil des Bildschirms. Diese beiden Erkenntnisse unterstützen die Richtigkeit von Subhypothese H1.3.
Allgemein wurde immer die Anzeige der Hand gewünscht, auch wenn dadurch keine messbare Verbesserung entstanden ist (siehe Abbildung 40). Kommentare zeigten, dass die Sichtbarkeit der Hand eine Sicherheit bei der Anwendung gab. Auch wurde die Effizienz besser bewertet, als sie gemessen wurde. Ausgenommen der Tests *Zahlenfolge*, *Regler* und *Ecken* schätzten ca. 83% der Probanden, dass sie für die Lösung der Aufgaben ohne *Hand* länger gebraucht hätten (siehe Abbildung 41). Besonders der Test *Klaviatur* sticht hervor, da hier keiner der Probanden glaubt diese Aufgabe ohne die *Hand* genauso schnell lösen zu können. Beim Test Regler hingegen, glauben 8 Teilnehmer ihn ohne die *Hand* genauso schnell

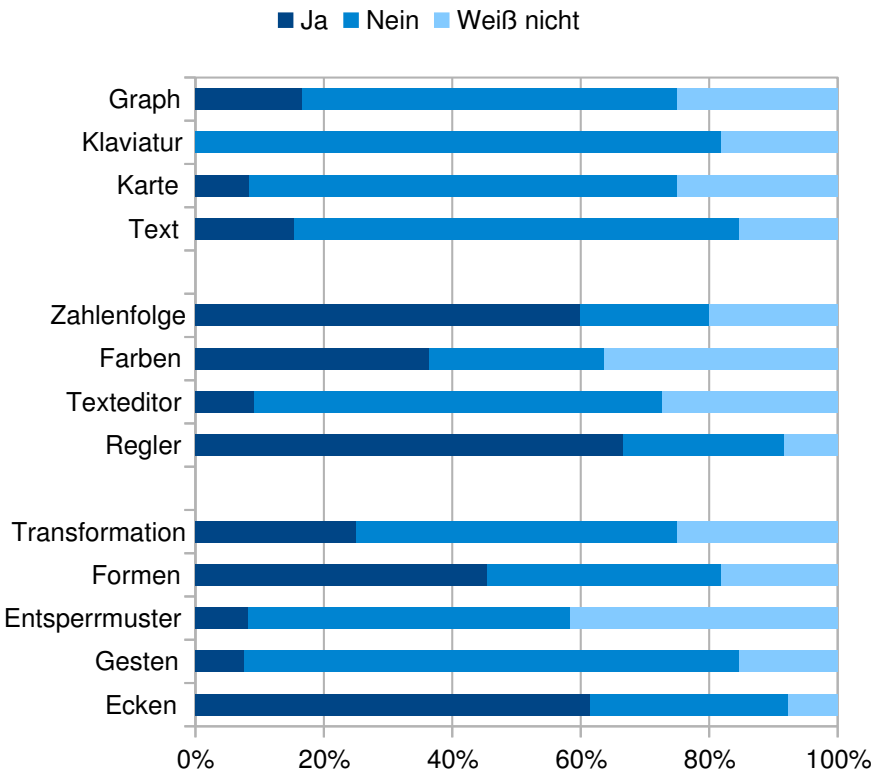

Abbildung 41: „Ich hätte die Aufgabe genauso schnell ohne die *Hand* lösen können.“

abschließen zu können. Im Hinblick auf die Testzeiten profitiert der Test *Regler* allerdings sehr deutlich von der Nutzung der *Hand*. Seine Testzeit verringerte sich um 55% (sieht Abbildung 38). Dass die Zeiten insgesamt nicht deutlich schneller ausfallen, ist unter anderem damit zu erklären, dass die Tests nur wenige Teilziele haben. So nimmt die grundlegende Erklärung der Aufgabe, die in beiden Fällen nötig ist, schon einen großen Teil der gesamten Testzeit ein. Das eigentliche Lösen der Aufgabe macht ca. zwei Drittel der Gesamtzeit aus.
Bei der Erklärung per *Hand* wird oft gewartet bis der Erklärende mit dem Zeigen fertig ist. Während bei der rein verbalen Erläuterung sofort ausprobiert wird und so schneller ein Ergebnis erreicht wird. Die Probanden zeigten Hemmungen mit den Bildschirm zu interagieren, während dieser schon von einer anderen *Hand* benutzt wird.
Wenn Erklärender und Ausführender der Ablauf der Aufgabe nicht klar ist, hat die Einblendung der Hand wenig Nutzen. Wider Erwarten ist während der Evaluation aufgefallen, dass die Probanden nicht immer effektiv ihre Hände zur Erklärung einsetzten, obwohl dies natürlich erscheint. Oft wurde vergessen die Hände zu be-

nutzen, wenn die Möglichkeit dazu bestand. Teilweise wurde durch den Ausführenden darauf aufmerksam gemacht: „Kannst du mir nicht einfach zeigen wo...“. Dies sollte jedoch bei Anwendern, die ein solches System regelmäßiger benutzen nicht auftreten. Bei Erklärenden, die sowohl die Aufgaben kennen, als auch die Einblendung der Hand bereits genutzt haben, ist der Vorteil deutlicher zu sehen. Trotz der nicht optimalen Nutzung der *Hand*, wünschen sich die Erklärenden, auch in Zukunft die *Hand* benutzen zu können (siehe Abbildung 42).

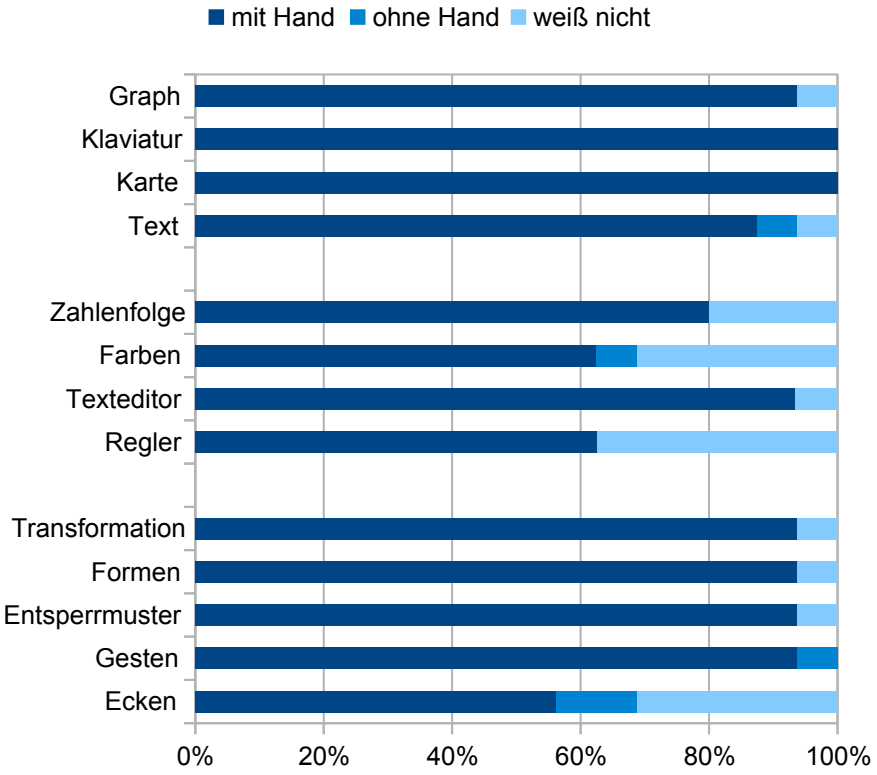

Abbildung 42: „Wenn du eine solche Aufgabe wieder erklären müsstest, würdest du sie lieber ... erklären?“

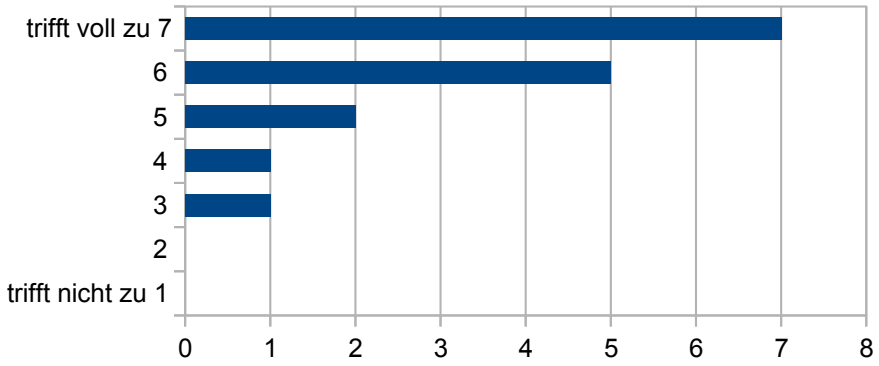

Abbildung 43: „Die Darstellung der Hand war ausreichend präzise.“

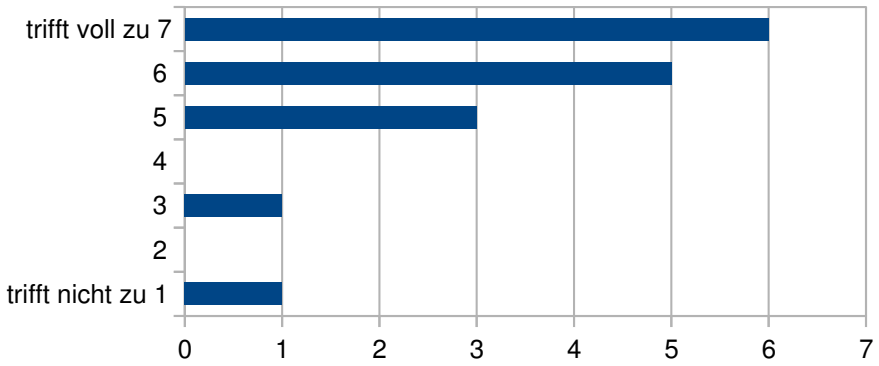

Abbildung 44: „Es wird deutlich wo der Bildschirm berührt worden ist.“

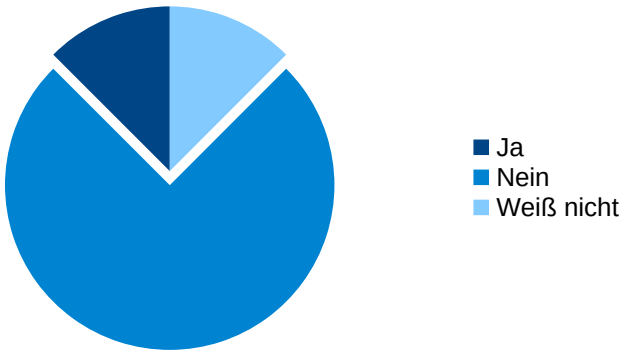

Abbildung 45: „Die *Hand* hat wichtige Bildschirminhalte verdeckt oder davon abgelenkt.“

4.2.2 Grafische Darstellung der Hand

Die grafische Darstellung der Hand wurde von den meisten Probanden als ausreichend präzise bewertet (siehe Abbildung 43). Auf die Frage nach der Präzision antworteten 12 der Probanden mit den zwei höchstmöglichen Werten. Abzüge gab es durch die wabernden Umrisse der *Hand*. Diese wurden mehrfach negativ in den Kommentaren erwähnt. Dadurch verschwimmen teilweise Finger und sind nicht mehr unterscheidbar. Dennoch reicht die Qualität aus, um zum Beispiel richtige Positionen anzeigen zu können. Dies entspricht der Annahme in Hypothese H2.2. Alle Probanden empfanden es als hilfreich zu sehen, wann der Erklärende den Bildschirm berührt. Die farbliche Kennzeichnung der Bildschirmberührung war für die meisten gut zu erkennen (siehe Abbildung 44). Dies unterstützt Hypothese H2.3. Allerdings wurde sich von einigen Anwendern eine deutlichere Kennzeichnung gewünscht. Außerdem war ein weiterer Wunsch das Anzeigen einer Spur, wie bei der eigenen Eingabe. So könnte man Gesten oder Wege im Ganzen sehen und sie leichter nachvollziehen. Allgemein wurde jedoch positiv angemerkt, dass eine einfachere Vermittlung von Informationen und der richtigen Eingabe ermöglicht wird. Durch die Visualisierung wird eine gute Veranschaulichung erreicht, die sich mit der mündlichen Erklärung ergänzt. Es wurde keine störende Latenz festgestellt. Bildschirminhalte waren weiterhin für die meisten genügend erkennbar. Abbildung 45 zeigt, dass nur 2 Probanden das Gefühl hatten, dass die *Hand* vom Bildschirminhalt ablenkt. Diese wünschten sich eine noch höhere Transparenz. Daher kann Hypothese H2.1 als erfüllt angesehen werden, da 75% der Probanden keine störende Verdeckung festgestellt haben.

Auf die Frage, ob die *Hand* eher störend als hilfreich waren, haben alle Probanden mit einem positiven Wert geantwortet (siehe Abbildung 46). Somit haben alle Probanden die *Hand* eher als hilfreich und nicht als störend empfunden. Damit lässt sich H2 allgemein als korrekt bewerten. Funktional reicht die grafische Qualität der *Hand* aus. Anhand von Abbildung 47 sieht man jedoch, dass sich die *Hand* grafisch noch ansprechender gestalten ließe. Der Mittelwert der Antworten liegt bei 5,5. Dies spiegelt sich auch in den Wünschen und Anregungen der Probanden wieder. Eine Möglichkeit die Farbe der *Hand* zu bestimmen wurde vorgeschlagen. Ebenso schönere Effekte für die Darstellung der Berührungen, wie zum Beispiel ein Welleneffekt.

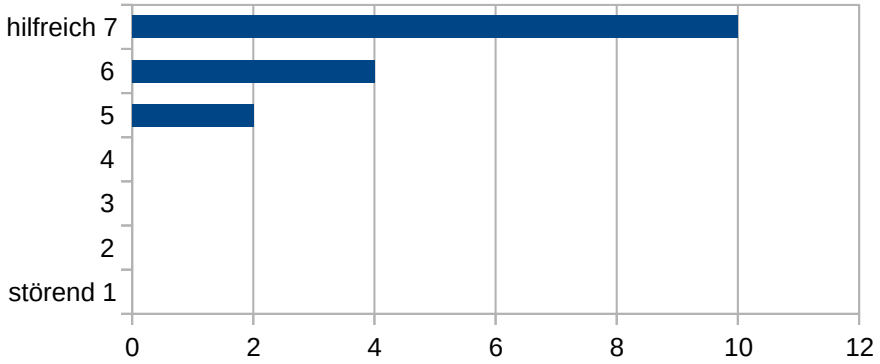

Abbildung 46: „Die *Hand* hat eher geholfen, als mich gestört.“

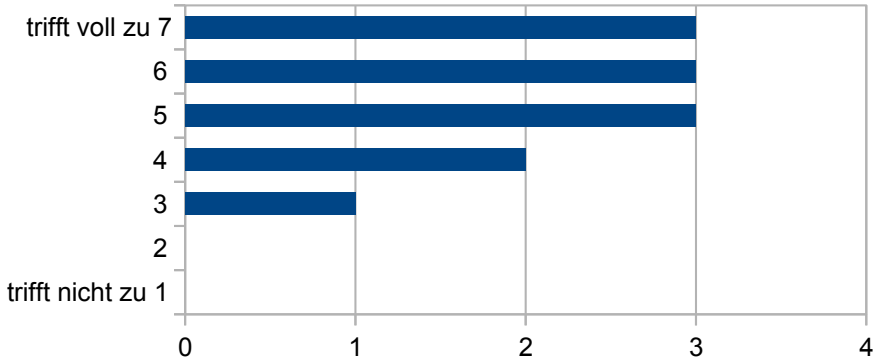

Abbildung 47: „Die Darstellung der Hand war grafisch ansprechend.“

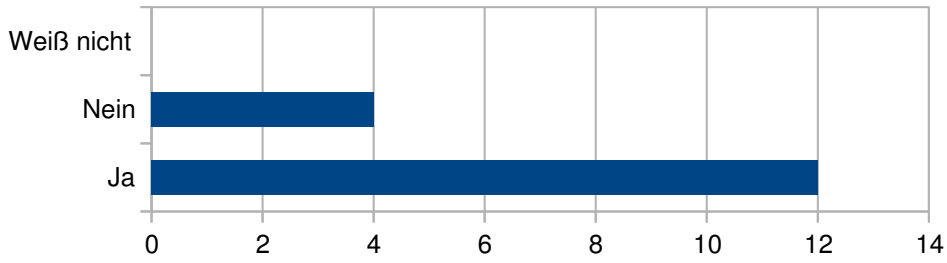

Abbildung 48: „Ich habe schon einmal per Computer mit einer entfernten Person eine ähnliche Problemstellung bearbeitet?“

4.2.3 Mögliche Anwendungsgebiete

Wie in Abbildung 48 zu sehen, haben zu Beginn der Tests 75% der Befragten angegeben, schon einmal selbst eine ähnliche Prblemstellung bearbeitet zu haben. Nach dem sie die Tests als Ausführender durchgeführt hatten, konnten sich 10 der Probanden vorstellen ein solches System *sehr wahrscheinlich* selbst einzusetzen (siehe Abbildung 49). Die Rolle das Erklärenden konnte noch einen weiteren Probanden davon überzeugen, sodass 11 mit *sehr wahrscheinlich* antworteten (siehe Abbildung 50). In beiden Fällen, gab keiner eine Antwort, die schlechter als der Neutralwert war. Auf die Frage nach möglichen Anwendungsgebieten, gaben die Probanden unter anderem die Antworten: *bei der Arbeit*, *Hilfe von unerfahrenen Familienmitgliedern* und *der Support bei der Bedienung von Touchscreen-Systemen*. Die Probanden konnten sich das System auch in kollaborativen Anwendungen vorstellen. So zum Beispiel bei gemeinsamen Softwareprojekten oder Präsentationen. Für den Einsatz in einem professionellen Umfeld wurde beispielsweise Hilfe nach der Einführung eines neues Softwaresystem genannt oder der Einsatz an öffentlichen Fahrkartenautomaten.

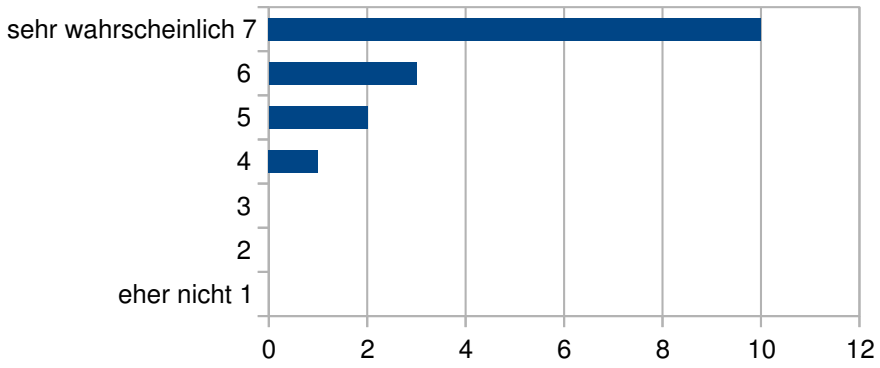

Abbildung 49: „Kannst du dir vorstellen, ein solches System zu nutzen um einer entfernen Person ein Programm zu erklären, einen Weg zu beschreiben oder ähnliches?“, nach Durchgang 1

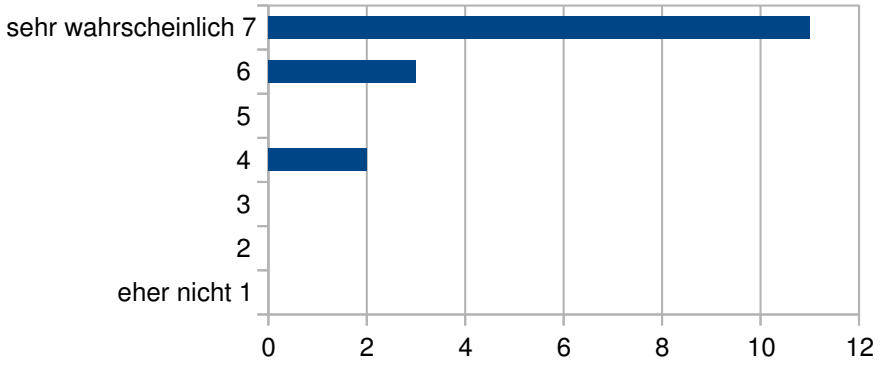

Abbildung 50: „Kannst du dir vorstellen, ein solches System zu nutzen um einer entfernen Person ein Programm zu erklären, einen Weg zu beschreiben oder ähnliches?“, nach Durchgang 2

5 Fazit

5.1 Die Testanwendung

Im Rahmen dieser Arbeit wurde ein Testsystem entwickelt, um zu überprüfen wie sich die Zusammenarbeit zweier Benutzer einer Touchscreen-Anwendung verbessern lässt. Dazu wurde evaluiert, inwiefern die Einblendung der Eingabehand einer der Benutzer einen Mehrwert bringt und ob Informationen effektiver ausgetauscht werden können. Das in Erklärenden und Ausführenden aufgeteilte System führt die Probanden durch eine Reihe verschiedener Testszenarien. Diese wurden für die Bewertung in 3 Typen unterteilt. In den einfachen Anwendungen vom Typ 1, benötigt man grundlegende Bedienkonzepte für die Eingabe an Touchscreen-Systemen. Diese profitieren deutlich von der Nutzung der *Hand*. Bei diesen sah man, das die Vermittlung von zusätzlichem Wissen stärker zur Verbesserung der Ausführungszeiten beiträgt. Die Bedienung an sich, war bereits bekannt. Die komplexeren Anwendungen vom Typ 2 zeigen weniger Verbesserung. Hier war es Aufgabe des Erklärenden, die richtige Bedienung zu vermitteln. Die Lösung mussten die Ausführenden selbst finden. Dadurch hat die eigentliche Erklärung weniger Einfluss auf die Testzeit. Besonders die Aufgaben vom Typ 3, welche auf der Vermittlung von Bildschirminhalten beruhen, konnten deutlich effizienter ausgeführt werden. Obwohl die *Hand* nicht immer effektiv eingesetzt oder sogar vergessen wurde, haben sich insgesamt deutliche Verbesserungen gezeigt. Im Allgemeinen hat die *Hand* bei allen Benutzern für mehr Sicherheit gesorgt. In der Rolle des Ausführenden würden, bei wiederholter Durchführung der Aufgabe, die Einblendung der Hand gewünscht werden. Ebenso würden die Erklärenden die Aufgaben lieber noch einmal mit der *Hand* erklären, als nur verbal. Dieser Wunsch besteht auch in den Fällen, in denen keine messbare Steigerung der Effizienz erreicht wurde. Insgesamt konnte gezeigt werden, dass eine Visualisierung der Hand einen Vorteil für die Effizienz der Bedienung von kollaborativen Touchscreen-Anwendungen bietet.

5.2 Visualisierung der Hand

Als zweiter Schwerpunkt dieser Arbeit wurde eine Visualisierung einer Hand, bei der Eingabe auf Touchscreen-Geräten, erstellt. Dazu wurden Tiefendaten eines Kinect Sensors verwendet. Diese wurden über ein Netzwerk gestreamt, um bei dem Empfänger in mehreren Shaderdurchläufen verarbeitet und angezeigt zu werden. Die *Hand* wurde transparent umgesetzt, um die Verdeckung von Bildschirminhalten zu minimieren. Durch Verwendung eines SSAO-Shaders kann ein plastischer Eindruck erreicht werden. Dieser hilft bei der Bestimmung von Position und Lage der Hand. Die Eingaben werden durch Einfärben der Finger verdeutlicht.
Das größte Problem ist die mangelnde Qualität der Eingabebilder durch den Kinect Sensor. Diese haben eine geringe Auflösung, unsichere Tiefendaten und dadurch entstehende Bildstörungen. Durch die nötigen Glättungen gehen noch weitere Details der Hand verloren. Unter diesen Bedingungen wären andere Verfahren nötig,

um ein genaueres Abbild der Hand zu erhalten. Die Hauptkritikpunkte der Visualisierung waren zum Beispiel wabernde Umrisse, das Verschwimmen von Fingern und nicht genügend Plastizität.
Im Rahmen der Testanwendung wurde evaluiert, ob die grafische Darstellung dazu geeignet ist, in einem kollaborativen Touchscreen-System eingesetzt zu werden. Die Verdeckung des Bildschirminhaltes konnte merklich verringert werden. Fast alle Probanden gaben an, dass die *Hand* keine wichtigen Bildschirminhalte verdeckt oder davon abgelenkt. Die Präzision und der plastische Eindruck waren ausreichend, um Position und Lage gut bestimmen zu können. Durch die Visualisierung der Bildschirmberührung konnten fast alle Nutzer die Eingabe des Erklärenden gut erkennen. Dadurch ist die Grundlage für eine effektive Unterstützung für den Anwender gegeben.
Wünsche der Probanden, wie andere Einstellungen der Transparenz und der Farbe der *Hand*, können leicht umgesetzt werden. Dies ermöglicht außerdem eine Anpassung an die jeweilige Anwendung, um zum Beispiel den Kontrast zur Hintergrundfarbe zu berücksichtigen. Auch die Höhenabhängigkeit der Transparenz kann je nach Wunsch aktiviert werden. Somit können die Präferenzen unterschiedlicher Nutzer berücksichtigt werden. Je nach Schwerpunkt auf Sichtbarkeit der *Hand*, beziehungsweise Verdeckung des Bildschirms, können diese variiert werden.
Insgesamt wurde die *Hand* als hilfreich empfunden. Die Probanden konnten sich außerdem vorstellen die *Hand* auch in anderen Anwendungen zu benutzen. Zum Beispiel, um einer anderen Person die Benutzung eines Touchscreen-Geräts oder Programms zu erklären oder an gemeinsamen Projekten zu arbeiten.
Die grafische Qualität kann jedoch noch verbessert werden. Außerdem kann die Visualisierung ansprechender gestaltet werden, indem andere Effekte hinzugefügt werden. Um die geforderten Kriterien grundlegend zu erfüllen, reichen die umgesetzten Methoden jedoch aus.

6 Ausblick

Die durchgeführten Tests werfen die Frage auf, wie sich das Testsystem in einer Testreihe mit einem einzigen festen Erklärenden verhält. Der Erklärende ist Experte in den einzelnen Aufgaben und kennt diese genau. So können Verzögerungen durch missverstandene Aufgabenstellungen und ineffiziente Nutzung der *Hand* minimiert werden. Diese Herangehensweise würde einer Art Support Center entsprechen, in dem eine geschulte Person Hilfestellung zu bekannten Problemen gibt.
Eine weitere interessante Erweiterung dieses Systems beinhaltet aktive Kollaboration aller Teilnehmer. Alle Benutzer können auf einer gemeinsamen Oberfläche an demselben Problem arbeiten. Die Unterscheidbarkeit der Hände mehrerer Benutzer, das Visualisieren der Berührungen und Arbeiten mit Transparenz spielen in einem System mit mehreren Teilnehmern eine interessante Rolle. Durch mehrere Hände werden die Kriterien der Verdeckung und Unterscheidbarkeit noch wichtiger.
Mit verbesserter Hardware, zum Beispiel der *Kinect for Windows v2* [fW14], könnte eine wesentlich präzisere Visualisierung der Hand erstellt werden. Die höhere Auflösung und genauere Abtastung der Tiefenwerte könnte das Verschmelzen benachbarter Finger verhindern und das Wabern der Kanten verringern. Außerdem könnte eine bessere Beleuchtung erreicht werden, da mehr Informationen über die Details der Hand vorliegen.
Der Visualisierung könnten weitere Effekte hinzugefügt werden. Beispielsweise andere Visualisierungen der Touchpunkte. Sich kreisförmige ausbreitende Wellen und eine Spur für das anhaltende Berühren des Bildschirms sind zwei der Vorschläge von Probanden. Auch aufwändigere Effekte, wie eine wasserähnliche Oberfläche, wären über Shader zu realisieren. Allerdings bliebe zu überprüfen, ob das Hinzufügen weiterer Effekte einen praktischen Nutzen liefert oder nur visuell ansprechender ist. Zu aufwändige oder flächendeckende Effekte könnten störend wirken und den Aspekt der verringerten Verdeckung des Bildschirminhaltes entgegenwirken.
Durch die bestehende weltweite Vernetzung durch das Internet und die preisgünstige Hardware, ist es denkbar ein solches System im alltäglichen Leben einzusetzen.

Abbildungsverzeichnis

Literatur

[Bol09] Klaas Wilhelm Bollhoefer. Microsoft surface und das natural user interface (nui). 2009.

[BS08] Louis Bavoil and Miguel Sainz. Screen space ambient occlusion. *NVIDIA developer information: http://developers. nvidia. com*, 6, 2008.

[Bux07] Bill Buxton. Multi-touch systems that i have known and loved. `http://www.billbuxton.com/multitouchOverview.html`, 2007. Stand 24.06.2014.

[CAHS06] Thomas Coogan, George Awad, Junwei Han, and Alistair Sutherland. Real time hand gesture recognition including hand segmentation and tracking. In *Advances in Visual Computing*, volume 4291 of *Lecture Notes in Computer Science*, pages 495–504. Springer Berlin Heidelberg, 2006.

[Cen] Microsoft News Center. Xbox execs talk momentum and the future of tv. `http://www.microsoft.com/en-us/news/features/2013/feb13/02-11xbox.aspx`. Stand 01.07.2014.

[Dop04] Oleg Dopertchouk. Recognition of Handwritten Gestures. `http://www.gamedev.net/page/resources/_/technical/game-programming/recognition-of-handwritten-gestures-r2039`, 2004. Stand 30.06.2014.

[ES03] George ElKoura and Karan Singh. Handrix: Animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 110–119, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[FK92] Susan R. Fussell and Robert M. Krauss. Coordination of knowledge in communication: Effects of speakers' assumptions about what others know. *Journal of Personality and Social Psychology*, 62(3):378–391, mar 1992.

[FSY+04] Susan R. Fussell, Leslie D. Setlock, Jie Yang, Jiazhi Ou, Elizabeth Mauer, and Adam D. I. Kramer. Gestures over video streams to support remote collaboration on physical tasks. *Hum.-Comput. Interact.*, 19(3):273–309, September 2004.

[fW14] Kinect for Windows. Kinect for windows v2. `http://www.microsoft.com/en-us/kinectforwindows/discover/features.aspx`, 2014. Stand 02.07.2014.

[fWB] Kinect for Windows Blog. Near mode: What it is, and isn't. http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/20/near-mode-what-it-is-and-isn-t.aspx. Stand 29.06.2014.

[Gui] Kivy Users Guide. Kivy faq. http://kivy.org/docs/faq.html. Stand 01.07.2014.

[Gui14] Kivy Programming Guide. Kivy architectural overview. http://kivy.org/docs/guide/architecture.html, 2014. Stand 25.06.2014.

[IAC+07] Shahram Izadi, Ankur Agarwal, Antonio Criminisi, John Winn, Andrew Blake, and Andrew Fitzgibbon. C-slate: A multi-touch and object recognition system for remote collaboration using horizontal surfaces. In *Proceedings of the Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop 2007)*. IEEE, 2007.

[Kaj09] Vladimir Kajalin. Screen space ambient occlusion. *ShaderX7: Advanced Rendering Techniques*, 2009.

[KASA08] Fakhreddine Karray, Milad Alemzadeh, Jamil Abou Saleh, and Mo Nours Arab. Human-computer interaction: Overview on state of the art. *International Journal on Smart Sensing and Intelligent Systems*, 1(1):137–159, mar 2008.

[KE12] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[MIL99] CHRISTOPHER A. MILLER. Bridging the information transfer gap: Measuring goodness of information fit. *Journal of Visual Languages and Computing*, 1999.

[MWW10] Meredith Ringel Morris, Jacob O. Wobbrock, and Andrew D. Wilson. Understanding users' preferences for surface gestures. In *Proceedings of Graphics Interface 2010*, GI '10, pages 261–268, Toronto, Ont., Canada, Canada, 2010. Canadian Information Processing Society.

[Neta] Microsoft Developer Network. Kinect for windows sensor components and specifications. http://msdn.microsoft.com/en-us/library/jj131033.aspx. Stand 01.07.2014.

[Netb] Microsoft Developer Network. Kinect fusion. http://msdn.microsoft.com/en-us/library/dn188670.aspx. Stand 01.07.2014.

[Nor10] Donald A. Norman. Natural user interfaces are not natural. *interactions*, 17(3):6–10, May 2010.

[Pic04] M. Piccardi. Background subtraction techniques: a review. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3099–3104 vol.4, Oct 2004.

[Sch10] Johannes Schoening. Touching the future: the rise of multitouch interfaces. 2010.

[SJF+13] Rajinder S. Sodhi, Brett R. Jones, David Forsyth, Brian P. Bailey, and Giuliano Maciocci. Bethere: 3d mobile collaboration with spatial input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 179–188, New York, NY, USA, 2013. ACM.

[Tay] Taymans. Gstreamer application development manual (1.3.90). `http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/index.html`. Stand 01.07.2014.

[TM90] John C. Tang and Scott L. Minneman. Videodraw: A video interface for collaborative drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 313–320, New York, NY, USA, 1990. ACM.

[uWB] Borns IT und Windows-Blog. Kinect for windows sdk mit neuerungen. `http://www.borncity.com/blog/2013/03/18/kinect-for-windows-sdk-mit-neuerungen/`. Stand 01.07.2014.

[VGF99] Elizabeth S. Veinott, Judith Olson Gary, and Xiaolan Fu. Video helps remote work: Speakers who need to negotiate common ground benefit from seeing each other. In *In Proceedings of CHI*, pages 302–309. ACM Press, 1999.

[Zha12] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–12, April 2012.