Bachelorarbeit im Studiengang
Informationstechnologie und Gestaltung
International

Betriebssystemunabhängiger Touchscreenaufbau auf Basis kostengünstiger Konsumerprodukte

André Schwarz
Matrikel-Nr.: 217495

Fachhochschule Lübeck
Fachbereich Informatik und Elektrotechnik

ma design

ma design GmbH & Co. KG
Abteilung Software Engineering

Betreuender Prüfer: Prof. Dr. Nane Kratzke

Zweitprüfer: Prof. Dr. Monique Janneck

Betreuer ma design: Frank Lindecke

Eidesstattliche Erklärung

Ich versichere, dass ich die Arbeit selbständig, ohne fremde Hilfe verfasst habe.

Bei der Abfassung der Arbeit sind nur die angegebenen Quellen benutzt worden. Wörtlich oder dem Sinne nach entnommene Stellen sind als solche gekennzeichnet.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, insbesondere dass die Arbeit Dritten zur Einsichtnahme vorgelegt oder Kopien der Arbeit zur Weitergabe an Dritte angefertigt werden.

Ort, Datum	Unterschrift

Danksagung

Ich bedanke mich besonders bei meinem betreuenden Professor, Herrn Nane Kratzke, bei meinem Betreuer, Herrn Frank Lindecke aus der Firma ma design und bei meinen dortigen Arbeitskollegen für deren Unterstützung. Desweiteren bedanke ich mich bei Oliver Mader, der mit freundlicherweise die Dokumentenvorlage zur Verfügung gestellt hat, die im Rahmen seiner Bachelorarbeit [Mad12] entstanden ist.

Inhaltsverzeichnis

1 Abbildungsverzeichnis

2 Einleitung

2.1 Hintergrund und Motivation

Die am meisten verwendeten Eingabegeräte technischer Art stellen die berührungsbasierten Geräte wie Maus und Tastatur dar. In dem Bereich der innovativen Eingabegeräte schießen besonders auf Technikmessen vielzählige Produkten hervor. Unter anderem gibt es die Möglichkeit des Touchscreens, der in modernen Smartphones und Laptops bereits Einzug gehalten hat. Es gibt noch die klassische Maus- und Tastatursteuerung, sowie Geräte, die auf Stifteingaben reagieren. Eine neue Sparte dieser technischen Innovationen wird per Touchgeste gesteuert, jedoch nicht auf einem Bildschirm, sondern auf einer Projektion. Diese Geräte haben es bislang noch nicht großflächig auf den Markt geschafft und kosten nach wie vor mehrere Tausend Euro.

Diese Arbeit beschäftigt sich mit der Touchscreen-Eingabe, welche allerdings aus Consumer-Hardware gebaut werden soll, wodurch eine wesentlich geringere Investition benötigt wird.

2.2 Zielsetzung

Das Ziel dieser Arbeit ist es, einen solchen Touchbeamer umzusetzen. Dieses Gerät bedingt einen Aufbau, bei dem das Bildschirmbild eines Computers mithilfe eines Beamers auf eine Fläche projiziert wird. Auf dieser Projektion soll nun eine Touchfunktion umgesetzt werden. Dazu wird ein Gerät benötigt, welches die Hand- und Fingerposition erkennen kann. Die Daten, die dieses Erkennungsgerät liefert, werden interpretiert und mithilfe mathematischer Berechnungen in Aktionen übersetzt. Diese Touchgesten werden zur Steuerung des Computers benutzt und sollen möglichst so genau sein, dass man damit die vergleichbaren Mausaktionen des

Computers durchführen kann. Diese Arbeit beschäftigt sich mit der Analyse, der Implementierung und den Berechnungen und Funktionen des Touchbeamers. Die Ergebnisse sollen in einer Java Applikation umgesetzt werden. Die Software hat das Ziel, keine Nutzer auszuschließen. Daher soll sie Betriebssystemunabhängig umgesetzt werden. Eine solche liegt nicht in Form einer Betriebssystemspezifischen ausführbaren Datei, wie zum Beispiel einer .exe für Windows oder .dmg für Mac vor, sondern in einem Format, welches alle Betriebssysteme starten können. Das hat den Vorteil, dass sie auf allen gängigen Plattformen genutzt werden kann, dazu zählen Windows, Mac und Linux.

2.3 Aufbau

Mithilfe von bekannten Touchgesten entwirft diese Arbeit ein innovatives Interaktionskonzept, basierend auf dem Prinzip des Touchscreens.

Im zweiten Kapitel werden die Grundlagen erläutert, die Hardware Geräte ausgewählt, sowie die nötigen Gegebenheiten erläutert, die dieses Projekt plattformunabhängig machen. Das dritte Kapitel beschäftigt sich mit der Analyse der Anforderungen, sowie wichtigen Informationen, die zum Aufbau des Systems und zur Umsetztung der Applikation dienen. Fernerhin wird die Oberfläche ausgewählt, die als Projektionshintergrund dienen soll. Im vierten Kapitel wird die Implementierung des Touchbeamers beschrieben. Dabei wird auf die eventuellen Probleme und deren Lösungsmöglichkeiten eingegangen. Schließlich werden die Frameworks beschrieben, die zur Umsetzung benötigt wurden. Die Teststrategie und die Nachweisführung bezeichnen den Inhalt des fünften Kapitels.

Das sechste Kapitel zieht ein Fazit aus diesem Projekt.

3 Grundlagen

Dieses Kapitel beschäftigt sich mit den Grundlagen des Touchbeameraufbaus. Im Vordergrund steht die richtige Auswahl des Geräts, welches die Position der Hände erkennen soll. Außerdem wird die Betriebssystemunabhängigkeit analysiert.

Im Bereich der Gestensteuerung war Sony der Vorreiter mit seiner Playstation EyeToy Kamera. Nintendo war der nächste, der sich dieses Prinzip der Steuerung zu Nutzen machte und setzte mit der Wii neue Maßstäbe auf dem Markt der Spielekonsolen. Die Kinect von Microsoft brachte daraufhin die Gestensteuerung auf einen neuen Level, dem Sony den Move Controller entgegen stellte. Das neueste Gerät auf dem Markt, das diese Art der Stererung für den Nutzer bereithält, ist das Leap Motion vom gleichnamigen Herseller. [Kle15]

Für dieses Projekt sind insbesondere das Leap Motion, sowie die Kinect hervorzuheben, da beide sich für dieses Projekt eigenen würden. Das ergibt sich aus dem Umstand, dass für beide Geräte ein SDK (Software Development Kit) vorhanden sind, damit Software-Entwickler auf die Hardware zugreifen können. Desweiteren gelten beide als moderne Geräte, die den neuesten Stand der Technik abbilden.

Die Geräte sind auf dem Consumer Markt erhältlich. Die Kosten für das Leap Motion belaufen sich auf 89,99 €. [1] [Mot15]

Die Kinect kostet aktuell 99 €, [2] welches noch mit dem low-cost Ansatz vereinbar ist. [com10]

Damit liegen beide Geräte im kostengünstigen Bereich und sind für dieses Projekt geeignet.

Nach Möglichkeit sollte hier ein kleines Gerät gewählt werden, welches dennoch den Anforderungen standhält. Durch die geringe Größe wird die Flexibilität des Touchbeamer Aufbaus erhöht. Somit kann er leicht an verschiedenen Stellen aufgebaut werden.

[1] Zeitpunkt des Preises: 02.03.2015
[2] Zeitpunkt des Preises: 02.03.2015

3.1 Anforderungen an die Hardware

Da dieses Projekt einen Aufbau aus Consumerprodukten hervorbringen soll, müssen bei den Anforderungen an die Hardware Einschränkungen gemacht werden. Sie sollte eine Genauigkeit mitbringen, welche es erlaubt, den Computer intuitiv zu steuern und stärkere Schwankungen oder Abweichungen vermeidet.

Ein weitereres Kriterium ist die Größe des Geräts. Da dieser Gesamtaufbau prototypisch erfolgt, spielt dies eher eine untergeordnete Rolle. Es wäre jedoch von Vorteil, wenn die Hardware möglichst klein ist. Auch sollte vermieden werden, dass für die Handerkennung mehr als ein Gerät benötigt wird, um einen übersichtlichen Systemaufbau zu gewährleisten. Außerdem wird so vermieden, dass durch ein Zusammenspiel von mehreren Geräten Ungenauigkeiten aufkommen. Zusätzlich ist noch ein Beamer vonnöten, der jedoch in der Lage sein soll, auf einer geringen Distanz zu projizieren. Natürlich wäre es von Vorteil, wenn auch der Beamer möglichst klein ist, jedoch trotzdem ein ausreichend helles Bild liefert, sodass der Nutzungsraum nicht zu stark abgedunktelt werden muss.

3.2 Kinect

Die Kinect wurde Anfang November 2010 auf den Markt gebracht. Zu diesem Zeitpunkt war sie eine absolute Neuerung. Sie wurde durch Microsoft und besonders im Bezug zur Xbox 360 im großen Stil beworben. Sie hat es erlaubt, mithilfe des gesamten Körpers ein Programm zu steuern, was besonders in der Spiele-Branche eingesetzt wird. Ihre Genauigkeit war zu diesem Zeitpunkt eine Sensation, wodurch die Kinect weltweit bekannt wurde.

Die Größe der Kinect beträgt 24.9cm x 6.6cm x 6.7cm.

Die Kinect ist eine auf Infrarotlicht basierende Kamera. Um einen Körper zu erkennen, wird ein Infrarot-Punktmuster erstellt und mithilfe von Infrarotkameras erkannt. Dadurch kann sie mit den eingebauten Tiefenkameras den relationalen Abstand zweier Objekte bestimmen. Daraus ergibt sich dann die Entfernung zur Kamera und der Abstand zwischen Objekten.

Die Kinect erfasst also mit Hilfe der Tiefensensoren die Umgebung. Die daraus resultierenden Ergebnisse unterliegen -je nach Abstand des Objektes- einer Genauigkeitsschwankung. Die Reichweite der Kinect umfasst einen Bereich von 0,5

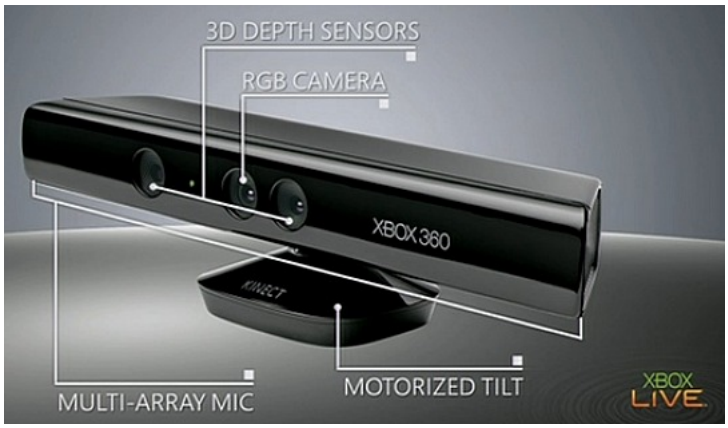


Abbildung 3.1: Darstellung der Kinect inklusive der Bauteile[MRS12]

m bis 5 m, dabei treten Schwankungen von einigen Millimetern bis zu 4 Zentimetern auf.[Cor12]

3.3 Leap Motion

Das Leap Motion ist ein Gerät,[3] das entwickelt wurde, um Bewegungen und Gesten vor dem Bildschirm zu erkennen und somit eine innovative Steuerung zu ermöglichen. Dabei soll die Maus bzw. Tastatur nicht ersetzt, sondern die Interaktionen mit dem Computer erweitern werden.

Abbildung 3.2: Darstellung des Leap Motions [Hon14]

Durch eine Genauigkeit von unter einem Millimeter [WBRF13]

ist das Leap Motion in der Lage, die kleinsten Bewegungen zu erkennen. Der Leap Motion Controller beinhaltet zwei Kameras sowie drei Infrarot LEDs. Durch den

[3]Das Leap Motion hat eine Größe von 7.62 x 3.05 x 1.27 cm

Einbau einer Weitwinkel-Linse ist der Interaktionsraum ungefähr acht Kubik Fuß (entspricht ca. 0,23 m^3) groß; dies entspricht einem Raum von 60 x 60 x 60 cm. Die Reichweite ändert sich, je nachdem wie viel Strom der Leap Controller durch den USB-Port bekommt.

Die Leap Software erstellt aus den Sensor-Rohdaten ein 3D Bild. Dazu ist ein komplizierter Algorithmus vonnöten, der ein Geheimnis der Herstellerfirma bleibt. Für das Tracking werden die Informationen nach Fingern oder Werkzeugen interpretiert und die Position festgestellt.

Die folgenden Strukturen sind in der API erkennbar. Dabei wird als gröbste Struktur der Unterarm erkannt. Die API registriert die Position des Ellbogens und die Richtung, in die der Arm zeigt. Des Weiteren wird die Hand erkannt, welche dem jeweiligen Arm zugeordnet werden kann. Die Hand wiederum wird in die einzelnen Finger aufgesplittet. Jeder Finger besteht aus vier einzelnen Knochen, welche gesondert erkannt und gezielt im Programm angesprochen werden können. In Abbildung 3.3

ist zu erkennen, welche Knochen das Leap Motion identifizieren kann. Das äußerste Fingerglied ist der „Distal phalanges“, der wichtigste Teil für den Touchbeamer. Der Punkt, an dem die Touchgeste stattfindet, kann zunächst durch die Erkennung der Hand und später die des Fingers (z.B. Zeigefinger = Indexfinger), ermittelt werden. Die folgenden Fingerglieder sind die „Intermediate phalanges“, “Proximal phalanges“ sowie die „Metacarpals“. Der Daumen besteht lediglich aus den ersten drei Gliedern.

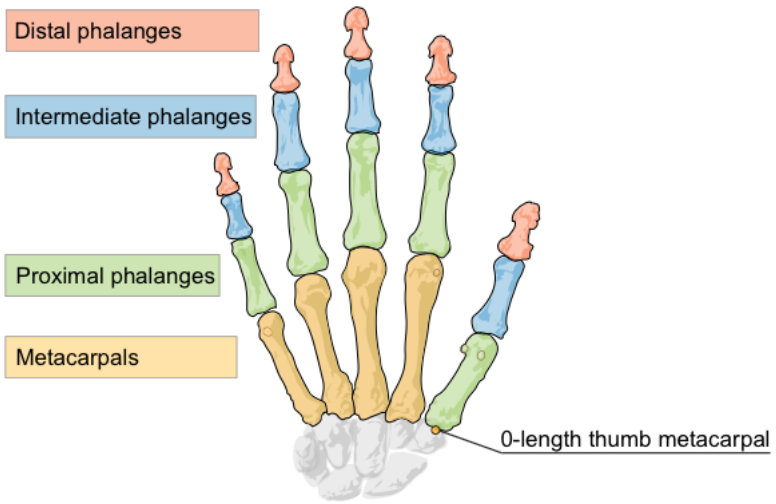


Abbildung 3.3: Übersicht über einzelne erkennbare Handknochen [Dev15b]

Diese genaue Erkennung der Hand und deren Knochengliedern erlaubt eine sehr genaue Positionsbestimmung. Das Leap Motion wird nur dann ungenau, wenn zwei oder mehrere Finger übereinander liegen. Zusätzlich kommt noch die Position des Leap Motions innerhalb des Aufbaus zum Tragen. Die Positionsbestimmung der Hand ist nicht möglich, wenn das Leap Motion die Hand nicht von „unten" sieht, also die Handfläche nicht zum Sensor gerichtet ist. Dieser Punkt muss beim Systemaufbau berücksichtigt werden.

3.4 Beamer

Da bei diesem Projekt eine Beamerprojektion als Touchfläche genutzt wird, ist es wichtig, dass bei der Wahl des Beamers einige Punkte beachtet werden. Dieses System wird durch den Low Cost-Ansatz getrieben, daher sollte ein Beamer ausgewählt werden, der bei möglichst geringem Kosteneinsatz die beste Leistung erzielt.

Der Beamer soll ein Bild darstellen können, welches in den Interaktionsraum des Leap Motion passt. Daher wird ein Gerät gewählt, welches ein kleines Bild darstellen kann und dabei einen möglichst geringen Abstand zur Leinwand benötigt. Desweiteren ist es von Vorteil, wenn der Beamer über einen integrierten Akku verfügt, um den Touchbeameraufbau flexibel aufzubauen. Eine weitere Eigenschaft, die der Beamer besitzen sollte, ist ein ausreichende Helligkeit. Diese muss hoch genug sein, damit die Projektion auch in einem nicht abgedunkelten Raum gut sichtbar ist.

Eine grobe Eingrenzung der angebotenen Beamer erfolgte durch die Festlegung auf die Produktpalette der Pico-Beamer. Diese Geräte sind speziell für kleine Projektionen konstruiert worden. Die Größe von Beamern dieser Art ist gegenüber Geräten in Normalgröße um ein Vielfaches kleiner, was eine große Rolle im Verhältnis zum Gewicht spielt. Die Pico-Beamer weisen meist ein Gewicht von unter 1000 g auf. Dadurch wird die Flexibilität bei der Gestaltung des Systemaufbaus verbessert.

Die Geräte der PicoPix PPX-Reihe von Philips boten eine gute Variation im Preis-Leistungsverhältnis. Der PicoPix PPX2055 ist der kleinste zur Auswahl stehende Beamer. Dieser hat eine Helligkeit von 55 Lumen und wird über die USB-Stromversorgung des Computers betrieben. Dieser Beamer ist mit einem aktuellen

Preis von 189,99 €[4] der am günstigsten zur Auswahl stehende. [Phi15a]

Eine Klasse besser ist der PicoPix PPX2480. Er verfügt über 80 Lumen und einen integriertem Akku. Der aktuelle Preis beläuft sich auf 244,99 €[5], damit liegt er im mittleren Preissegment. [Phi15b]

Der beste Beamer, der noch im Rahmen des low-cost-Ansatzes möglich ist, ist der PicoPix PPX. Mit 140 Lumen ist dieser der Hellste, bei einem aktuellen Preis von 449,99 € [6] aber auch der teuerste. [Phi15c]

Wenn man nun die Beamer vergleicht, fällt auf, dass der PicoPix PPX2055 nicht den Anforderungen entspricht, da er keinen eigenen Akku besitzt. Außerdem erreicht er nicht die geforderte Helligkeit, da der Nutzungsraum sehr stark abgedunkelt werden muss.

Die beiden anderen zur Auswahl stehenden Beamer, entsprechen den Anforderungen. Sie sind hell genug, damit der Nutzer den Aufbau auch in einem nicht so stark abgedunkelten Raum nutzen kann.

Um dem low-cost-Ansatz bestmöglich zu erfüllen, wird der Beamer im mittleren Preissegment gewählt. Der PicoPix PPX2480 ist ein guter Kompromiss zwischen Helligkeit und Preis.

3.5 Geräteauswahl

Um nun das passende Gerät für dieses Projekt auszuwählen, werden die Spezifikationen verglichen.

Die Kinect wurde konstruiert, um den gesamten Körper zu erkennen und dessen Bewegungen wahrzunehmen. Die dabei auftretenden Genauigkeitsschwankungen sind bei einem Abstand von einigen Metern erstaunlich gering.

Das Leap Motion hat eine wesentlich geringere Reichweite. Die dabei auftretenden Ungenauigkeiten liegen bei dem Gerät im ein hundertstel Millimeterbereich. Für den Touchbeamer Aufbau wird ein geringer Abstand benötigt, dieser beläuft sich auf 10 bis 50 cm. Auf diese Distanz erkennt die Kinect die Hand nicht einwandfrei. Daher wird für den Aspekt Genauigkeit das Leap Motion gewählt. [WBRF13]

[4] Zeitpunkt des Preises: 02.03.2015
[5] Zeitpunkt des Preises: 02.03.2015
[6] Zeitpunkt des Preises: 02.03.2015

Ein weiterer wichtiger Punkt ist die Größe des Geräts. Das Leap Motion ist um ein Vielfaches kleiner als die Kinect. Dadurch ergibt sich ein übersichtlicherer Aufbau des Systems. Die Wahl für das passende Gerät fällt daher auf das Leap Motion.

3.6 OS-Unabhängigkeit

Da der Touchbeameraufbau unter anderem das Ziel hat OS-Unabhängig (Betriebssystemunabhängig) zu sein, muss dieser Aspekt vor der Implementierung geplant werden. Um das umzusetzen, muss eine geeignete Programmiersprache gewählt werden. Der folgende Abschnitt beschäftigt sich mit Java und dem SDK.

3.6.1 Programmiersprache Java

Eine entscheidende Anforderung an das Projekt ist, dass das Programm auf Windows, Mac und Linux lauffähig sein soll. Es muss daher eine Programmiersprache gewählt werden, dessen Code sich auf jeder Plattform compilieren und damit auch ausführen lässt. Jede Sprache hat ihre Vor- und ihre Nachteile, daher gibt es keine perfekte Wahl. Java bietet jedoch den entscheidenden Vorteil, dass die Plattformunabhängigkeit leicht umzusetzen ist. Diese ergibt sich aus dem Vorgang, der aus dem Quellcode ein lauffähiges Programm macht. Der geschriebene Java-Quellcode wird in einen Maschinencode, den Bytecode umgewandelt. Dieser Bytecode kann nun von einem Interpreter genutzt werden, um das Programm auszuführen.

Java arbeitet mit einer virtuellen Maschine. Diese virtuelle Maschine selbst ist nicht plattformunabhängig, sondern ein natives Programm, welches auf dem Zielrechner installiert werden muss. Hier befindet sich der Interpreter für die verschiedenen Plattformen, welche alle den selben Bytecode interpretieren können. Dadurch entsteht die Plattformunabhängigkeit von Java. Der Quellcode muss also nur einmal kompiliert werden und kann dann auf Windows, Mac und Linux-Systemen genutzt werden.

3.6.2 Java Standard: AWT

Das Abstract Window Toolkit (AWT) ist ein Bestandteil der Java Foundation Classes, welche eine Ansammlung von Schnittstellen für die grafische Oberflächen-Programmierung mit Java darstellt. Dieses Toolkit stellt eine Standard API zur

Verfügung, mit der eine grafische Benutzeroberfläche geschaffen werden kann. Der Vorteil dabei liegt in der Plattformunabhängigkeit von AWT. Dies wurde speziell dafür entwickelt, dass eine Oberfläche auf allen gängigen Systemen angezeigt werden kann. Die Unabhängigkeit wird dadurch erreicht, dass alle komplexeren Elemente, die nicht auf jeder Plattform dargestellt werden können, keineswegs mit in das Toolkit aufgenommen wurden. Dadurch wird AWT zum kleinsten gemeinsamen Nenner und kann auf allen Plattformen verwendet werden.

3.6.3 Java AWT Robot

Der Robot ist eine Klasse, die vom AWT-Paket zur Verfügung gestellt wird, um automatische Maus- und Tastaturaktionen zu realisieren. Der Robot wird in Form einer Ereigniswarteschlange erstellt, deren Aktionen nacheinander ausgeführt werden sollen. Ferner kann er die Maus zu bestimmten Positionen bewegen und dort verschiedene Mausklicks ausführen. Es ist also das Objekt, welches den Klick ermöglicht und die Touchgeste in externe Anwendungen überträgt. Da die Robot-Klasse aus dem AWT-Paket stammt, ist auch diese plattformunabhängig.

3.6.4 Leap Motion SDK

Das Leap Motion SDK kann für verschiedene Plattformen eingebunden werden. Es stellt native Bibliotheken für Windows, Mac und Linux bereit. Diese können alle in den Kompilierungsvorgang integriert werden, sodass man auf jeder Plattform das System nutzen kann.

4 Analyse

Auf dem Markt gibt es einige Lösungen, um eine Projektion mithilfe von Touchgesten zu steuern. Systeme wie Microsoft Pixelsense [Mic15]

können schnell mehrere Tausend Euro kosten. Diese Arbeit beschäftigt sich mit einer Lösung, die mit Consumer Hardware aufgebaut wird, jedoch das Bestmöglichste aus dieser Hardware herausholt. Dabei sollen die Grenzen im Bezug auf die Genauigkeit ausgelotet werden.

Das folgende Kapitel beschäftigt sich mit der Analyse der Anforderungen. Dabei werden die Produkt- und Software-Anforderungen beschrieben und einander zugeordnet. Desweiteren wird der Projektionshintergrund analysiert und anhand festgelegter Kriterien ausgewählt.

4.1 Anforderungsanalyse

Um ein Produkt zu erstellen, welches sinngemäß nutzbar ist, müssen folgende Produktfunktionen und Software-Anforderungen erfüllt werden. Dieser Abschnitt analysiert und spezifiziert die Anforderungen und erstellt eine übersichtliche Zuordnung.

4.1.1 Produkt-Anforderungen

Der Kern des Touchbeamers besteht aus einer Java-Applikation, die aus dem Hintergrund heraus agiert. Für den Nutzer geht es darum, mit Touchgesten externe Anwendungen zu steuern, die im Vordergrund arbeiten. Dabei soll der Bildschirm des Computers mit einem Beamer angezeigt werden. Zusammen mit der Projektion

wird für den Nutzer die Möglichkeit geschaffen, ein Touchscreen-Erlebnis zu erzeugen. Um dies zu ermöglichen, muss die Fläche, die der Nutzer festlegt, kalibriert werden.

Nach diesem Vorgang soll ein Klick nach bestimmten Kriterien ausgelöst werden. Damit der Nutzer mit dieser Steuerung produktiv arbeiten kann, muss der Klick äußerst genau erkannt werden. Seine Position soll dann so berechnet werden, dass er auf dem Bildschirm angezeigt und ausgeführt werden kann.

Da das Produkt plattformunabhängig sein soll, muss darauf geachtet werden, dass das Produkt auf Windows-, Mac- und Linux-Systemen nutzbar ist. Aus dieser Aufgabenstellung ergeben sich folgende Produktfunktionen.

PF 1 Daten vom Leap Motion auslesen

PF 2 Kalibrieren

PF 3 Klick erkennen

PF 4 Steuerung durchführen

PF 5 OS Unabhängigkeit

4.1.2 Software-Anforderungen

Die Software bildet den funktionalen Hintergrund des Produktes. Ihre Aufgabe ist es, alle anfallenden Berechnungen auszuführen, um den Touchbeamer möglichst exakt arbeiten zu lassen. Beim Start soll die Software eine Benutzeroberfläche zur Verfügung stellen, mithilfe derer sie kalibriert werden kann. Dies ist nötig, da die Position des Beamers und des Leap Motion immer unterschiedlich sein kann. Bei der Kalibrierung werden alle Eckpunkte der Projektion ausgelesen und gespeichert.

Nach der Validierung der Ebene kann die Hauptaufgabe der Software, die Erkennung einer Touchgeste gestartet werden. Um diese Aufgabe durchführen zu können, muss die Software die Daten filtern, die das Leap Motion für jeden Frame sendet. Dabei durchsucht sie für jeden Frame die Strukturen des Arms. Die Untersuchung fängt bei jeder Hand an, die in dem Erkennungsbereich des Leap Motion liegt. Danach überprüft die Software die Finger, wobei nur der Zeigefinger relevant ist. Sollte dieser Finger erkannt werden, wird die Position der Fingerspitze ausgelesen.

Der nächste Schritt ist dann die Berechnung der Entfernung zwischen der Fingerspitze und der kalibrierten Ebene. Dadurch kann berechnet werden, ob sich ein Finger innerhalb des Touchraums befindet. Sollte ein Finger in der Nähe der Projektionsfläche registriert werden, muss unterschieden werden, welche Aktion vom Nutzer durchgeführt wurde. Dieser bewegt zum Klicken seinen Finger in Richtung der Ebene, dort kann er jedoch bleiben und verschoben werden. Dadurch wird kein vollständiger Klick ausgeführt, sondern nur ein Drücken der Maustaste. Dadurch hat der Nutzer die Möglichkeit, den Verschiebe-Modus zu nutzen. Während der Verschiebung bleibt die Maustaste gedrückt und wird erst dann losgelassen, wenn der Finger den Touchbereich verlassen hat.

Ein Klick ist genau genommen nur eine Annäherung der Zeigefingerspitze an die Ebene. Um, trotz geringer Abweichungen in der Genauigkeit beim Kalibrieren, das Produkt nutzen zu können, muss auch bei der Klickerkennung eine Toleranz vorhanden sein. Daher liegt der Finger bereits im Touchbereich, obwohl er sich noch vor oder hinter der Ebene befindet.

Mit den Koordinaten der Fingerspitze wird der genaue Punkt auf der Ebene berechnet, damit wird die vorher gewählte Toleranz wieder ausgeglichen. Mit diesen Berechnungen soll der Klickpunkt berechnet werden. Die Koordinaten dieses Punktes beschreiben jedoch immer noch einen Punkt im dreidimensionalen Raum, ausgehend vom Leap Motion. Diese Koordinaten müssen so umgerechnet werden, dass sie einem konkreten Punkt im Koordinatensystem in Pixeln des Bildschirms entsprechen. Dadurch kann die Position des Mauszeigers bestimmt werden. Diese wird nun nach außen hin weitergegeben, damit die Mausaktionen durchgeführt werden können. Da die Software plattformunabhängig sein soll, muss die Erkennung der Hand und das Durchführen der Aktionen auf einem Windows Computer, einem Mac und einem Linux-System durchgeführt werden können.

Aus diesen festgelegten Funktionen, ergeben sich folgende Anforderungen an die Software:

[Req. 1] Programm muss im Hintergrund laufen

[Req. 2] GUI zum Kalibrieren

[Req. 3] Eckpunkte des projizierten Bildes speichern

[Req. 4] Ebene aufspannen

[Req. 5] Ebene validieren

[Req. 6] Leap Daten filtern

[Req. 7] Erkennung der Zeigefinger

[Req. 8] Erkennung der Knochenspitze (Intermediate phalanges)

[Req. 9] Erkennen der Position der Zeigefingerspitze

[Req. 10] Erkennen, ob sich der Finger im Touchbereich befindet

[Req. 11] Unterscheidung Klicken / Verschieben

[Req. 12] Verhältnis Bildschirmauflösung / Leap Bereich berechnen

[Req. 13] Weiterreichen der Geste in externe Anwendung

[Req. 14] Gestenverarbeitung auf verschiedene Plattformen übertragen (Windows, Mac, Linux)

4.1.3 Zuordnung

In der folgenden Tabelle werden die Software-Anforderungen den Produkt-Anforderungen zugeordnet, wodurch ersichtlich wird, welche jeweiligen Anforderungen benötigt werden. Hier sind im Besonderen PF 2 das Kalibrieren und PF 4 die Auswertung des Klicks zu nennen. Dies sind die wichtigsten Anforderungen, ohne die das Produkt nicht nutzbar wäre.

PF	1	2	3	4	5
Req. 1	X				
Req. 2		X			
Req. 3		X			
Req. 4		X			
Req. 5		X			
Req. 6			X		
Req. 7			X		
Req. 8			X		
Req. 9			X		
Req. 10			X		
Req. 11				X	
Req. 12				X	
Req. 13				X	
Req. 14					X

4.2 Kalibrieren der Software

Das Kalibrieren ist die erste Interaktion die der Nutzer mit dem Touchbeamer macht. Hier ist es besonders wichtig, dass diese so genau wie möglich arbeitet. Die Kalibrierung ist nötig, da die Projektionsfläche nicht festgelegt ist und sich die Position ändern kann. Dadurch wird die Position der Fläche eindeutig festgelegt. Um die Software zu kalibrieren, werden die vier Eckpunkte der Projektionsfläche eingelesen und als Vektoren gespeichert. Dazu wird jeweils ein Eckpunkt berührt und dann der jeweilige Button auf dem Graphical User Interface (GUI) geklickt.

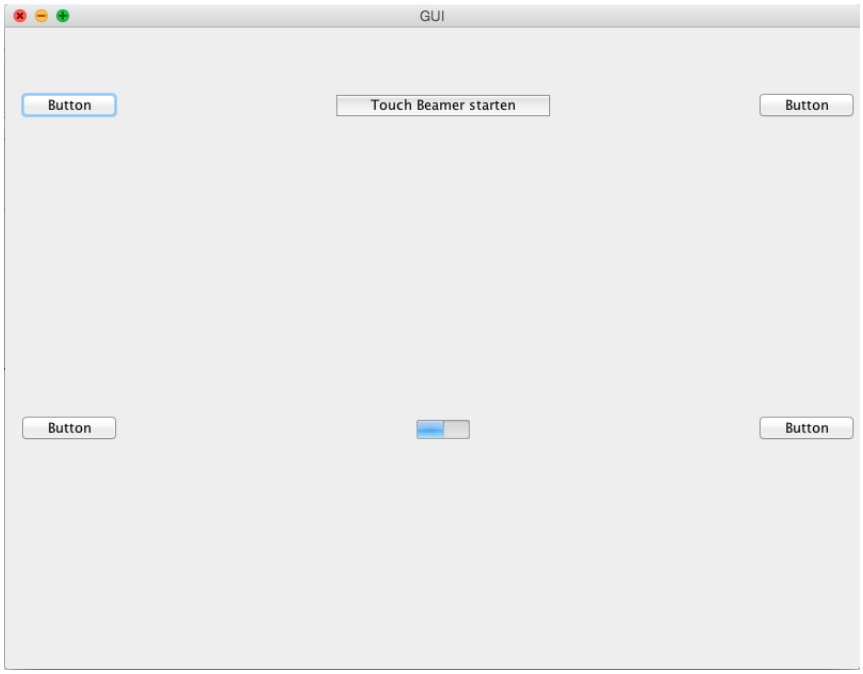


Abbildung 4.1: GUI zum Kalibrieren der Software

Sobald eine Ecke kalibriert wurde, werden die Koordinaten auf der GUI angezeigt, dadurch kann der Nutzer sehen, dass im Datenspeicher ein Vektor hinterlegt wurde. Dieser Vorgang wird für jeden der vier Eckpunkte durchgeführt.

Der Koordinatenursprung der Vektoren liegt zentral auf dem Leap Motion. Wie die X-, Y- und Z-Achse aufgebaut sind, wird in folgender Abbildung gezeigt.

Mithilfe der ersten drei dieser Eckpunkte kann innerhalb der Software eine Fläche erstellt werden. Dazu werden die linke obere und untere, sowie die obere rechte Ecke genutzt. Die Ebene wird ebenso wie die Punkte als Vektorebene gespeichert. Danach kann überprüft werden, ob die Ebene valide ist. Durch kleine Schwankungen bei der Erkennung und Lokalisierung der Hand kann es passieren, dass die Ebene falsch aufgespannt wird. Dadurch wird die Ebene nicht an der Position

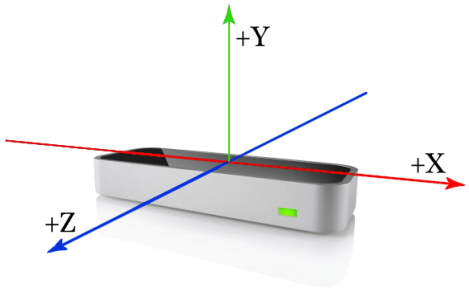


Abbildung 4.2: Koordinatensystem des Leap Motion Hardware Controllers [Dev15a]

erstellt, an der sich die Projektion befindet. Um diese Abweichung zu erkennen, wird der vierte erstellte Punkt verwendet, um die Ebene zu überprüfen. Da der vierte Punkt auch auf der Projektion liegt, muss er sich optimalerweise in der Ebene oder in geringem Abstand zur Ebene befinden. Die Distanz zwischen dem Punkt und der Ebene muss geringer als die Toleranzgrenze sein. Dadurch werden geringe Schwankungen ignoriert, die es sonst erheblich erschweren würden eine korrekte Ebene zu erstellen. Da das Leap Motion eine Genauigkeit von unter einem Zentimeter erreicht, ist auch die Ebene in diesem Genauigkeitsbereich. Jede Schwankung die auftritt, würde dadurch die Funktionalität beeinträchtigen. Daher wird eine Toleranzgrenze von 20 Millimetern gewählt.

Sobald dieses Kriterium erfüllt ist, ist die Ebene valide und schließt dadurch den Kalibrierungsvorgang ab.

4.3 Projektionsfläche

Da der Touchbeameraufbau die Touchgeste nicht auf einem herkömmlichen Bildschirm, sondern auf einer eigenen Oberfläche umsetzt, muss die Wahl des richtigen Projektionshintergrundes umsichtig geplant sein. Der folgende Abschnitt beschäftigt sich mit dessen Auswahl und beschreibt den Entscheidungsvorgang für eine Oberfläche. Dabei werden zunächst einige Hintergründe, mit dem Kriterium die bestmögliche Erkennung mit dem Leap Motion zu liefern, überprüft. Anschließend wurden diese Oberflächen im Zusammenhang mit dem Projekt getestet. Um die Vorteilhafteste für den Touchbeamer zu wählen, müssen beide Ergebnisse verglichen und miteinander kombiniert werden.

Die Versuche werden mithilfe des von Leap Motion bereit gestellten Programms durchgeführt. Dieses bietet die Möglichkeit, den Visualizer zu nutzen, einem Tool,

um das Skelett der erkannten Hand in einem Koordinatensystem darzustellen. Damit ist es möglich, einzelne Finger und Knochen genau zu beobachten.

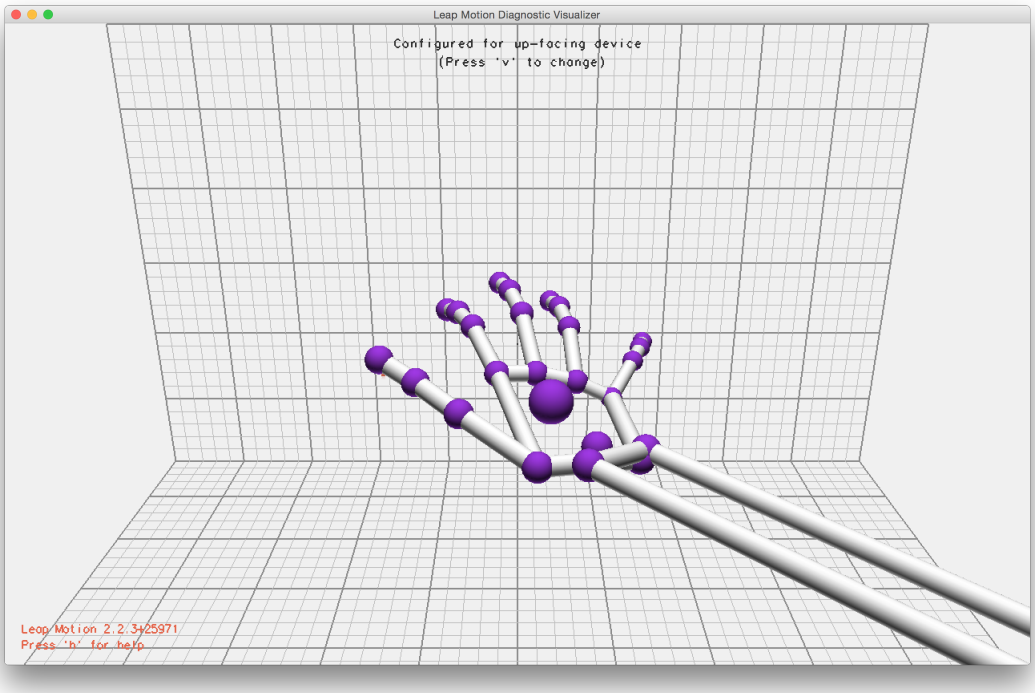


Abbildung 4.3: Visualizer des Leap Motion

Desweiteren bietet es die Möglichkeit, Bilder aus der Perspektive beider Leap Motion-Kameras zu sehen. Diese werden im Infrarotbereich aufgenommen und in Graustufen angezeigt.

4.3.1 Kriterien für das Leap Motion

Das Leap Motion wurde entwickelt, um eine Gestensteuerung des Computers zu ermöglichen, die frei vor dem Bildschirm stattfindet. Der Touchbeameraufbau benutzt das Leap Motion allerdings, um eine Touchgeste umzusetzen, die direkt auf einer Oberfläche stattfindet. In diesem Abschnitt wird lediglich darauf eingegangen, wie sich das Leap Motion vor diesem Hintergrund verhält. Dazu wurden drei Kriterien aufgestellt, die dieses Verhalten beschreiben.

Das für den Einsatzzweck am weitesten entfernte Kriterium ist die Handerkennung im Abstand von 10 cm. Der Versuch mit diesem Kriterium wurde mit einer still gehaltenen Hand durchgeführt, dabei kommt er dem eigentlichen Nutzungszweck des Leap Motion am nächsten. Dieser Umstand spiegelt sich auch am Ergebnis

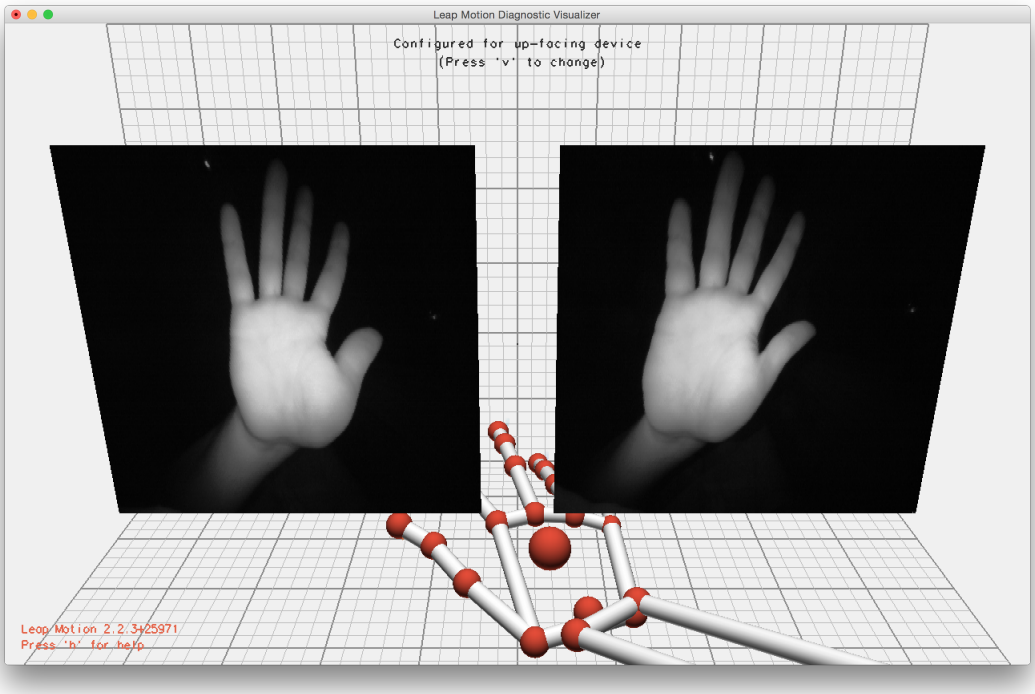


Abbildung 4.4: Perspektive des Leap Motion im Infrarotbereich

dieses Versuchs wider. Keine der 15 Versuchsoberflächen hat eine schlechtere Bewertung als eine 3 bekommen. Besonders hervorzuheben sind hier die Leinwand, die ehemals für die Diaprojektion genutzt wurde, sowie der Fernseher. Diese beiden Hintergründe erzeugen keine Schwankungen bei der Erkennung der Hand. Dadurch stellen sie eine sehr gute Oberfläche aus Sicht des Leap Motion dar. Verschiedene Holz- und Stoffarten sowie Folien und Fliesen weisen leichte Schwankungen auf. Diese äußern sich durch ein leichtes Zittern bei der Handerkennung, welches man im Visualizer sehr gut beobachten kann. Die Projektionsflächen aus Filz und lackiertem Holz haben stärkere Schwankungen hervorgerufen. Dadurch wird eine Nutzung für das Projekt erschwert.

Bei dem zweiten Versuch, mit dem Kriterium Handerkennung im Abstand von einem Zentimeter, fällt das Ergebnis negativer aus. Fast jede Oberfläche hat bei diesem Versuch eine schlechteres Verhalten gezeigt. Trotzdem sind die Dia-Leinwand und der Fernseher unter den vorteilhaftesten Oberflächen vertreten. Außerdem sind noch die Fotofarbfolie und das raue Metall geeignet. Die restlichen Oberflächen erzeugen bei der Erkennung starke Schwankungen oder Verkrümmungen der erkannten Hand. Dabei handelt es sich um unnormal verdrehte Finger. Der Filzstoff hat bei diesem Versuch am ungünstigsten abgeschnitten. Sobald die Hand dicht an diesen Stoff gehalten wurde, ist sie komplett verschwunden bzw. nur noch

sporadisch aufgetaucht.

Die Bewegung vor der Projektionsfläche ist essentiell wichtig für dieses Projekt, daher wurde auch dieser Punkt getestet. Dabei wurde in verschiedenen Abständen die Hand vor der Oberfläche bewegt. Das gewünschte Verhalten zeigt sich dabei in einer ruhigen gleichmäßigen Bewegung, die nicht zittert oder Abweichungen aufweist. Die Bewertung ist ähnlich zu dem vorigen Versuch, einzig der Filzstoff zeigte hier ein besseres Verhalten.

Aus Sicht des Leap Motion zeigt sich, dass die Nutzung der Dia-Leinwand und des Fernsehers zu bevorzugen sind. Beide zeigen eine sehr gute bis gute Erkennung der Hand. Außerdem überzeugen sie bei der Erkennung während einer Bewegung.

Der Stoff Duvetyne [1], ist ein Material, das eine äußerst geringe Reflexion aufweist. Dadurch kann es sehr wirkungsvoll als Hintergrund eingesetzt werden. Die Hand wird ohne Probleme erkannt und auch Bewegungen ohne Schwankungen registriert. Der Stoff weist eine derart hohe Opazität auf, dass fast sämtliches Licht blockiert wird. Durch diesen Umstand ist er der am besten geeignete Hintergrund für diesen Aufbau.

Oberfläche	Handerkennung 10cm	Handerkennung 1cm	Bewegung
Duvetyne	1*	1*	1*
Dia-Leinwand	1	2	2
Fernseher	1	2	2
Fotofarbfolie	2	2	2
raues Metall	2	2	2
Frischhaltefolie	2	3	3
Naturholz	2	3	3
moderne Leinwand	2	3	3
schwarzer Stoff	2	3	3
Fliesen	2	4	4
Glas	2	4	4
weißer Stoff	2	4	4
weiße Tapete	2	4	4
weiß lackiertes Holz	3	4	4
Filz	3	5	4

1 = keine Schwankungen

[1] http://en.wikipedia.org/wiki/Duvetyne

2 = leichte Schwankungen

3 = starke Schwankungen

4 = verkrümmte Hand

5 = keine durchgängige Handerkennung

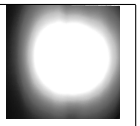
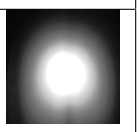
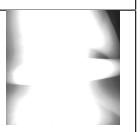
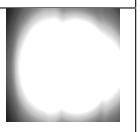
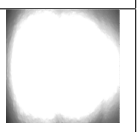
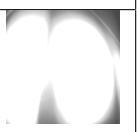
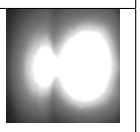
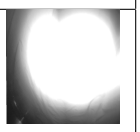
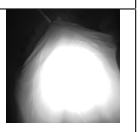
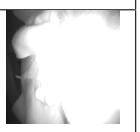
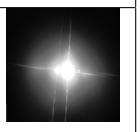
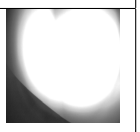
4.3.2 Kriterien des Beameraufbaus

Die ersten Versuche haben die Oberflächen allein aus Sicht des Leap Motion begutachtet. In diesem Abschnitt werden die Oberflächen im Bezug zum Gesamtprojekt getestet.

Der erste Versuch zeigt, ob die Oberflächen für den Einsatz als Projektionshintergrund des Beamers nutzbar sind. Dazu wurde auf alle Oberflächen dasselbe Bild projiziert. Dessen Qualität wurde anhand einer Bewertungsskala benotet. Diese umfasst vier Punkte, welche von sehr geeignet bis schlecht geeignet reichen. Die Bewertung wurde in die unten stehende Tabelle eingefügt. Besonders hervorzuheben ist hier die moderne Leinwand, welche die maximale Qualität, je nach Beamer, widerspiegelt. Gut geeignet sind ebenfalls weiße Stoffe, weiß lackiertes Holz sowie eine Dia-Projektionsleinwand. Nicht geeignet hingegen sind Glas, der Fernseher und die Fotofarbfolie. Bei diesen Materialien ist das Beamerbild nur sehr schwer oder gar nicht zu erkennen.

Der zweite Versuch in diesem Abschnitt beschäftigt sich mit der Infrarot-Reflexion. Um ein gutes Bild mit einem Beamer zu erhalten, ist es hilfreich, wenn die Projektionsfläche reflektiert. Wie sich dieses Verhalten auf das Leap Motion auswirkt zeigt dieser Versuch. Dazu wurde mittels Visualizer (Abbildung 4.3) ein Bild der Oberflächen im Infrarotbereich aufgenommen. Diese Bilder wurden in unten stehender Tabelle den Oberflächen zugeordnet.

Duvetyne wurde in der ersten Versuchsreihe als besonders passend dargestellt. In der zweiten Versuchsreihe wurde aber klar, dass dieser Stoff nicht geeignet war, um ein Beamerbild darauf zu projizieren. Aufgrund seiner hohen Absorption von Licht war das Bild kaum erkennbar.

Oberfläche	geeignet für Beamer	Infrarot Reflexion	
moderne Leinwand	1		
Dia-Leinwand	2		
weißer Stoff	2		
weiß lackiertes Holz	2		
weiße Tapete	2		
Fliesen	2		
Metall	3		
Frischhaltefolie	3		
Naturholz	3		
schwarzer Stoff	3		
Filz	3		
Fernseher	4		
Fotofarbfolie	4		
Duvetyne	4		
Glas	4		

1=sehr geeignet

2=gut geeignet

3=mäßig geeignet

4=schlecht geeignet

4.3.3 Auswahl der Oberfläche

Um die vorteilhafteste Oberfläche auszuwählen, müssen beide Versuchsreihen miteinander verglichen werden. Dabei erfüllt die am besten geeignete einen Kompromiss aus der Genauigkeit bei der Handerkennung und der Qualität des Beamerbildes.

Da das Leap Motion mit Infrarotlicht arbeitet, sind externe Infrarotquellen negativ für die Erkennung der Hand. Sobald der Leap Motion Controller eine solche erkennt, wird diese herausgerechnet und damit ausgeglichen. Bei dieser Berechnung verschlechtert sich die Genauigkeit, was bis zum Ausfall der Handerkennung führen kann. Dieser Umstand wird sichtbar, wenn die Infrarotbilder aus der zweiten Versuchsreihe in Unterabschnitt 4.3.2 mit den Ergebnissen der Reihenfolge aus der ersten Versuchsreihe in Unterabschnitt 4.3.1

verglichen werden. Um diese Bilder zu bewerten, muss bekannt sein was auf ihnen erkennbar ist. Sie zeigen die Menge des Lichtes, die von der Oberfläche im Infrarotbereich reflektiert wird. Je heller ein Bild ist, desto stärker wird das Licht zurück geworfen. Da sich dieser Umstand jedoch negativ auf die Funktionalität des Leap Motion auswirkt, ist eine Oberfläche geeigneter, je dunkler das Bild ist.

Sobald eine Oberfläche das Infrarotlicht stark reflektiert, wird die Genauigkeit des Leap Motion beeinträchtigt. Dies ist besonders gut an dem weißen Stoff zu erkennen. Aus dem Bild in der Tabelle ist die sehr gute Reflexion ersichtlich, was der Qualität des Beamerbildes zugute kommt. In oben stehender Tabelle ist der weiße Stoff jedoch erst an 12. Stelle vertreten welches darauf zurück zuführen ist, dass diese Oberfläche sehr stark reflektiert und dadurch die Erkennung durch das Leap Motion stark beeinträchtigt ist.

Eine geeignete Oberfläche, darf höchstens leichte Schwankungen bei der Handerkennung hervorrufen und muss mindestens „2=gut geeignet" sein, eine Beamerprojektion darauf abzubilden. Mit diesen Einschränkungen kommen durch die erste

Versuchsreihe vier Oberflächen in Frage. Die Dia-Leinwand, der Fernseher, die Fotofarbfolie sowie das raue Metall erzeugten leichte Schwankungen.

Die zweite Versuchsreihe kennzeichnete fünf Oberflächen als geeignet. Die Dia-Leinwand, weiße Stoffe, weiß lackiertes Holz, weiße Tapete sowie Fliesen waren gut geeignete Projektionshintergründe.

Die Versuchsreihen ergeben ein eindeutiges Ergebnis; dieses ist ein Kompromiss aus allen geforderten Kriterien und Anforderungen. Die Dia-Leinwand ist sowohl für die Handerkennung gut geeignet, da sie nur leichte Schwankungen hervorruft, als auch als Projektionsfläche. Daher wird sie für die weitere Produktentwicklung als Hintergrund genutzt.

5 Implementierung

Dieses Kapitel behandelt die Implementierung, dazu wird die Architektur dieses Projektes erläutert. Außerdem werden die genutzten Frameworks erläutert und deren Implementierung gezeigt. Um einen Überblick zu bekommen wie die wichtigsten Teile des Touchbeameraufbaus umgesetzt wurden, werden die Kalibrierung sowie die Berechnung des Touchbereichs umfangreich beschrieben.

5.1 Architektur

Bevor die Umsetzung der Software gestartet werden kann, wird auf die Architektur des Projektes eingegangen. Dabei wird ein besonderes Augenmerk auf die System- und Softwarearchitektur gelegt.

5.1.1 Systemarchitektur

Um einen guten Systemaufbau zu ermöglichen, müssen gewisse Umstände beachtet werden. Da das Leap Motion nur eine begrenzte Reichweite von maximal 60 cm hat, wobei die Genauigkeit mit größerer Entfernung abnimmt, darf es nicht zu weit von der Projektion entfernt sein. Des Weiteren gibt es Schwierigkeiten bei der Erkennung der Hand, wenn das Leap Motion nicht auf die Handinnenfläche gerichtet ist. Daher muss darauf geachtet werden, dass der Aufbau exakt gewählt wird, damit eine möglichst verlässliche Erkennung der Hand erfolgt.

Der Beamer sollte jedoch einen größeren Abstand zur Leinwand haben.

In dem Bereich, in dem das Leap Motion eine gute Genauigkeit aufweist, ist das Bild etwa 26 Zentimeter hoch.

Es sollte nach Möglichkeit ein Kompromiss gefunden werden, der beide Kriterien bestmöglichst erfüllt. Um das zu erreichen, müssen das Leap Motion und der

Beamer unterschiedlich weit von der Projektionsfläche entfernt sein. Dafür wird der Touchbeamer so aufgebaut, dass der Beamer auf einem Stativ in Richtung der Projektionsfläche zeigt. Das Leap Motion liegt nach oben gerichtet mit einem Abstand von zehn Zentimetern vor der Leinwand.

Die optimale Größe der Projektion wurde durch einen Versuch ermittelt. Diese liegt in dem Bereich, wo das Leap Motion die größte Genauigkeit bietet. Dabei wurde der minimale Abstand des Leap Motion ermittelt. Hierbei war der Visualizer vonnöten, um festzustellen, wie die Hand erkannt wurde und welchen Schwankungen und Abweichungen diese unterlag. Der optimale Abstand von der Projektion zum Leap Motion beträgt hier 17 cm.

Die Höhe der Projektion wird durch die maximale Reichweite des Leap Motion erreicht, diese beträgt 43 cm. Innerhalb dieser Grenzen liegt die Projektionsfläche optimal für die Erkennung der Hand.

5.1.2 Softwarearchitektur

Der folgende Abschnitt beschäftigt sich mit der Softwarearchitektur des Touchbeamers.

Die Software arbeitet in drei Schritten. Daher wurde auch die Architektur so gestaltet, dass diese drei Schritte einzeln implementiert werden können. Dadurch ergibt sich der wichtige Umstand, dass einzelne Komponenten getestet und einfach optimiert werden können.

Der `LeapListener` ist die Schnittstelle zwischen dem Leap Motion Hardware Controller und der Software. Letzterer sendet durchgehend für jeden Frame alle Daten, die in seinem Erkennungsbereich sind. Dazu gehören alle Hände und Finger.

Die Daten, die direkt vom Leap Motion kommen, werden mithilfe des RXJava Frameworks weitergeleitet. Dies wird in der `LeapListener` Klasse umgesetzt. Dazu wird ein `BehaviorSubject` (Unterabschnitt 5.2.1) genutzt. Um die Daten zu versenden, nutzt der Leap Motion Hardware Controller den Typ `Frame`. Wenn ein solcher an den LeapListener geschickt wird, überprüft er automatisch die Erkennung einer Hand und nachfolgend die Registrierung eines Zeigefingers. Sobald ein solcher erkannt wurde, werden dessen Daten weitergeleitet. Hierzu wird der Typ `Finger`, der von dem Leap Motion SDK bereitgestellt wird, genutzt. Um die

Daten weiterzuleiten, wird die `onNext` Methode des Subjects aufgerufen und die Fingerdaten übergeben.

Der erste Schritt der Softwarearbeit besteht in der Kalibrierung, welche die Möglichkeit bietet, eine Projektionsfläche nutzen zu können, die zuvor nicht festgelegt wurde. Dadurch entsteht die Möglichkeit, den Touchbeameraufbau an jedem Ort zu installieren, ohne dass das Setup beibehalten werden muss.

Für die Kalibrierung ist der `CalibrationController` zuständig. Dieser startet beim Erstellen die `GUI`, welche die Bedienung für die Kalibrierung ermöglicht. Der genaue Kalibrierungsvorgang wird im Abschnitt 5.4 beschrieben. Die Daten die durch die Kalibrierung entstanden sind, werden in der `Data` Klasse gespeichert. Desweiteren hat der `CalbrationController` eine Methode, die eine Ebene erstellen kann. Diese wird mithilfe der kalibrierten und gespeicherten Eckdaten erstellt.

Der zweite Schritt besteht in der Auswertung der Daten, diese Aufgabe übernimmt der `TouchController`. Er nimmt sich die Daten, die von dem `LeapListener` in das Subject weitergeleitet wurden. Mithilfe einer Berechnung, die im Abschnitt 5.5 beschrieben ist, wird der Bereich überprüft. Sollte ein Finger innerhalb einer gewissen Toleranzgrenze registriert worden sein, wird ein Klick ausgelöst. Dessen Position befindet sich jedoch in einem anderen Bezugssystem als der Monitor. Die Koordinaten des Klicks bezeichnen nur dessen Abstände im 3D Raum zum Leap Motion. Um diese Koordinaten sinnvoll nutzen zu können, müssen sie in das Bezugssystem des Monitors umgerechnet werden. Diese Aufgabe wird direkt im `TouchController` durchgeführt.

Sobald ein Klick registriert worden ist, wird der Robot genutzt, dies ist der dritte Schritt. Dieses kann die Kontrolle über die Maus übernehmen und Klicks ausführen. Dazu verschiebt er die Maus zu einem bestimmten Punkt (`robot.mouseMove()`) und führt einen Klick aus(`robot.mousePress()`).

5.2 Datenfluss

Während der Umsetzung kam es bei der Übertragung der Koordinaten und Positionen zu größeren Datenmengen. Sie traten an der Schnittstelle von der Software und dem Hardware Controller auf. Dieser sendet je nach Stärke des Prozessors 20 - 30 Frames pro Sekunde.

Der `LeapListener` filtert die Daten, sodass diese nicht mehr weitergeleitet werden, wenn keine Hand erkannt wurde. Sobald jedoch eine Hand im Erkennnungsbereich vorhanden ist, werden die Daten wieder übertragen.

Daher wurde der Touchbeamer an einigen Stellen mit einem Datenfluss umgesetzt. Dies hat den Vorteil, dass größere Datenmengen nur in einen Datenstrom geführt werden müssen. Ein Objekt, das diese Daten benötigt, kann diesen Strom anzapfen und die Daten entsprechend nutzen. Die Bibliothek RXJava bietet diese Möglichkeit an.

5.2.1 RXJava

RXJava [Git15] ist eine Bibliothek, die es ermöglicht das Beobachter Muster (engl. Observer Pattern) zu nutzen. Bei diesem Pattern, gibt es zwei Akteure, zum einen das Subject, als beobachtbares Objekt, zum anderen ein oder mehrere Observer, als beobachtende Objekte. Dadurch lässt sich eine Beobachtung implementieren, ohne dass die beobachtenden Objekte in gleichmäßigen Abständen den Status des Subjects anfragen müssen.

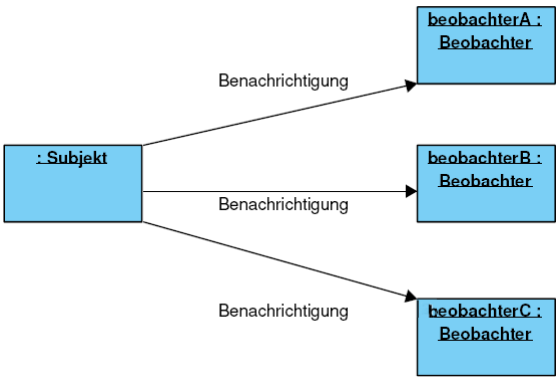


Abbildung 5.1: Darstellung der Benachrichtigungen mithilfe des Observer Patterns [Tor14]

Wenn das Beobachter Muster angewandt wird, können sich mehrere Observer auf ein Subject registrieren. Sollte sich nun das Subject ändern, teilt es dies allen registrierten Observern mit.

Diese Push-Notification ist besonders schnell, da keine Daten mitgeliefert werden, sondern nur eine Benachrichtigung darüber, dass sich das Subject geändert hat. Desweiteren gibt es noch die Push-Update Notification, welche zusätzlich noch Daten mitliefert, die die Änderungen beschreiben. [Tor14]

Wenn dieses Framework benutzt wird, existieren meist Schwierigkeiten, die auftreten, wenn man die Änderung eines Wertes erfahren möchte. Durch das Observer Pattern kann dieses Problem elegant gelöst werden. Es ist außerdem möglich mehrere Daten aus diesem Datenstrom zu ziehen und diese wie gewünscht zu verarbeiten. Diese Funktionalität wurde bei diesem Projekt genutzt, um die Daten vom Leap Motion Hardware Controller zur Software zu übertragen.

5.3 Apache Common Math 3

Das Apache Common Math3 Framework [Fou15] ist eine Sammlung von Funktionen mathematischer Art. Dieses Framework umfasst eine Vielzahl an Funktionen, die in verschiedene Packages unterteilt sind.

Für dieses Projekt wurde das Package `geometry.euclidean.threed` verwendet. Darin befinden sich die Klassen mit den Berechnungen für die Ebene `Plane` und die Vektoren `Vector3D`.

Für den Touchbeamer Aufbau ist die Klasse `Plane` sehr nützlich, da sie die Möglichkeit bereit stellt, eine Ebene zu erstellen. Die Ebene wird aus Vektoren gebildet, dazu werden drei der vier Vektoren genutzt, die bei bei der Kalibrierung erstellt wurden. Dadurch wird auf die Projektionsfläche eine Vektorebene gelegt, die es erlaubt, die Interaktionen zu berechnen.

5.4 Kalibrieren

Das Kalibrieren ist der erste Schritt des Touchbeamers. Um diesen umzusetzen, wurde der `CalibrationController` erstellt. Um die Kalibrierung durchführen zu können, greift der Controller auf den Datenstrom zu, der vom `LeapListener` erstellt wurde. Diese Daten werden in der `calibrate` Methode genutzt. Sie bekommt als Parameter den Namen des Eckpunktes auf der Fläche, die kalibriert werden soll. Diese Namen lauten topLeft, topRight, bottomLeft, bottomRight. Der Controller wird von der `GUI` aus gesteuert. Diese stellt für alle Eckpunkte einen Button bereit, der die `calibrate` Methode mit dem richtigen Parameter aufruft. Sobald dies erfolgt ist, wird aus dem Datenstrom ein Vektor erstellt. Dieser wird dann in der `Data` Klasse gespeichert, wie alle weiteren Eckpunkte.

Nachdem diese komplett eingelesen wurden, kann die Ebene, die auf der Projektion liegt, erstellt werden. Dazu wird die Methode `createPlane` verwendet. Sie bildet aus den Vektoren der Eckpunkte eine Ebene. Dazu verwendet sie die Plane, die das Apache Math 3 Framework (Abschnitt 5.3) zur Verfügung stellt. Um eine Ebene zu erstellen, werden drei Punkte benötigt. Dies repräsentieren drei der vier gespeicherten Eckpunkte. Der Vierte wird verwendet, um die Ebene zu validieren und damit sicherzustellen, dass sie für den weiteren Verlauf der Software nutzbar ist. Dazu wird der Abstand des vierten Eckpunktes zu der Ebene errechnet. Diese Distanz darf eine Toleranz von 20 mm in beide Richtungen aufweisen, damit die Ebene noch als valide gilt. Sollte dies der Fall sein, wird sie in die `Data` Klasse gespeichert. Auf der `GUI` wird angezeigt, dass die Kalibrierung erfolgreich war. Sollte die erstellte Ebene nicht valide sein, wird sie nicht gespeichert und dem Nutzer durch die `GUI` mitgeteilt, dass die Kalibrierung fehlgeschlagen ist. Sollte dies passieren, kann der Nutzer den Vorgang von vorne starten.

5.5 Berechnung Touchbereich

Sobald der Kalibrierungsvorgang erfolgreich beendet wurde, wird der `TouchController` gestartet. Dieser hat die Aufgabe, den Datenstrom auszulesen und die erhaltenen Daten auszuwerten.

Der Erkennungsbereich unterteilt sich in drei Ebenen.[1] Der erste Bereich wird betreten, wenn eine Hand erkannt, aber noch nicht in die sensitive Sphäre hinein gelangt ist. Diese beginnt jeweils 80 mm^2 vor und hinter der kalibrierten Ebene. Sobald eine Hand in diesem Bereich registriert worden ist, wird die Bewegung des Mauszeigers aktiviert und folgt in diesem Bereich der Fingerspitze. Somit wird dem Nutzer ermöglicht, dass er zuerst die Position sehen kann, bevor er einen Klick ausführt. Der eigentliche Klick, also die Touchgeste, wird ab einem Bereich von 25 mm^3 aktiviert. Wenn ein Finger in diesen Bereich vordringt, wird die linke Maustaste gedrückt. Sobald der Finger den Bereich wieder verlässt, wird die linke Maustaste losgelassen. Damit wird die mögliche Unterscheidung zwischen einem Klick und einem Verschieben umgesetzt.

Um die Position zu bestimmen, wohin der Mauszeiger bewegt werden soll, müssen

[1] Dieser wurde durch einen Versuch ermittelt, bei dem getestet wurde bei welchen Abständen das System am komfortabelsten zu Nutzen ist
[2] Durch diesen Abstand ist es möglich, nur den Mauszeiger zu bewegen
[3] Innerhalb dieses Bereichs wird ein Klick ausgelöst

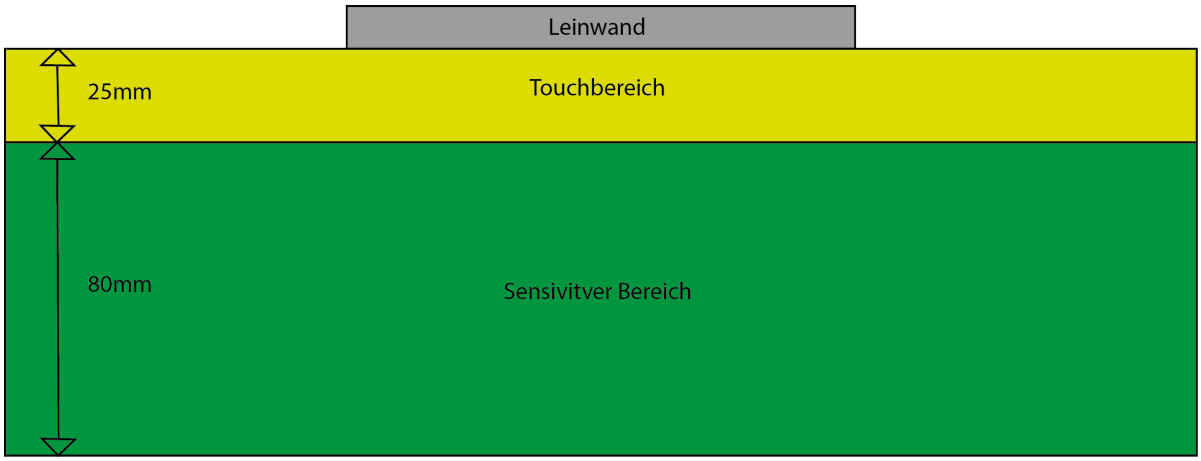


Abbildung 5.2: Unterscheidung der Touchbereiche

die Koordinaten in Werte umgerechnet werden. Diesen Werten kann der Computer eine eindeutige Position auf dem Bildschirm -den Pixeln, zuordnen.

Dazu wird zunächst der Abstand i auf der X-Achse des Klickpunktes P zu dem Eckpunkt A berechnet, dieser ergibt sich auf der Differenz der X-Koordinaten. Desweiteren wird die Länge der Linie a benötigt. Diese Linien werden aus der Perspektive des Leap Motion errechnet, also in Millimetern. Wenn nun diese Abstände dividiert werden, wobei der Abstand a der Divisor ist, erhält man eine Zahl, welches das Verhältnis der X-Koordinate des Klickpunktes P zur Gesamtlänge der kalibrierten Seite a ist.

Um nun aus diesem Verhältnis eine Pixel-Koordinate zu bekommen muss diese Verhältniszahl mit der Bildschirmauflösung multipliziert werden. In Abbildung 5.4 wird dieses Verfahren für die X-Achse gezeigt, die Bildschirmbreite ist in diesem Fall 800 Pixel groß. Für die Y-Koordinate wird ebenfalls nach diesem Muster verfahren.

Nach Berechnung der beiden Koordinaten ist der Punkt auf dem Bildschirm eindeutig bestimmt und diese Position kann zur Verschiebung des Mauszeigers genutzt werden.

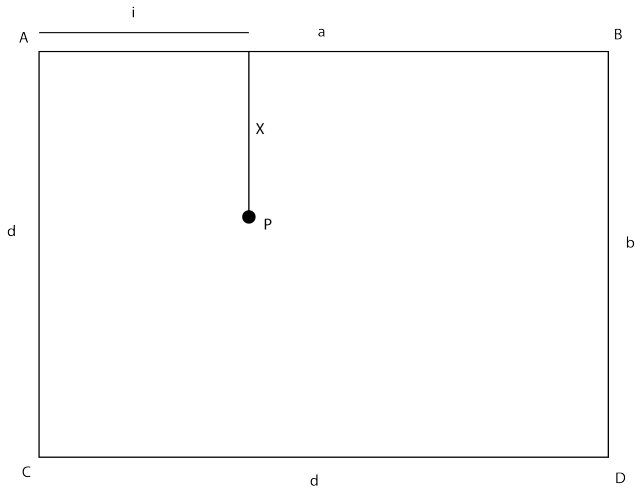


Abbildung 5.3: Skizze der Berechnung der Pixelkoordinaten

```
public double getXCoordinate(Vector3D intersectionPoint) {
double xCoordinateLeap = ((intersectionPoint.getX()
- data.getCornerTopLeft().getX()) /
(data.getCornerTopRight().getX() - data.getCornerTopLeft().getX()));

return xCoordinateLeap * 800;
}
```

Abbildung 5.4: Code Beispiel anhand der getXCoordinate() Methode

6 Test

Ein wichtiger Punkt der Software-Entwicklung ist die Nachweisführung. Um die Funktionalität zu belegen, muss die Software getestet werden. Die Tests unterteilen sich in zwei Gruppen. Zum einen gibt es Unittests. Diese werden für jede Methode, die getestet werden soll, geschrieben. Ein Unittest besteht aus zwei Komponenten, zum einen aus dem festen Wert, der in eine Methode gegeben wird, zum anderen aus einem Wert, der als Ergebnis dient. Der Unittest führt die Methode mit dem festen Wert aus. Dann wird das Ergebnis, das die Methode zurückliefert, mit dem Erwartungswert verglichen. Sollten beide Werte gleich sein, ist der Test korrekt verlaufen und die Methode arbeitet wie erwartet. Sollte der Rückgabewert jedoch nicht übereinstimmen, schlägt der Test fehl und die Methode wurde falsch implementiert. Für jede Methode werden immer mehrere Tests geschrieben, dadurch kann überprüft werden, wie sie sich verhält, wenn verschiedene Werte übergeben werden. Wichtig dabei ist, dass jeder Test auf ein bestimmtes Ergebnis hin prüft. Sollte eine Methode eine Fehlermeldung zurückgeben, falls ein falscher Parameter übergeben wurde, muss der Test darauf ausgelegt sein und das Ergebnis validieren.

6.1 Nachweisstrategie

Für den Touchbeamer werden die wichtigen Methoden getestet. Die Methode, die nach dem Kalibrierungsvorgang die Ebene validiert, muss geprüft werden, da das ganze System nicht funktionieren kann, wenn diese fehlerhaft oder ungenau ist. Dabei handelt es sich um die Methode `createPlane` in dem `CalibrationController`. Beim testen dieser Methode wurden zwei TestFälle entwickelt, der Positiv- und Negativtest. Dazu wurde jeweils eine Ebene erstellt, einmal mit Daten die validiert werden können und einmal mit falschen Daten. Durch diese Tests konnte nachgewiesen werden, dass die Kalibrierung richtig arbeitet und eine valide Ebene erstellt werden kann.

Die eigentliche Funktionalität, bestehend aus der Erkennung der Klickgeste und dem Umrechnen in ein Bezugssystem, mit dem die Position des Mauszeigers auf dem Bildschirm festgestellt werden kann, muss ebenfalls genau getestet werden. Die Methoden, die diese Aufgaben übernehmen, befinden sich in dem `TouchController`. Um die Mausposition zu bestimmen, werden zwei Methoden genutzt. Dabei handelt es sich um die `getXCoordinate()`und `getYCoordinate` Methoden, welche die Position der Touchgeste in Pixeln zurück geben. Um diese beiden zu testen, wurden die benötigten Daten mit dem Framework Mockito[1] gemockt. Das bedeutet, dass bei einem Zugriff auf die Data Klasse, gezielt Daten zurückgegeben werden. Dadurch ergibt sich der Vorteil, dass verschiedene kalibrierte Eckpunkte verwendet werden können, ohne dass wirklich eine Kalibrierung stattgefunden hat. Die Tests beschreiben jeweils verschiedene Positionen eines Klicks. Dabei wird getestet, ob die Mausposition korrekt berechent wird.

Wenn die Unittests alle korrekt verlaufen sind, bedeutet dies, dass die einzelnen Methoden richtig arbeiten. Der nächste Schritt ist nun ein Integrationstest. Hierbei soll das Zusammenspiel der Methoden getestet werden. Dazu wird das Programm gestartet und alle Funktionen werden automatisch durchgeführt. Damit dieser Test korrekt durchläuft, müssen die einzelnen Methoden zusammenarbeiten.

Im Falle des Touchbeamers, kann jedoch die Funktionalität nicht automatisch getestet werden, da das Leap Motion die Bewegungen und die Positionen einscannen muss. Daher wird ein Integrationstest geschrieben, der halbautomatisch abläuft. Dieser Test soll die Funktionen zusammenhängend testen und benötigt dann nur noch die Bewegungen einer Hand, die manuell durchgeführt werden müssen.

Dazu wurde eine Testanwendung geschrieben, bei der die Funktion des Touchbeameraufbaus getestet wurde. Diese Anwendung besteht aus vier Buttons und einem Slider. Um den Test erfolgreich abzuschließen, müssen alle Buttons angeklickt und der Slider vollständig verschoben werden. Sobald dies geschehen ist, wird das jeweilige Element ausgegraut. Durch diesen Test konnte bestätigt werden, dass die Funktionalität erfolgreich umgesetzt wurde.

Eine Produktanforderung ist die Betriebssystemunabhängigkeit. Um diese zu testen wurde das Projekt auf einem Mac, einem Windows 7 und einem Ubuntu-Linux System getestet. Auf jeder der Plattformen konnte die Software erfolgreich gestartet und genutzt werden.

[1] http://mockito.org

7 Fazit

Das Konzept des Touchscreens auf eine Projektion zu übertragen ist eine innovative Möglichkeit der Steuerung eines Systems. Im Rahmen dieser Thesis wurde ein Vorschlag für eine Umsetzung dieses Konzeptes gemacht, dabei wurde auf alle auftretenden Probleme eingegangen. Dazu wurden zunächst die Grundlagen erläutert und die Anforderungen aufgestellt. Während der Analyse traten Schwierigkeiten bei der Wahl des Projektionshintergrundes auf, welche jedoch durch festegelegte Versuchskriterien gelöst werden konnte.

Die Nutzung des Leap Motion als Erkennungsgerät war eine besondere Herausforderung, da dieses nicht für diesen Einsatzzweck konzipiert wurde. Es war vor dem Start des Projektes nicht klar, ob die genutzten Konsumer-Hardware Produkte für das Projekt geeignet waren.

Anhand einer soliden Analyse konnte jedoch eine Umsetzung erfolgen, die durch eine Testanwendung validiert werden konnte. Dieses Konzept der Umsetzung erzeugt eine Software, die die geforderte Funktionalität bietet, jedoch einige Schwächen im Bezug zur Genauigkeit aufweist. Dies wurde durch den Umstand der nicht hundert prozentig geeigneten Projektionsfläche verstärkt. Trotzdem kann das System sinngemäß genutzt werden.

Um den low-cost Ansatz zu erfüllen wurde auf kostengünstige Consumer Hardware gesetzt. Das Leap Motion (89,99 €) und der Beamer (244,99 €) haben ingesamt Kosten von 334,98 € verursacht. Dies entspricht dem festgelegten Ansatz. Jedoch wurde die Kosteneinsparung auf Kosten der Genauigkeit umgesetzt.

Das Projekt hat mir große Freude bereitet, da es für jede Herausforderung eine Lösung gab, die dazu beigetragen haben das Projekt erfolgreich abzuschließen. Es konnte gezeigt werden, dass das Prinzip des Touchscreens auf einer Projektion auch mit Consumer Hardware umgesetzt werden konnte.

Literaturverzeichnis

[com10] COMPUTER bora: *Xbox360 Microsoft Kinect Sensor*. http://bora-computer.de/shopware.php?sViewport=detail&sArticle=1477. Version: 02 2010. – Zuletzt abgerufen am: 02.03.2015

[Cor12] CORELLIANROGUE: *Kinect's Accuracy Analysed!* http://123kinect.com/kinect-forums/thread-3729.html. Version: 06 2012. – Zuletzt abgerufen am: 19.2.2015

[DAC14] DACH, Philips GmbH M.: *PicoPix Taschenprojektor*. http://www.philips.de/c-p/PPX3614_EU/picopix-taschenprojektor. Version: 2014. – Zuletzt abgerufen am: 12.01.2015

[Dev15a] DEVELOPERS, Leap M.: *Coordinate Systems*. https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Coordinate_Mapping.html. Version: 2015. – Zuletzt abgerufen am: 27.02.2015

[Dev15b] DEVELOPERS, Leap M.: *Introducing the Skeletal Tracking Model*. https://developer.leapmotion.com/documentation/objc/devguide/Intro_Skeleton_API.html. Version: 2015. – Zuletzt abgerufen am: 18.01.2015

[Fou15] FOUNDATION, The Apache S.: *Apache Common Math 3 Framework*. http://commons.apache.org/math/javadocs/api-3.3/index.html. Version: 2015. – Zuletzt abgerufen am: 15.02.2015

[Git15] GITHUB: *RXJava*. https://github.com/ReactiveX/RxJava. Version: 2015. – Zuletzt abgerufen am: 10.02.2015

[Hon14] HONG, Kaylene: *Leap Motion's CEO wants its gesture control in cars, as a software upgrades to track hand nears*. http://thenextweb.com/insider/2014/05/09/

leap-motions-ceo-wants-its-gesture-control-in-cars-as-a-software-upgra
Version: 05 2014. – Zuletzt abgerufen am: 03.01.2015

[Kle15] Klempau, Kristoffer: *Leap Motion Controller*. http://www.mediacenter-pc.com/leap-motion-controller. Version: 2015. – Zuletzt abgerufen am: 03.01.2015

[Mad12] Mader, Alexander O.: *Berührungslose Benutzerinteraktion für .NET-basierte Software-Anwendungen*. September 2012. – Bachlor-Thesis, Fachhochschule Kiel

[Mic15] Microsoft: *Welcome to Microsoft PixelSense*. http://www.microsoft.com/en-us/pixelsense/default.aspx. Version: 2015. – Zuletzt abgerufen am: 25.02.2015

[Mot15] Motion, Leap: *Leap Motion*. https://www.leapmotion.com/product. Version: 2015. – Zuletzt abgerufen am: 02.03.2015

[MRS12] Magnor, Marcus ; Ruhl, Kai ; Scholz, Alexander: *Flying with the Kinect*. http://www.cg.cs.tu-bs.de/teaching/labs/ss11/sep. Version: 09 2012. – Zuletzt abgerufen am: 11.01.2015

[Phi15a] Philips: *PicoPix PPX2055 Taschenprojektor*. http://www.philips.de/c-p/PPX2055_EU/picopix-taschenprojektor. Version: 2015. – Zuletzt abgerufen am: 20.02.2015

[Phi15b] Philips: *PicoPix PPX2480 Taschenprojektor*. http://www.philips.de/c-p/PPX2480_EU/picopix-taschenprojektor. Version: 2015. – Zuletzt abgerufen am: 20.02.2015

[Phi15c] Philips: *PicoPix PPX3614 Taschenprojektor*. http://www.philips.de/c-p/PPX3614_EU/picopix-taschenprojektor. Version: 2015. – Zuletzt abgerufen am: 20.02.2015

[Tor14] Tornau, Christoph: *Das Observer-Pattern / Beobachter-Muster*. http://www.tornau.name/2014/02/das-observer-pattern-beobachter-muster. Version: 02 2014. – Zuletzt abgerufen am: 15.03.2015

[WBRF13] Weichert, Frank ; Bachmann, Daniel ; Rudak, BartholomÃ€us ; Fisseler, Denis: *Analysis of the Accuracy and Robustness of the Leap Motion Controller*. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3690061. Version: 05 2013. – Zuletzt abgerufen am: 18.02.2015