Rosette 4.0

Experiment, Theorie und Praxis am Beispiel eines kinetischen, bildgebenden Körpers

Diplomarbeit

Ausgeführt zum Zweck der Erlangung des akademischen Grades
Dipl.-Ing. für technisch-wissenschaftliche Berufe

am Masterstudiengang Digitale Medientechnologien an der
Fachhochschule St. Pölten, **Masterklasse Experimentelle Medien**

von:
Fabian Kindl, BSc
dm161560

Betreuer/in und Erstbegutachter/in: Dipl.-Ing. Christian Munk
Zweitbegutachter/in: Mag. Markus Wintersberger

Wien, 09.09.2018

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

- ich dieses Thema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

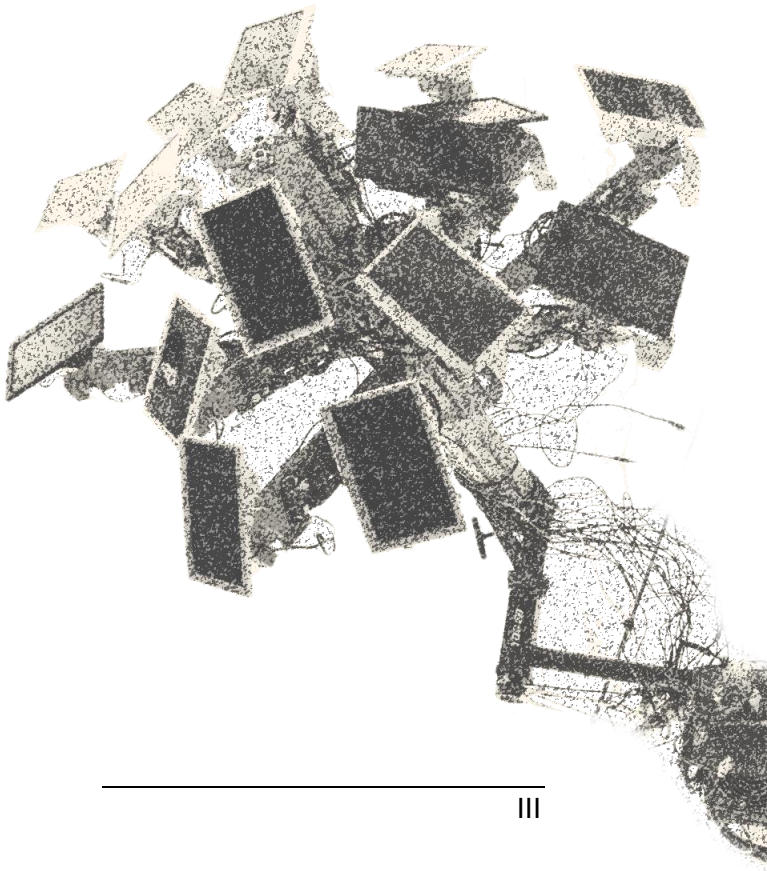
Diese Arbeit stimmt mit der vom Begutachter bzw. der Begutachterin beurteilten Arbeit überein.

Wien, 09.09.2018 .....

Ort, Datum Unterschrift

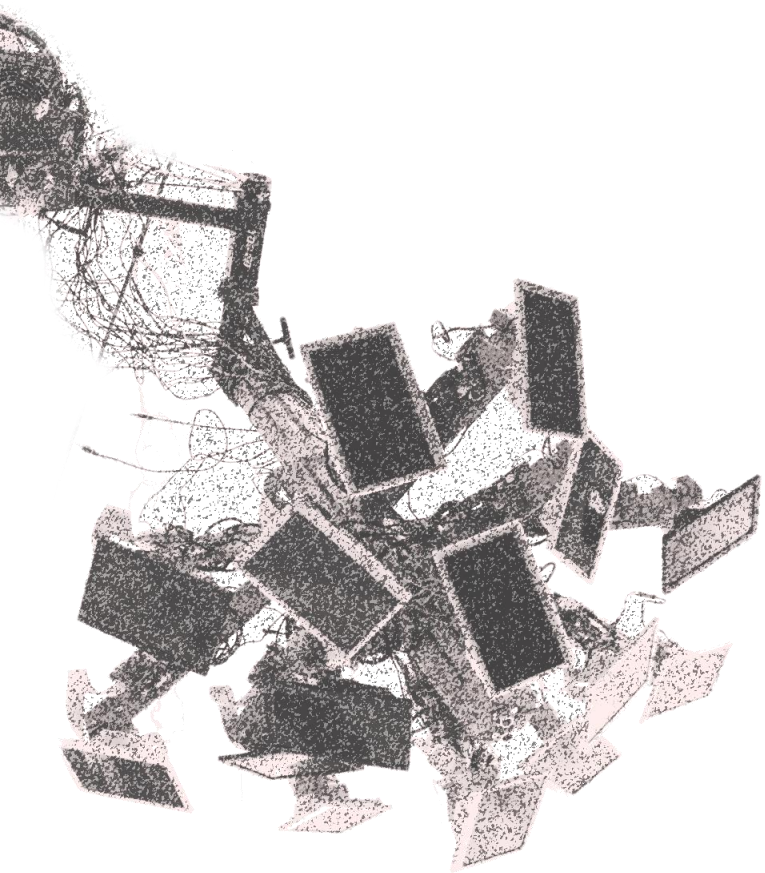
Kurzfassung

Diese Arbeit beschäftigt sich mit der Entwicklung und der Umsetzung von Modulen, aus denen kinetische, bildgebende Körper gebaut werden können, sowie mit der zentralen Steuerung dieser. Jedes Modul besteht unter anderem aus einem Display, welches mit Motoren die Lage im Raum verändern kann. Aus 16 Modulen wird im Rahmen einer künstlerischen Arbeit ein interaktiver, kinetischer, bildgebender Körper geschaffen. Während einer Ausstellung wird diese Installation in Form qualitativer Befragung getestet. Diese Arbeit liefert, neben der Dokumentation der gesamten Entwicklung und Umsetzung, Antworten auf Umsetzbarkeit, Bewegungen, Anordnungen, Interaktionsmuster und Umgebung. Außerdem bildet diese Arbeit eine Basis für weitere Projekte dieser Art, indem das gesamte erarbeitete Material in Form eines Bauplans zur freien Verfügung gestellt wird, sodass die Module sowie die Steuerung dieser hard- und softwareseitig nachgebaut werden können.



Abstract

This work deals with the development and implementation of modules from which kinetic, imaging bodies can be built. It also covers the development of the central controller. Part of the module is a display, which can be moved in space. In the context of an artistic work, an interactive, kinetic, imaging body out of 16 modules is created. This installation will be tested in the form of a qualitative survey during an exhibition. In addition to documentation of the entire development and implementation, this work provides answers to feasibility, movements, arrangements, interaction patterns and environment. Furthermore, this work forms a basis for similar projects, as the entire developed material is provided as blueprint. In this way hard- and software of the modules and the controller can be recreated.



Inhaltsverzeichnis

Danksagung:
Ohne sie wäre die Arbeit nicht möglich gewesen!

Danke an

Markus Wintersberger - Christian Munk - Thomas Wagensommerer

für die Inspiration und Unterstützung,

Patrik Lechner

für die Touchdesigner Skills,

Christoph Braun

für die 3D Druck Unterstützung,

Matthias Husinsky

für die vvvv Skills,

FH Sankt Pölten

für die Unterstützung,

Magdalena Müllner,

für die Hilfe von allen Seiten,

David Kindl,

für den Radständer,

den Youtube-Kanal GXP2008,

für die genialen Körpersounds,

alle Probanden,

für das ehrliche Feedback,

Zimmerei Kindl,

für die zur Verfügungstellung von Material und Maschinen,

und an alle erwähnten Softwareentwickler,

für die zur Verfügung gestellten Programme.

1 Einleitung

Durch die ständige Weiterentwicklung und Verkleinerung der Computer ist die Größe einiger Modelle mittlerweile auf einige Zentimeter geschrumpft. Diese Tatsache eröffnet eine Vielzahl von neuen Projektmöglichkeiten, da durch den geringeren Platz- und Strombedarf andere räumliche Anordnungen möglich werden. Die geringeren Produktionskosten ermöglichen immer mehr Menschen den Kontakt mit solchen kleinen Computern. Dadurch können sich Communities bilden und in diesen, Wissen ausgetauscht werden.

Diese Arbeit beschäftigt sich mit der Planung und der Umsetzung einzelner Module, die zum Bau von kinetischen, bildgebenden Körpern verwendet werden können. Jedes Modul soll eigenständig durch einen kleinen Einplatinencomputer betrieben werden, um dadurch die Skalierbarkeit für den gesamten Körper beziehungsweise die gesamte Installation zu gewährleisten.

Im Zuge dieser Arbeit wird der, mit Hilfe von Presumptive Design gestaltete, Werdegang der Idee vorgestellt. Kapitel 2 geht näher auf die verwendete Methode Presumptive Design ein. In Kapitel 3 werden drei spannende Projekte im fachlichen Umfeld vorgestellt, um einen Bezug auf wichtige Thematiken in diesem Bereich zu bekommen. Kapitel 4 beschreibt den Entwicklungsweg, welcher die Module und die Installation geformt hat. Dieser beginnt von der ersten Idee und endet bei einem funktionstüchtigen Prototyp. Aus diesen gewonnenen Erfahrungen, Ideen und Pläne wird eine interaktive Installation zusammengesetzt. Kapitel 5 beschreibt die Funktionsweise dieser Installation im Überblick und beschreibt Tools, die für die Umsetzung notwendig sind. Kapitel 6 und 7 erklären die Hardware und Software der Installation im Detail. Die erarbeiteten Pläne und Files werden zur freien Verfügung gestellt. Kapitel 8 erklärt die notwendigen Schritte, die zur Umsetzung solch einer Installation notwendig sind. Die fertige Installation wird ausgestellt und einer qualitativen Befragung unterzogen. Details dazu sind in Kapitel 9 zu finden. Das letzte Kapitel 10 schließt die Arbeit mit der Beantwortung der Forschungsfragen ab und gibt einen Ausblick über mögliche Weiterentwicklungen.

1.1 Zielsetzung, Fragestellung und Methodik

Das Ziel dieser Arbeit ist es, eine Installation zu schaffen, welche ein immersives Erlebnis durch Deformation der Bildfläche ermöglicht. Die Deformation soll mit Hilfe einzeln beweglicher Module gewährleistet werden. In dieser Arbeit werden die notwendigen Module in enger Zusammenarbeit mit der Zielgruppe entwickelt und umgesetzt. Mit Hilfe der entstandenen Module soll eine gesamte Installation geschaffen werden, welche aus Sensorik, Datenverarbeitung, Visualisierung, Aktorik und Sonifikation bestehen soll und mit der ein menschlicher Anwender in Symbiose treten kann.

Die Zielgruppe dieser Installation sind Menschen in meinem Umfeld. Einige haben technischen Hintergrund, andere nicht. Jedoch soll für beide ein immersives Erlebnis geschaffen werden.

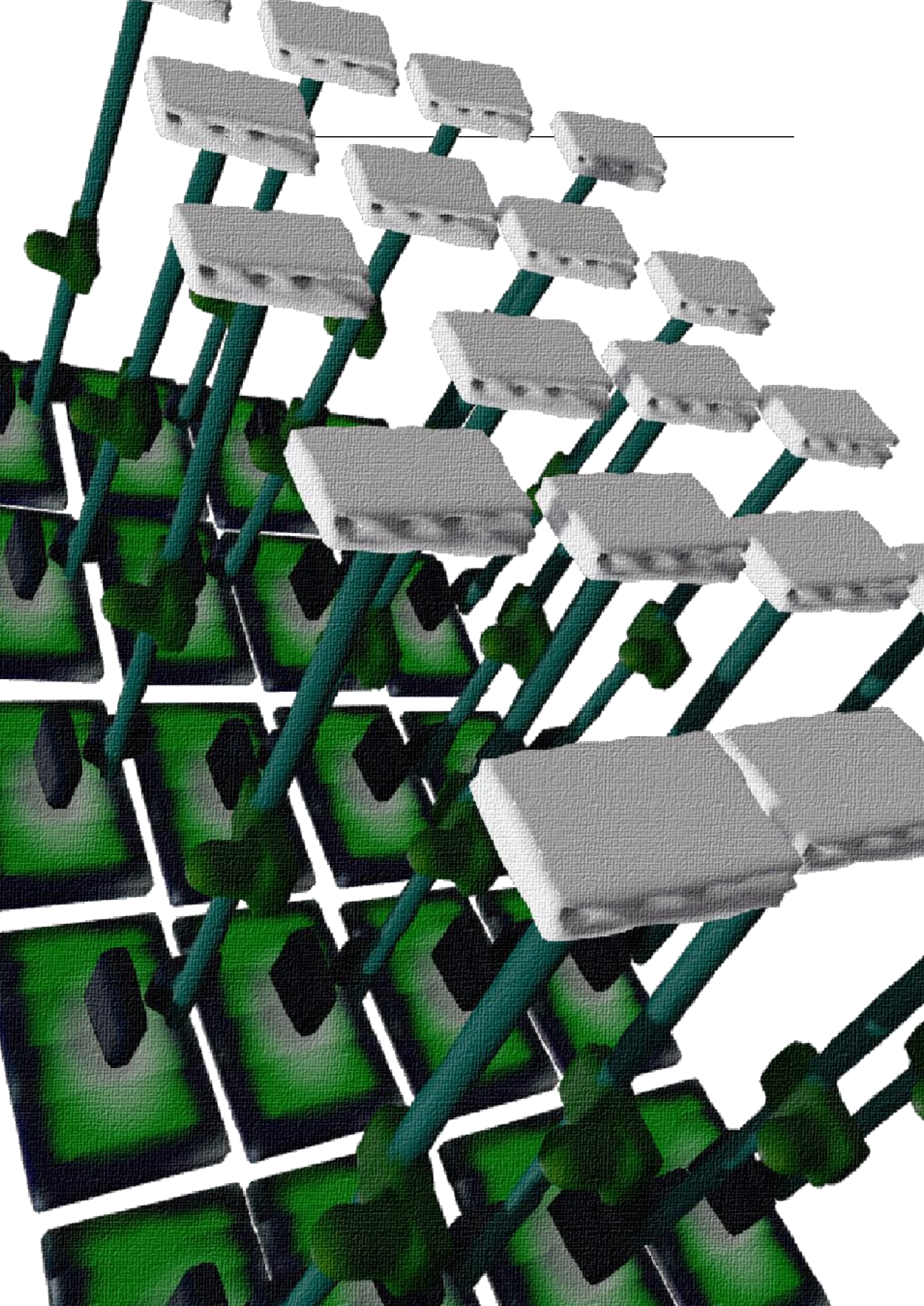
Die in dieser Arbeit entstandenen Skizzen, Programmcodes und so weiter sollen in Form eines Bauplans dem Leser für den Nachbau, oder dem Bau ähnlicher Installationen zur Verfügung gestellt werden. Außerdem soll durch die fertige Installation, dem Benutzer die Möglichkeiten einer deformierbaren Bildfläche erlebbar gemacht werden.

Um schon von Beginn an eng mit der Zielgruppe zusammenzuarbeiten, wird für den Entwicklungsprozess und für die Beantwortung der Forschungsfragen die wissenschaftliche Methode Presumptive Design verwendet. Kapitel 2 geht näher auf diese Methode ein. Nach der Fertigstellung der Installation wird diese der Zielgruppe vorgestellt. Dabei findet eine qualitative Befragung in Form offener Gespräche statt. Durch Interpretation dieser Aussagen sollen Folgeschlüsse auf die Wirkung der Installation gezogen und einige Forschungsfragen beantwortet werden. Ebenso sollen daraus weitere Ideen, Möglichkeiten sowie Fragestellungen dazu entstehen.

Aus den Gegebenheiten ergeben sich folgende Forschungsfragen:

- Welche Umsetzung bzw. Komponenten stellen sich als kostengünstig aber trotzdem zweckerfüllend heraus?
    - Diese Frage wird im Laufe des Entwicklungsprozesses in Kapitel 4 beantwortet.
- Welchen Mehrwert bietet die Bewegung für die Installation?
    - Diese Frage wird im Zuge der qualitativen Befragung in Kapitel 9 beantwortet.
- Welche Anordnungen sind mit solchen Modulen möglich bzw. wirken besonders immersiv?

    - Teilweise findet diese Frage in Kapitel 4, teilweise in der qualitativen Befragung in Kapitel 9 Antwort.
- Welche Interaktionsmuster eignen sich für Installationen dieser Art?
    - Diese Frage wird ebenfalls im Zuge der qualitativen Befragung in Kapitel 9 beantwortet.
- Wie soll die Umgebung der Installation beschaffen sein?
    - Teilweise findet diese Frage in Kapitel 4, teilweise in der qualitativen Befragung in Kapitel 9 Antwort.



2 Presumptive Design

Presumptive Design (PrD) ist eine Forschungs-Technik für Design, die es kleinen sowie großen Organisationen erlaubt, die Ziele und Bedürfnisse ihrer Zielgruppe zu identifizieren. Diese Methode ist schnell und günstig. Somit können Risiken eines Projektes kosteneffizient, in kürzester Zeit minimiert werden (Frishberg & Lambdin, 2015, S.4).

Presumptive Design ist nicht nur auf ein Produkt oder Service anwendbar. Es kann für jede Art von Projekt verwendet werden, wobei unter Projekt jedes Bestreben gemeint ist, dass mit materieller Auswirkung oder Risiko zusammenhängt (Frishberg & Lambdin, 2015, S.6).

Bei PrD geht es darum, in kürzester Zeit die Bedürfnisse der externen Stakeholder (Kunden, User, etc.) zu verstehen. Die Schwierigkeit dabei ist es, deren Denkweise, Neigung, übliche Arbeitsweise und deren Definition von Begeisterung zu erkennen und zu begreifen. Die Autoren (Frishberg & Lambdin, 2015) vergleichen diese Aufgabenstellung mit folgender Analogie:

Abbildung 1 – PrD Analogie

Die gesuchte Information verbirgt sich verschlossen im Kopf des externen Stakeholders, wie die glatte Oberfläche eines Teichs. PrD wirft einen Stein in den Teich hinein und verursacht dadurch Wellen an der Oberfläche, wie in Abbildung 1 ersichtlich. Diese einzigartigen Wellen repräsentieren die Reaktionen der

externen Stakeholder auf die Annahmen der internen Stakeholder. Das Team kann diese Wellen erfassen und so auf die Reaktionen reagieren, indem es die zuvor festgelegten Annahmen verändert. Dieser Zyklus kann immer wieder wiederholt werden. So entsteht ein Dialog zwischen internen und externen Stakeholdern, der die verborgenen Informationen der letzteren hervorbringen soll (Frishberg & Lambdin, 2015, S. 30).

2.1 Ablauf und Struktur

Zu Beginn werden die internen Stakeholder zu einer sogenannten „Creation Session“ eingeladen. In großen Organisationen nehmen hier beispielsweise Mitglieder aus den Bereichen des Marketings, der Entwicklung, des Verkaufs, des User Experience Designs und der Geschäftsleitung teil. In diesen „Creation Sessions“ entsteht in der gemeinsamen Zusammenarbeit ein „Artefakt“. Es beinhaltet die Erwartungen des Teams an das gemeinsame Vorhaben und an die Bedürfnisse der externen Stakeholder. Abschnitt 2.2 geht näher auf den Begriff „Artefakt“ ein. Abhängig vom Umfang und Aufwand des Vorhabens kann eine „Creation Session“ ein paar Stunden bis ein paar Tage dauern (Frishberg & Lambdin, 2015, S.5).

Anschließend wendet sich ein Forschungsteam mit dem entstandenen „Artefakt“ an die externen Stakeholder. Diese können beispielsweise Benutzer oder Kunden sein. In diesen sogenannten „Engagement Sessions“ treten die externen Stakeholder mit dem „Artefakt“ in Interaktion. Dabei gilt es für sie oder ihn ein bestimmtes Ziel zu erfüllen. Diese Session kann, wieder abhängig von der Größe des Vorhabens, etwa eine halbe Stunde dauern. Aus dieser Interaktion zwischen Nutzer und „Artefakt“ können unzählige Daten und Reaktionen gesammelt werden. Die wichtigsten Ergebnisse, die aus der Session abgeleitet werden können, sind die Hinweise wie die Annahmen der internen Stakeholder mit den Bedürfnissen der externen Stakeholder resonieren (Frishberg & Lambdin, 2015, S.5).

Alle gewonnen Informationen aus der „Engagement Session“ können anschließend analysiert werden und in den Entstehungsprozess des nächsten „Artefakts“ miteinfließen. Abbildung 2 zeigt den Ablauf und das Timing eines PrD-Zyklus.

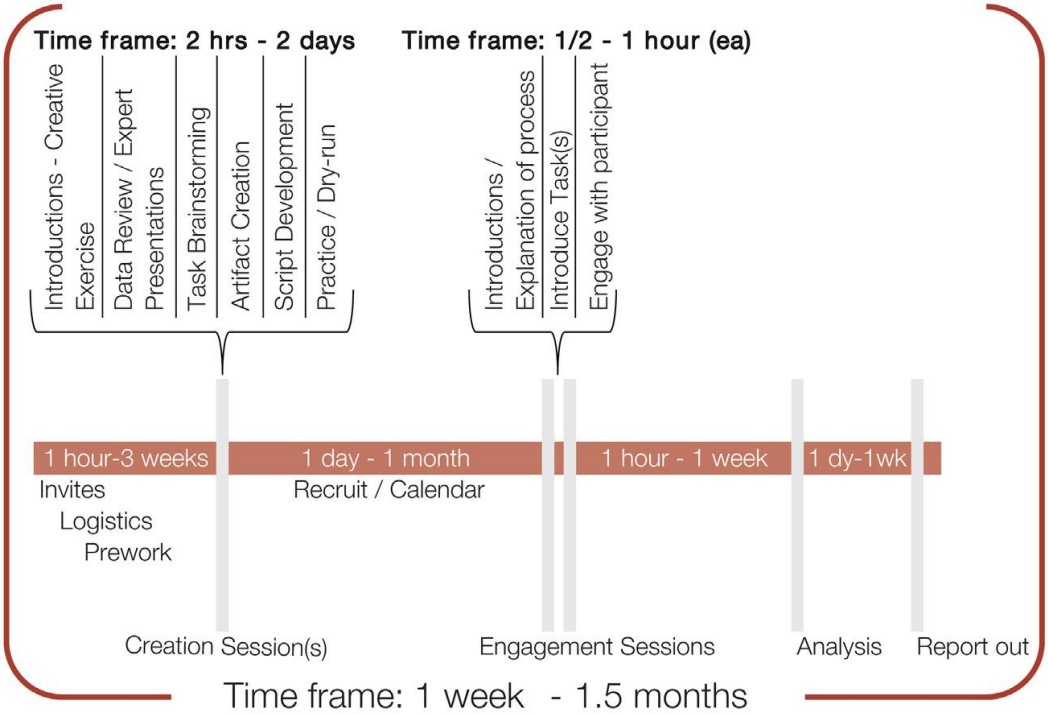


Abbildung 2 – Die typische PrD Timeline (Frishberg & Lambdin, 2015, S. 5)

2.2 Das „Artefakt“

„Objects are a form of knowledge about how to satisfy certain requirements, about how to perform certain tasks.“ – Nigel Cross (Frishberg & Lambdin, 2015, S. 240)

Wie in den vorangegangenen Abschnitten beschrieben, repräsentiert ein „Artefakt“ die Annahmen der internen Stakeholder und provoziert Reaktionen der externen Stakeholder (Frishberg & Lambdin, 2015, S. 69). Ein „Artefakt“ muss dabei nicht unbedingt, wie im Zitat von Nigel Cross, ein Objekt sein. Es muss lediglich eine physikalische Repräsentation der ersten Schätzungen des Teams für die Bedürfnisse der externen Stakeholder sein. Diese Vermutungen des Teams sollen dabei so schnell wie möglich in die physikalische Welt gebracht werden. Dadurch können in der Interaktion miteinander die „Artefakte“ verfeinert und zu einem Gemeinsamen verbunden werden (Frishberg & Lambdin, 2015, S. 270).

Es kann beispielsweise auch aus einer Nachricht oder einer Ankündigung bestehen (Frishberg & Lambdin, 2015, S. 62). Die Autoren (Frishberg & Lambdin, 2015) beschreiben in ihrem Buch weitere Formen von „Artefakten“ die in bisherigen „Creation Sessions“ entstanden sind:

- Mehrere Aussagen zur Vision
- Ein grob umrissener Businessplan
- Darstellung eines Produkts oder eines Interfaces (Frishberg & Lambdin, 2015, S. 69)
- Ein Stück Karton mit Klebezetteln und Pfeifenreinigern
- Sketches
- Wireframe (Frishberg & Lambdin, 2015, S. 270)

Wie in den obigen Beispielen hier angeführt, kann ein „Artefakt“ unzählige Formen annehmen. Wichtig dabei ist nur wie und warum es scheitert (Frishberg & Lambdin, 2015, S. 62). Der nächsten beiden Abschnitte behandeln die Unterschiede von PrD zu „User Centered Design“ und zu „Rapid Prototyping“.

2.3 Abgrenzung zu „User Centered Design“

„User Centered Design“ (UCD) und PrD beruhen auf denselben vier Entwicklungsstufen. UCD startet mit der Identifizierung des Problems, formuliert es durch Analysen, entwickelt daraus Konzepte und fertigt daraus schließlich eine Lösung in Form eines Prototypen (Know – Reflect – Make – Act) (Frishberg & Lambdin, 2015, S. 28).

Wie rechts in Abbildung 3 ersichtlich, stellt PrD diese Abfolge auf den Kopf. Es startet mit Konzepten und geht schnell zu Lösungen über. Diese werden erkundet und die Erkenntnisse daraus formuliert. Diese 180° Drehung verändert die Rahmenbedingungen in der Form, dass sich das Team direkt mit den wesentlichen Problemen und Bedürfnissen auseinandersetzt und es gezwungen wird sofort ihre Vermutungen zu äußern um diese anschließend dem Kunden oder User vorzulegen (Frishberg & Lambdin, 2015, S. 29).

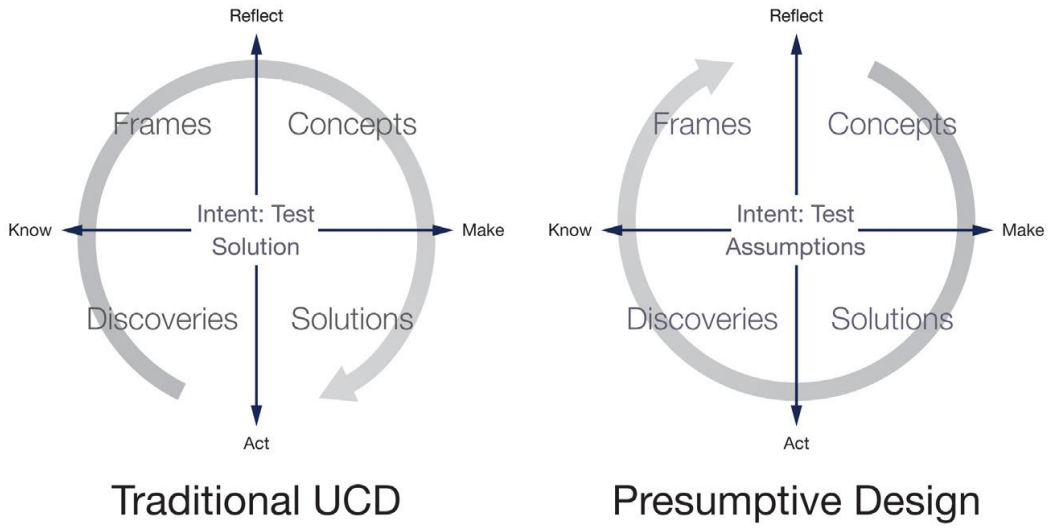


Abbildung 3 – UCD versus PrD (Frishberg & Lambdin, 2015, S.68)

PrD ist ein ständiges Wechselspiel zwischen Wissen und Machen, Kritik, Reflektion und Skizzieren. Es bindet alle Stakeholder von Anfang an mit ein, somit wird von Beginn an ein schnellerer und tieferer Einblick in die erforderlichen Bedürfnisse ermöglicht (Frishberg & Lambdin, 2015, S. 35).

2.4 Abgrenzung zu „Rapid Prototyping“

Viele der obig beschriebenen Ansätze sind ähnlich zu denen des „Rapid Prototyping“. PrD verwendet „Rapid Prototyping“ um die Annahmen der internen Stakeholder gegenüber den Bedürfnissen der externen Stakeholder zu erfassen. Dennoch unterscheiden sich die beiden grundlegend. Während das Ergebnis von „Rapid Prototyping“ eine Lösung für die Designfrage hervorbringen soll, fokussiert sich das „Artefakt“ von PrD auf das Problemfeld der externen Stakeholder. Das „Artefakt“ wird designed, um es nach der „Engagement Session“ zu verwerfen. Es ist also zum Scheitern verurteilt und dies soll keineswegs negativ betrachtet werden (Frishberg & Lambdin, 2015, S.12).

Das erste Prinzip im folgenden Abschnitt behandelt den Umgang mit dem Scheitern des „Artefakts“.

2.5 Die fünf Prinzipien

2.5.1 Design to fail

Dieses Prinzip behandelt das Scheitern nicht als negatives Ereignis, sondern als notwendigen Schritt für Verbesserungen. Dabei ist eine grundlegende Veränderung der Denkweise erforderlich. Es ist notwendig die pessimistische Betrachtungsweise auf eine Optimistische zu verlagern (Frishberg & Lambdin, 2015, S. 65-66). Das Problem einer pessimistischen Betrachtungsweise ist, dass man sich ausschließlich auf die Schäden, die mit einem speziellen Misserfolg verbunden sind konzentriert und dabei das mögliche Problem des gesamten Vorhabens außer Acht lässt. Außerdem besteht für Anhänger der pessimistischen Sichtweise die Gefahr, keines Falls Misserfolge einfahren zu wollen. Dadurch werden wichtige Schritte zur Verbesserung versäumt, deshalb besteht die Gefahr ganze Projekte zum Scheitern zu bringen (Frishberg & Lambdin, 2015, S 59).

Nur mit einer optimistischen Denkweise ist es also möglich Misserfolge kontrollieren zu können, um aus Ihnen durch intelligentes Scheitern Innovationen entstehen zu lassen. Denn Entdeckungen werden oft dann gemacht, während man auf Fehler stößt und anfangs keine Erklärung für diese hat. Anstatt diese Fehler durch Änderung der Lösung zu eliminieren, verändert sich für den Anwender das Verständnis und die Sichtweise des Problems. So können mit Hilfe intelligenten Scheiterns in kürzester Zeit ein riesiges Problemfeld erkundet und verschiedene Möglichkeiten veranschaulicht werden (Frishberg & Lambdin, 2015, S. 65-66). Dabei ist der Prozess des intelligenten Scheiterns viel wichtiger als das Ergebnis selbst (Frishberg & Lambdin, 2015, S. 63).

Als Mitgrund für diese eingebürgerte, noch existierende pessimistische Denkweise nennen die Autoren die Entstehung der Industrialisierung. Die Aufgabe von Ingenieuren und Konstrukteuren war beziehungsweise ist es Fehler unter einen definierten Toleranzbereich zu minimieren um die Funktionsweise bestimmter Prozessketten aufrechterhalten zu können (Frishberg & Lambdin, 2015, S. 59).

Folgendes Zitat veranschaulicht den Umgang mit auftretenden Fehlern, die mit der pessimistischen Denkweise und deren obig angeführten Ursprung einhergehen:

> *„Errors must be identified and reduced, if not eliminated entirely“ (Frishberg & Lambdin, 2015, S. 59).*

Wenn es also eine pessimistische Betrachtungsweise gibt, muss es auch eine Optimistische geben. Am Übergang vom Ersten zum Zweiten bewegt man sich in den Bereich des Designs und der Kunst.

Schulz K. meint dazu folgendes:

> *"the experience of being wrong is hardly limited at all. Surprise, bafflement, fascination, excitement, hilarity, delight: All these and more are a part of the optimistic understanding of error" (Schulz, 2010, S. 27).*

Gerade deshalb ist es umso wichtiger den Entwicklungsweg in optimistischer Denkweise zu gehen, um ein positives Gesamtergebnis entstehen zu lassen, welches Überraschung, Begeisterung, Verblüffung und Heiterkeit mit sich bringt.

2.5.2 Create, discover, analyse

In den Abschnitten 2.1 und 2.3 wurde bereits näher auf den Ablauf von PrD Bezug genommen. Er unterscheidet sich zu anderen Methoden in der Form, dass zuerst ein „Artefakt" erschaffen wird und mit dessen Hilfe die Problemfelder erkundet werden. Bei anderen Design-Methoden wie beispielsweise dem „User Centered Design" werden zuerst Problemfelder erforscht und aus diesen Ergebnissen dann Konzepte und Prototypen entwickelt. Im Gegensatz dazu verwendet PrD das zu Beginn erschaffene „Artefakt" als Vehikel zur Entdeckung des Problemfeldes, und nicht zur Fokussierung auf ein bestimmtes Problem (Frishberg & Lambdin, 2015, S. 79). Die Autoren (Frishberg & Lambdin, 2015) bringen dazu folgende Analogie:

> *„Without this "surveying of the land" we won't know what other hills there are that could have been climbed" (Frishberg & Lambdin, 2015, S. 73).*

Deshalb sind bei PrD das Erschaffen und Erforschen sehr eng miteinander verbunden. Es geht immer um die Reaktionen der externen Stakeholder auf das „Artefakt". PrD zielt also darauf ab, ein tiefes Verständnis über die Ziele und Aufgaben der externen Stakeholder zu bekommen. Es soll dabei nicht die Usability oder das Interface des „Artefakts" an sich getestet werden (Frishberg & Lambdin, 2015, S. 71).

Nachdem ein „Artefakt" in den „Creation Sessions" geschaffen worden ist, wird es den externen Stakeholdern ohne Vorbehalt, Schutz- und Abwehrhaltung vorgelegt. Dabei gilt es für sie, eine bestimmte Aufgabe zu erfüllen. Ohne weitere Führung erkunden sie dadurch das riesige Problemfeld und geben einen tiefen Einblick in ihre Sichtweise (Frishberg & Lambdin, 2015, S.71). Daraus können dann tieferliegende Werte, Bedürfnisse und Gedankenmuster der Zielgruppe entdeckt und somit das „Artefakt" darauf aufbauend verändert werden (Frishberg & Lambdin, 2015, S. 15).

2.5.3 Make assumptions explicit

Wir versuchen die Welt um uns herum zu verstehen, versuchen die umliegende Umgebung zu untersuchen und davon zu lernen. Allerdings betrachten wir die Umgebung durch den Filter unserer Annahmen, die sich aus unseren bisherigen Erfahrungen, unserem Wissen und Können formen. Die Autoren (Frishberg & Lambdin, 2015) vergleichen dies mit folgender Analogie: Wir können die umliegende Landschaft nur durch die Fenster eines Gebäudes erkunden. Da die Betrachtung der Landschaft durch andere Fenster innerhalb des Gebäudes nur einen geringen Mehrwert liefert, ist es also notwendig, das Gebäude und somit unsere Annahmen zu verlassen, um die Landschaft aus frischen, unterschiedlichsten Perspektiven kennen zu lernen, um sie so besser verstehen zu können (Frishberg & Lambdin, 2015, S. 82).

Das Verlassen des Gebäudes beziehungsweise die Minimierung der Filterung durch unsere Annahmen wir bei PrD folgend umgesetzt. In den „Creation Sessions“ wird das Gebäude, also die Annahmen des Teams, in Form eines „Artefakts“ wiedergegeben. Dieses Gebäude wird dann in den „Engagement Sessions“ in die Landschaft der Stakeholder gesetzt, die deren Annahmen repräsentiert. Dadurch können im entstehenden Dialog mit den Stakeholdern die Landschaften und die neuen Sichtweisen auf das Gebäude erkundet werden (Frishberg & Lambdin, 2015, S. 82).

Bei PrD ist es also wichtig nach dem Bau des Gebäudes dieses mit Hilfe der Stakeholder schnell zu verlassen um es aus deren Sichtweisen sehen zu können. PrD erzeugt somit eine Welt, in der tiefe Konversationen und aufmerksames Engagement ausgelöst werden, um dadurch neue Entdeckungen möglich zu machen (Frishberg & Lambdin, 2015, S. 83).

2.5.4 Iterate, iterate, iterate

Der Einsatz von PrD gestaltet sich am Effektivsten, wenn man die billigsten Wege für die Anfertigung des „Artefakts“ und die Durchführung der „Engagement Sessions“ findet, da dadurch die schnellsten und meisten Wiederholungen ermöglicht werden. Dabei ist es wichtig schnell zu erkennen was aus einem Durchgang gelernt wurde, um gleich den nächsten starten zu können. Es ist wichtig zu bedenken, dass mehr Wiederholungen bei ein paar wenigen Leuten schneller mehr Einblicke bringen, als weniger Wiederholungen bei vielen Leuten (Frishberg & Lambdin, 2015, S. 17). So gelingt es mit Hilfe jedes weiteren Durchgangs die Annahmen zu überprüfen und weiterzuentwickeln, das Wissen über die Stakeholder und deren Bedürfnisse zu verbessern und das Problemfeld

indem man sich befindet weiter kennenzulernen (Frishberg & Lambdin, 2015, S. 94).

2.5.5 The faster we go, the sooner we know

PrD fordert den Anwender auf sofort zu starten, unverzüglich. Je schneller man beginnt, desto schneller kann man intelligent scheitern, daraus lernen und wieder scheitern. Und das immer und immer wieder, um durch jeden Zyklus näher an die Vorstellungen und Bedürfnisse der Stakeholder heranzukommen und das so schnell wie möglich (Frishberg & Lambdin, 2015, S. 108). Je länger das Treffen mit Stakeholdern hinausgezögert wird, desto später bekommt man Einblicke in deren Bedürfnisse und desto weniger Chancen bekommt man auf weitere Treffen, die wiederum weitere Einblicke liefern (Frishberg & Lambdin, 2015, S. 109).

2.6 PrD in dieser Arbeit

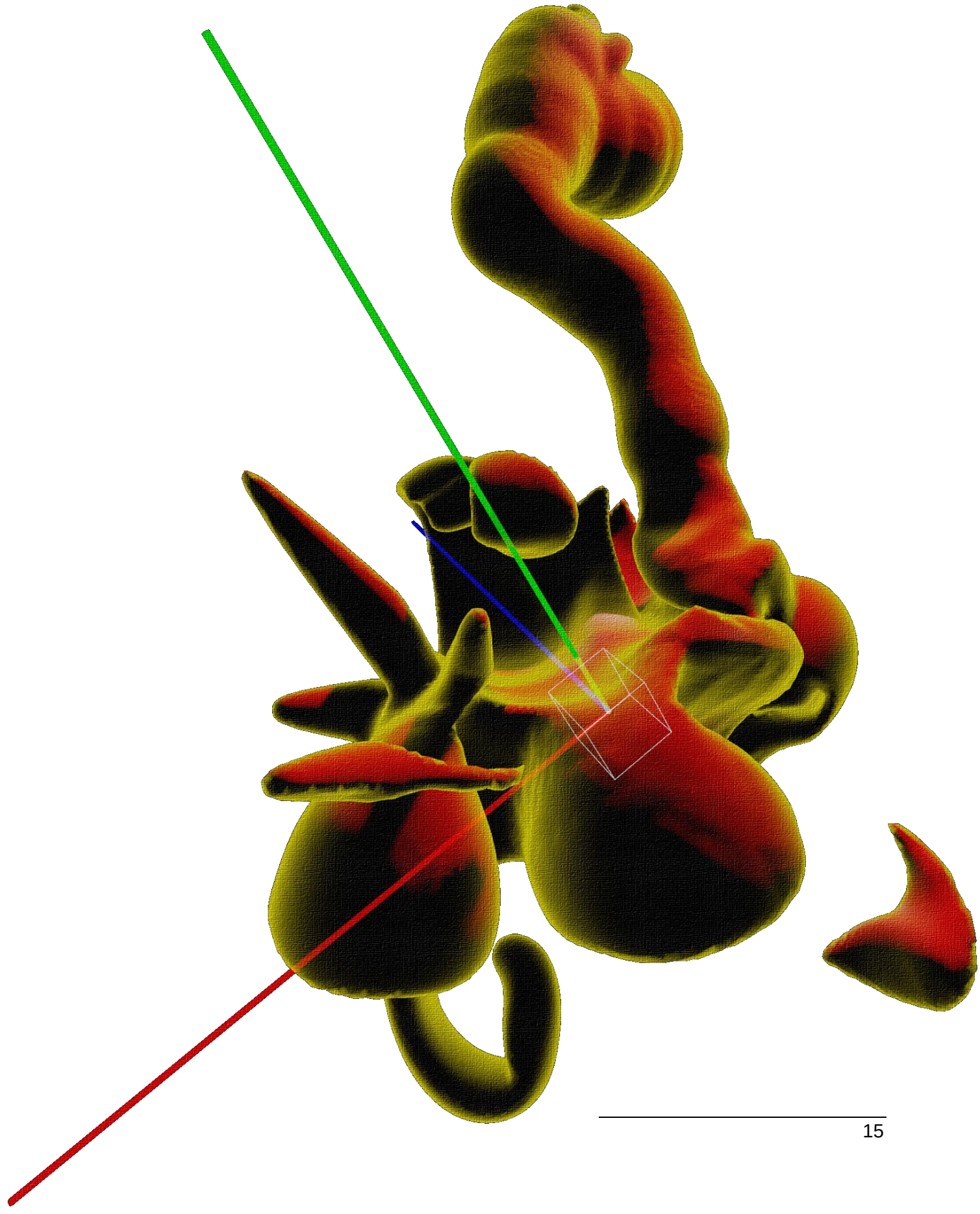
Im bisherigen Teil von Kapitel 2 wurde näher auf die Methodik von Presumptive Design eingegangen. Wichtige Prinzipien und Definitionen wurden behandelt und es wurde auf den Ablauf Bezug genommen.

In dieser Arbeit wird die Methode Presumptive Design benutzt um die Zielgruppe von Beginn an mit einzubeziehen und um den Entwicklungsprozess der Installation gemeinsam zu gehen. Allerdings gibt es einige Adaptionen zu den obig beschriebenen Begriffen, die vor allem mit der Projektgröße zusammenhängen. Im Buch (Frishberg & Lambdin, 2015) wird meist davon ausgegangen, dass Problemstellungen innerhalb von großen Firmen gelöst werden sollen. Da die Ressourcen dieser Arbeit nicht mit denen von großen Firmen vergleichbar sind, werden die Begriffe im folgenden Absatz klarer definiert.

Der interne Stakeholder dieser Arbeit ist dessen Verfasser. Dieser gestaltet in den „Creation Sessions“ „Artefakte“, die seinen Vorstellungen über das Wesen der Installation am nächsten sind. In regelmäßigen Abständen werden diese „Artefakte“ in den „Engagement Sessions“ den externen Stakeholdern vorgelegt. Die Zielgruppe und somit auch externen Stakeholder dieser Installation sind Menschen im persönlichen Umfeld des Verfassers. Einige haben technischen Hintergrund, andere nicht. Jedoch soll für beide ein immersives Erlebnis geschaffen werden. Erste „Artefakte“ können Skizzen oder Visualisierungen der Installation sein. Diese werden Mitstudierenden, Lektoren oder anderen Personen im Umfeld des Verfassers gezeigt. Deren Reaktionen bilden Grundlage für Weiterentwicklungen am „Artefakt“ und der Arbeit an sich. Dieser Zyklus wird so

oft wiederholt bis die zur Verfügung stehende Zeit ausgeht. Als fertiger Prototyp wird eine bewegliche Videoinstallation innerhalb einer Black Box erwartet. Allerdings kann sich der erwartete Entwicklungsweg aufgrund der Reaktionen verändern und somit kann sich das erwartete Ergebnis in viele Richtungen entfalten. Die Visionen und möglichen Wege sowie die umgesetzten Schritte werden in Kapitel 4 dokumentiert. Diese Änderungen können beispielsweise die Umgebung, die Anordnung, den Content und die Deformation betreffen.

Das nächste Kapitel 3 beschäftigt sich mit ähnlichen Projekten, die der Vision in dieser Arbeit nahekommen. Deren Hintergründe sowie Details zur Umsetzung werden behandelt. Sie sollen einerseits das Hineindenken in diese Arbeit erleichtern, andererseits werden sie verwendet um zu gewissen Punkten zu referenzieren und Vergleiche zu bestimmten Parametern zu machen.



3 Relevante Projekte im fachlichen Umfeld

Dieses Kapitel behandelt drei verschiedene bereits umgesetzte Projekte, die Ähnlichkeiten zur Vision in dieser Arbeit haben. Das erste von den drei Projekten wurde bereits in den 1930er Jahren umgesetzt. Alexeieff und Parker entwickelten eine Apparatur, die es mit Hilfe von vielen Stäbchen und Licht erlaubt animierte Cartoons zu erstellen. Beim nächsten Projekt, im Abschnitt 3.2, geht es um eine interaktive Installation in einem Kaffeeladen, die aus vielen Kacheln aufgebaut ist. Das letzte Projekt im Abschnitt 3.3 ist die Arbeit „MegaFaces“ von Asif Khan, welches für die Darstellung von riesigen 3D-Gesichtern bei den Olympischen Winterspielen im Jahr 2014 verwendet wurde. Dieses Kapitel soll eine grundlegende Wissensbasis liefern, um das Hineindenken in diese Arbeit zu erleichtern. Außerdem wird immer wieder auf einige Parameter in diesen Arbeiten Bezug genommen und Vergleiche gemacht.

3.1 Alexandre Alexeieff & Claire Parker – Apparatus for producing images

Unter dem Namen „Apparatus for producing images“ wurde 1934 ein Patent von Claire Parker veröffentlicht. Dieser Apparat dient zur Herstellung von Bildern für die Produktion von animierten Cartoons (US2100148A, 1937).

Diese Apparatur entstand aus der Zusammenarbeit zwischen Alexandre Alexeieff und Claire Parker. Alexeieff arbeitete zuvor viel mit Xylographie und Gravuren, die ihm Inspiration für die folgende Arbeit waren (Neupert, 2011, S. 82). Um 1930 wechselte er von statischen Illustrationen zu Bewegtbild und lerne so Claire Parker kennen (Neupert, 2011, S. 83). Zusammen entwickelten sie den ersten Pinscreen und es entstand die sogenannte „Pinscreen Animation“ (Neupert, 2011, S. 84). Abbildung 4 zeigt die beiden beim gemeinsamen Arbeiten mit dieser Apparatur.

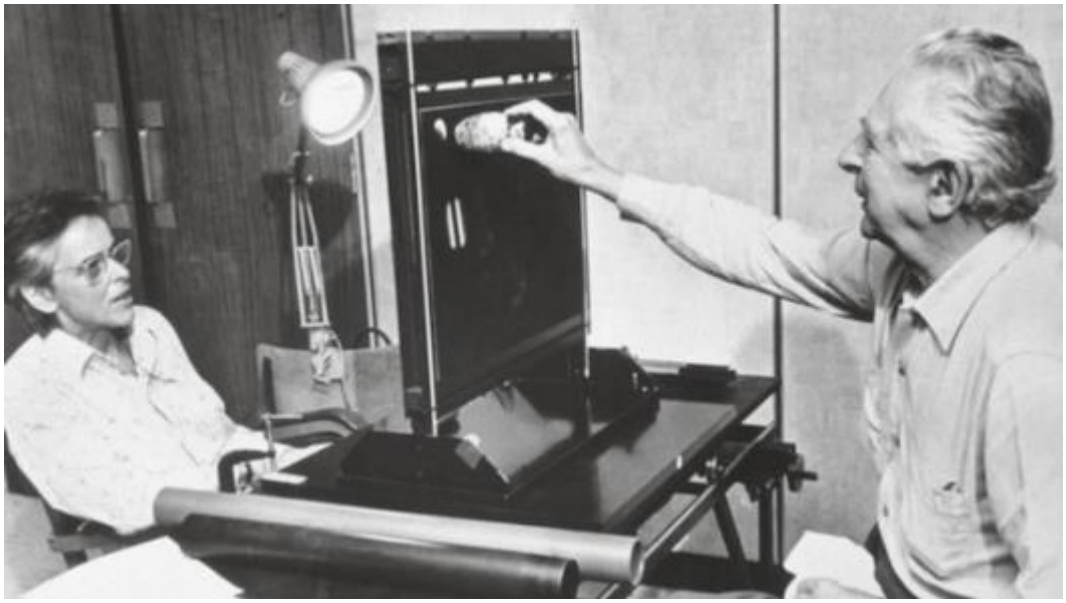


Abbildung 4 – Parker und Alexeieff (Selvedge, 2018)

Bei dieser Erfindung handelt es sich um eine Apparatur, die viele kleine Stäbchen in Position hält. Wie links in Abbildung 5 ersichtlich, formen die Spitzen dieser Stäbchen gemeinsam eine zweidimensionale Fläche. Der rechte Bereich in Abbildung 5 zeigt diese Apparatur von der Seite. Man erkennt, dass jedes dieser Stäbchen weiter raus oder rein gedrückt werden kann (US2100148A, 1937).

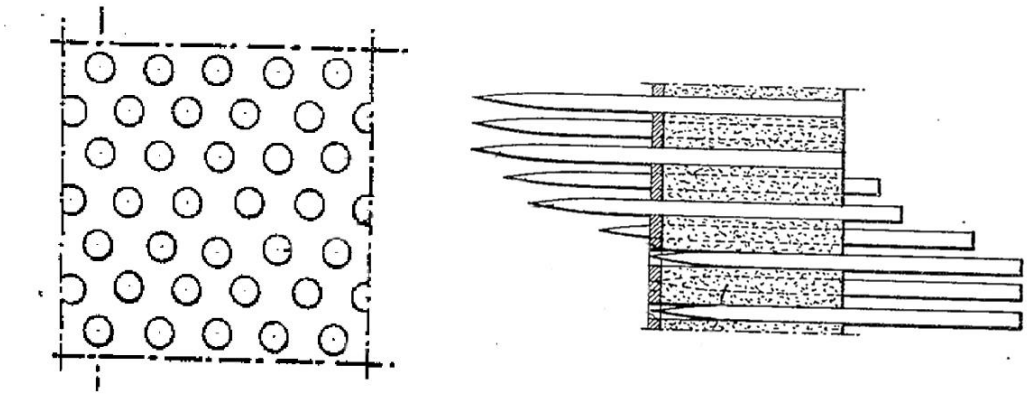


Abbildung 5 – Front- und Seitenansicht „Apparatus for producing images“ (US2100148A, 1937)

Wird diese Fläche nun schräg beleuchtet, bilden sich, abhängig von der herausragenden Weite jedes Stäbchens, unterschiedlich große Schatten ab. Beim Beispiel in Abbildung 6 wird die Fläche von vier unterschiedlichen Lichtquellen beleuchtet. Das linke Stäbchen ist weit rausgeschoben und wirft somit einen

großen Schatten, das rechte Stäbchen ist nur etwas rausgeschoben und wirft einen kleinen Schatten (US2100148A, 1937).

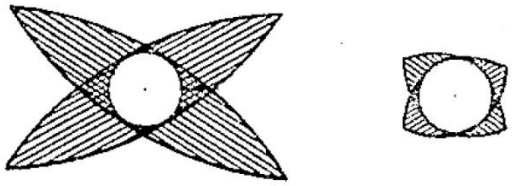


Abbildung 6 – Unterschiedliche Schattenbildung (US2100148A, 1937)

Diese Tatsache macht es also möglich, mit der Position jedes einzelnen Stäbchens und der daraus resultierenden Schattenbildung, ein Bild zu malen. Im Gegensatz zu klassischer Farbe ist es mit dieser Apparatur ohne weitere Probleme möglich Korrekturen durchzuführen, in dem einfach die Positionen einzelner Stäbchen verändert werden können. Diese sind so fixiert, dass sie sich mit etwas Druck verschieben lassen, aber fest genug sitzen um bei eventueller Erschütterung ihre Position zu halten (US2100148A, 1937).

Der beleuchtete Apparat kann mit einer Kamera abfotografiert werden, um so das auf der Platte entstehende Schwarz-Weiß-Bild einzufangen. Nach dem Abfotografieren können Änderungen an den Stäbchen vorgenommen werden und dieses neu entstehende Bild kann wieder eingefangen werden, und so weiter. Nach Aneinanderreihung dieser einzeln aufgenommenen Bilder, entsteht die Illusion einer Bewegung, ähnlich wie bei Stop-Motion-Filmen (US2100148A, 1937).

Für das „Malen“ dieser Bilder, beziehungsweise das Verschieben der Stäbchen können unterschiedlichste Werkzeuge verwendet werden, je nach dem was für eine Ästhetik geschaffen werden soll und wie viele Stäbchen auf einmal bewegt werden sollen. Im Patent von Claire Parker sind zum Beispiel Spachteln verschiedener Größe und zylindrische Rollen angeführt. Abbildung 7 zeigt Skizzen einer Walze, mit der eher flächige Strukturen und Verläufe „gezeichnet“ werden können, und einer Rolle, mit der bestimmte Muster umgesetzt werden können (US2100148A, 1937).

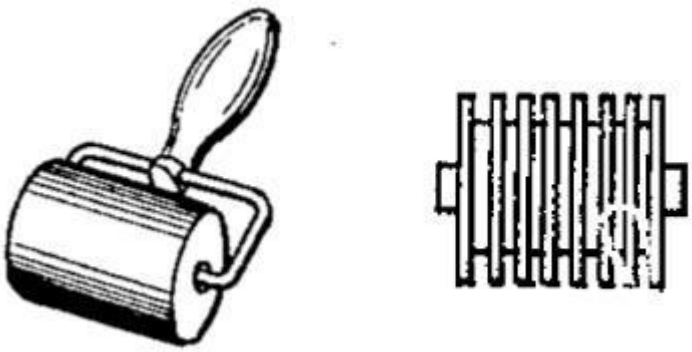


Abbildung 7 – Werkzeuge „Apparatus for producing images“ (US2100148A, 1937)

Nachdem Claire Parker und Alexandre Alexeieff deren erste Apparatur fertiggestellt hatten und alle Prozesse optimiert hatten, produzierten sie 1933 innerhalb von 18 Monaten ihren ersten Film „Night on Bald Mountain“[1]. Dieser hat eine Laufzeit von 8 Minuten. Es existierten dabei keinerlei Storyboards oder ähnliche Leitfäden. Alexeieff übersetzte direkt seine Visionen mit Parkers Hilfe auf deren Pinscreen um. Wie in obiger Abbildung 4 zu sehen ist, saß Alexeieff auf der einen und Parker auf der anderen Seite des Pinscreens, sodass jeder die Stäbchen zum anderen drücken konnte und dies in konstanter kreativer Interaktion resultierte (Neupert, 2011, S. 84). Abbildung 8 zeigt ein Bild aus „Night on Bald Mountain“. Durch den Einsatz des Pinscreens, entsteht eine eigene Ästhetik. Es ergibt sich ein Schwarz-Weiß-Bild indem Strukturen eines Rasters erkennbar sind.

[1] https://www.youtube.com/watch?v=wYbjW7XrWDo

Abbildung 8 – Night on Bald Mountain, Alexeieff & Parker (2ndviolinist, 2013)

Im Patent von Claire Parker (US2100148A, 1937) sind neben den bereits gebauten und erprobten Systemen auch einige interessante Visionen angeführt. Beispielsweise werden neben der klassischen Nutzung als Instrument für die Produktion von Animationen auch die Einsatzmöglichkeiten bezüglich Drucktechnik und Fotogravur erwähnt. Außerdem wird die Möglichkeit zur automatischen Positionierung der Stäbchen vorgestellt. Abbildung 9 zeigt die schematische Darstellung einer möglichen Umsetzung. Hierbei soll mit Hilfe zweier Elektromagneten die Position des Stäbchens gesteuert werden. Die Ansteuerung der gesamten Apparatur, könnte mit Hilfe eines klassischen analogen Fernsehgeräts oder ähnlichen erfolgen. Dabei wird jeder Elektromagnet zu bestimmten Zeitpunkten mit der richtigen Stromstärke versorgt, um die Stäbchen gesteuert positionieren zu können (US2100148A, 1937). So könnte beispielsweise ein Wiedergabegerät entstehen, dass die Helligkeitsinformationen eines Fernsehbilds mechanisch in Tiefenpositionen der Stäbchen übersetzt.

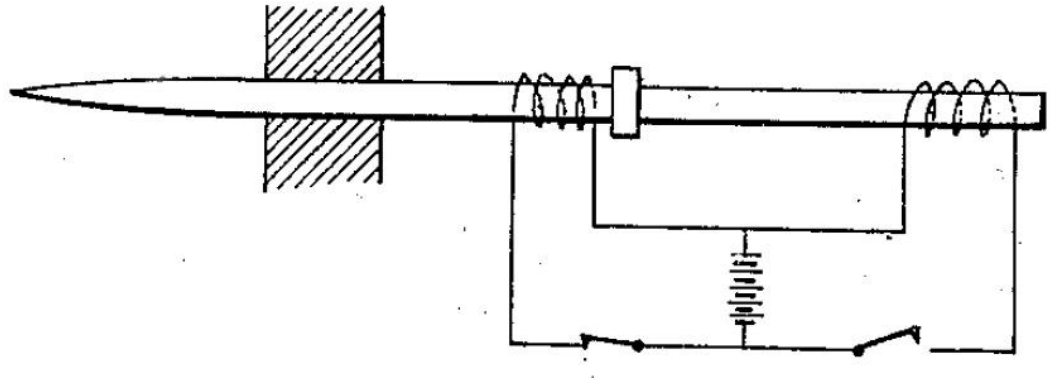


Abbildung 9 – Automatische Positionierung „Apparatus for producing images“

3.2 Daniel Rozin - Last Chance to Shine

Die in diesem Abschnitt besprochene Arbeit ist ähnlich zu der Vorherigen. Bewegungen und Licht spielen eine wichtige Rolle. Diese Arbeit liegt weniger weit zurück, sie nutzt deshalb wesentlich moderne Technik um ihr Ziel zu erfüllen.

Der Künstler Daniel Rozin arbeitet im Feld der interaktiven digitalen Kunst. Seine Installationen und Skulpturen verändern sich und reagieren meist auf die Anwesenheit des Betrachters und laden ihn ein, ein Teil des Werks zu werden. Oft arbeitet er mit Spiegeln oder Ähnlichem, um den Betrachter reflektiert und verändert abzubilden. Er unterrichtet außerdem an der „Tisch School of the Arts“ in New York City. Abbildung 10 zeigt ein paar seiner Werke. Er arbeitet meist mit einer großen Anzahl an Elementen. Beispielsweise verwendet er reflektierende Teile wie Spiegel, Troll oder Pinguin Puppen, die durch Drehung auf den Betrachter reagieren („Daniel Rozin Interactive Art“, 2018).

Abbildung 10 – Auswahl Arbeiten von Daniel Rozin („Daniel Rozin Interactive Art“, 2018)

Das in diesem Abschnitt besprochene Werk „Last Chance to Shine“[2] entstand als Auftragsarbeit für die Kaffeemarke Nespresso. Es dient als repräsentatives Stück für einen Vorzeigeladen in Manhattan. In Abbildung 11 sieht man die fertig installierte, runde Skulptur mit einem Durchmesser von etwa 2,4 Metern (Plaugic, 2018).

Abbildung 11 – „Last Chance to Shine“ im Nespresso Flagship-Store (Plaugic, 2018)

Diese Skulptur besteht aus 832 quadratischen Kacheln. Wie der Name „Last Chance to Shine“ referenziert, sind diese Kacheln aus Aluminiumresten der Nespresso-Kapsel-Produktion hergestellt. Wie in Abbildung 12 ersichtlich, schnitt Rozin dafür Streifen dieser Aluminiumreste zurecht und presste diese mit einer eigens angefertigten Form zu Quadraten. Für die Farbgestaltung verwendete er Aluminium in Blaugrün, Orange, Silber und Schwarz. An der Rückseite jeder Kachel ist eine Halterung angebracht. Diese dient zur Befestigung an einem Schrittmotor, mit dem die Neigung jeder Kachel gesteuert werden kann (Plaugic, 2018).

[2] https://www.theverge.com/2018/1/9/16850280/daniel-rozin-interactive-art-interview-video-nespresso-last-chance-to-shine

Abbildung 12 – Kachel „Last Chance to Shine“ (Plaugic, 2018)

Rozin verwendet für jede Kachel einen Schrittmotor, der es entlang einer Achse neigbar macht. Insgesamt umfasst die Skulptur also 832 Schrittmotoren. Diese werden insgesamt von 52 Controllern gesteuert, wobei jeder Controller jeweils 16 Schrittmotoren regelt. In Abbildung 13 ist ein Modul mit einem Controller und 16 Schrittmotoren zu sehen. 52 dieser Module formen sein gesamtes Werk (Plaugic, 2018).

Abbildung 13 – Schrittmotor „Last Chance to Shine“ (Plaugic, 2018)

Mit den Überlegungen und der Entwicklung der Bewegungen startete Daniel Rozin nach der Fertigstellung der Skulptur. Wie auch bei vielen seiner vergangenen Arbeiten spielt hier die Interaktion mit dem Betrachter eine große Rolle. Als „Sinnesorgane“ der Skulptur verwendet er vier passive Infrarot-Sensoren (PIRs), siehe Abbildung 14 rechts. Diese Halbleitersensoren dienen dazu Temperaturveränderungen wahrzunehmen und diese elektrisch weiterzugeben. Rozin bringt diese Sensoren so an, dass jeder von den vieren einen bestimmten Bereich abdeckt. Dadurch ist es möglich Personen in der Nähe der Skulptur zu detektieren und Bewegungen auswerten zu können. Eine große Herausforderung laut Rozin war es, aus einer so geringen Menge an Informationen eine reichhaltige Ausgabe zu schaffen. Es galt also aus Informationen wie „Person ist hier“ und „Person bewegt sich dort hin“ komplexe und beeindruckende Bewegungsmuster zu gestalten (Plaugic, 2018).

Abbildung 14 – Daniel Rozin & PIR „Last Chance to Shine“ (Plaugic, 2018)

Die Kacheln erzeugen durch Kontakt mit der Platte an den Grenzen ihrer Bewegungen ein Klackern. Das Klackern aller 832 Kacheln erzeugt den Eindruck eines sanften Rauschens, ähnlich dem eines Regenstabs. Wie in Abbildung 14 links zu sehen ist es mit Rozin’s Skulptur möglich komplexe Bewegungsverläufe umzusetzen, die in beeindruckenden Lichtreflexionen resultieren (Plaugic, 2018).

3.3 Asif Khan – MegaFaces

Das Projekt „MegaFaces“[3] entstand im Zusammenhang mit den Olympischen Winterspielen in Sotschi im Jahr 2014. Es wurde dort für einen Monat ausgestellt. Der Auftraggeber war der russische Telekommunikationsbetreiber MegaFon. Das Werk wurde vom Londoner Architekten Asif Khan konzipiert. Dabei handelt es sich um eine bewegliche Fassade am MegaFon Pavillon, welche es ermöglicht dreidimensional menschliche Gesichter zu rekonstruieren. Abbildung 15 zeigt diese Arbeit. Dessen Aufbau erinnert stark an den des in Abschnitt 3.1 beschriebenen Pinscreens von Alexeieff und Parker. Viele Elemente bilden durch lineare Verschiebung eine dreidimensionale Form im Raum ab. Das Projekt „MegaFaces“ wurde innerhalb eines Jahres umgesetzt (iart, 2014).

[3] https://iart.ch/en/-/die-kinetische-fassade-des-megafaces-pavillons-olympische-winterspiele-2014-in-sotschi

Abbildung 15 – MegaFaces (iart, 2014)

Entwickelt wurde „MegaFaces“ vom Schweizer Studio iart bis ins kleinste Detail. Ziel dieser Installation war es 3D-Selfies der Besucher und Sportfans riesig abzubilden. Umgesetzt wurde dies mit einer eigens entwickelten Fassade mit einer Größe von 18 mal 8 Metern. Diese besteht aus 11.000 teleskopierbaren Zylindern, auch Aktuatoren genannt. An der Spitze jedes Aktuators ist eine lichtdurchlässige Kugel angebracht, welche von der Innenseite durch eine RGB-LED beleuchtet werden kann. Abbildung 16 zeigt die von innen beleuchteten Kugeln an der Fassade (iart, 2014).

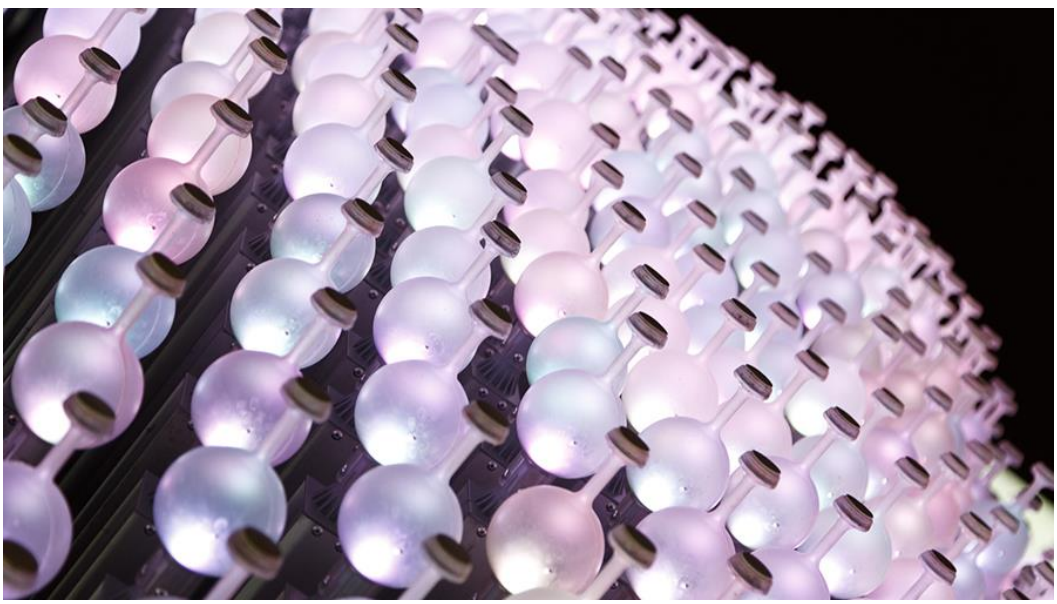


Abbildung 16 – RGB-LEDs „MegaFaces“

Der Auszug jedes Aktuators kann zwischen 0 und 2 Metern elektrisch positioniert werden. Jede Einheit, bestehend aus Aktuator und RGB-LED, repräsentiert ein Pixel der gesamten Fassade. Da jede Einheit einzeln angesteuert werden kann, ist es, wie in Abbildung 15 ersichtlich, der gesamten Fassade möglich die Form und Farbe von 3D-Gesichter nachzubilden (iart, 2014).

Im Zeitraum der Olympischen Winterspiele wurden minütlich Fotoscans der Besucher gemacht. Für die Aufnahmen wurden von iart eigene Fotokabinen entwickelt, in denen sich die Besucher fotografieren lassen konnten. Sieben dieser Fotokabinen wurden direkt im MegaFon Pavillon installiert und weitere auf öffentlichen Plätzen in 30 Städten in ganz Russland aufgestellt. In Abbildung 17 ist das Team bei der Installation der Fotokabinen zu sehen.

Abbildung 17 – Installation Fotokabinen „MegaFaces“ (iart, 2014)

Für die Übersetzung der Gesichter in den 3D-Raum wurden 5 Kameras verwendet. Diese erzeugen zum selben Zeitpunkt aus unterschiedlichen Perspektiven Bilder, aus denen durch Triangulation auf das 3D-Modell gerechnet werden kann. Die erzeugten 3D-Modelle wurden automatisch in Steuerdaten für die Installation konvertiert und in die Timeline eingefügt. Die aufgenommenen Gesichter wurden so nach und nach auf der Fassade wiedergegeben. Die Teilnehmer wurden per SMS verständigt, wann ihre Gesichter auf der Fassade zu sehen seien werden. Auf der riesigen Fassade fanden zeitglich jeweils 3 Gesichter nebeneinander Platz, wobei jedes eine Höhe von 8 Metern einnahm (iart, 2014).

Wie am Beginn dieses Abschnitts erwähnt hat diese Arbeit große Ähnlichkeiten zu dem Pinscreen von Alexeieff und Parker in Abschnitt 3.1. Im Patent von Parker (US2100148A, 1937) wurde die Möglichkeit erwähnt, die Positionen der Stäbchen mit Hilfe von Elektromagneten zu steuern, um die Form und damit die

Schattenbildung der gesamten Apparatur zu automatisieren. Im Projekt „MegaFaces“ wurde diese Vision umgesetzt. Die Aktuatoren von „MegaFaces“ repräsentieren hier die Stäbchen des Pinscreens. Die beiden Arbeiten unterscheiden sich allerdings enorm in ihrer Größe. Wie in Abbildung 4 zu sehen, misst der Pinscreen lediglich ein paar Dezimeter, die Fassade von „MegaFaces“, wie in Abbildung 18 ersichtlich, allerdings 18 mal 10 Meter.

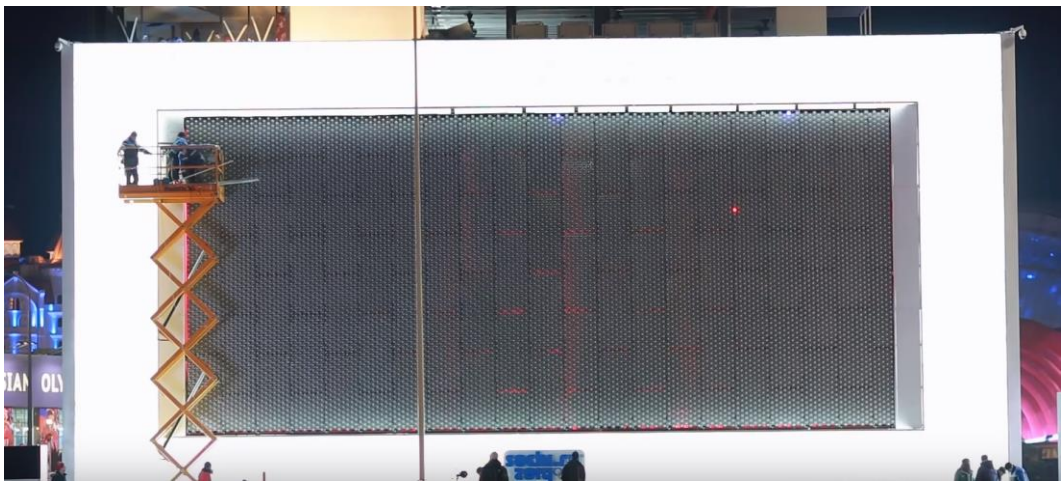


Abbildung 18 – Installation Fassade „MegaFaces“

Das Team von iart ging im Entwicklungsprozess sukzessive mit der Anzahl der Aktuatoren und der damit verbundenen Größe um. Sie starteten mit Tests einzelner Aktuatoren und steigerten immer wieder die Anzahl. Sie experimentierten auch, wie in Abbildung 19 zu sehen, mit beweglichen Stoffen, die die Anmutung zu menschlichen Gesichtern enorm erhöhen. Bei der finalen Fassade, die bei den Olympischen Winterspielen im Einsatz war, wurden jedoch keine zusätzlichen Stoffe für die Oberflächengestaltung verwendet (iart, 2014).

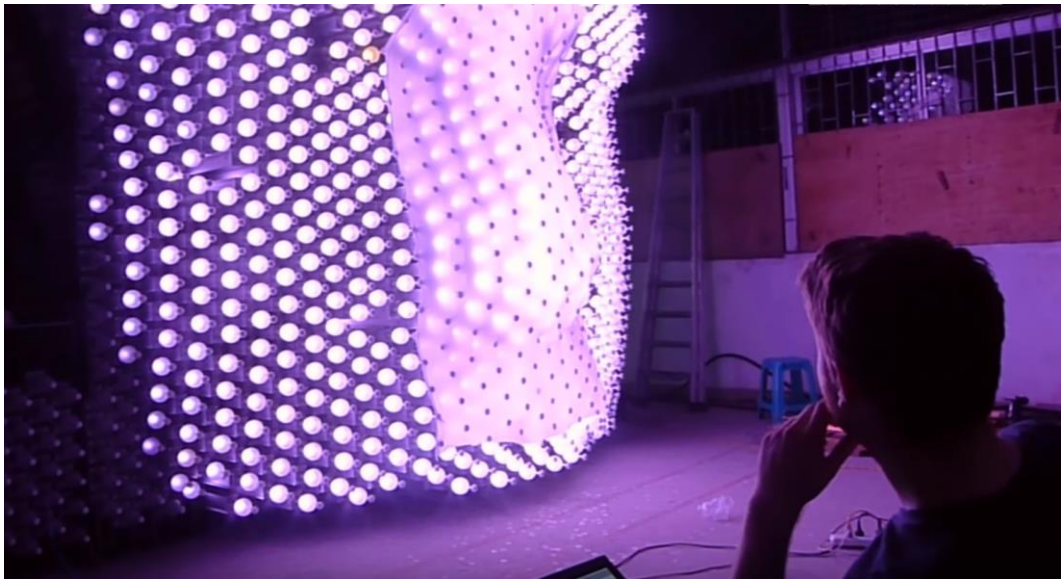
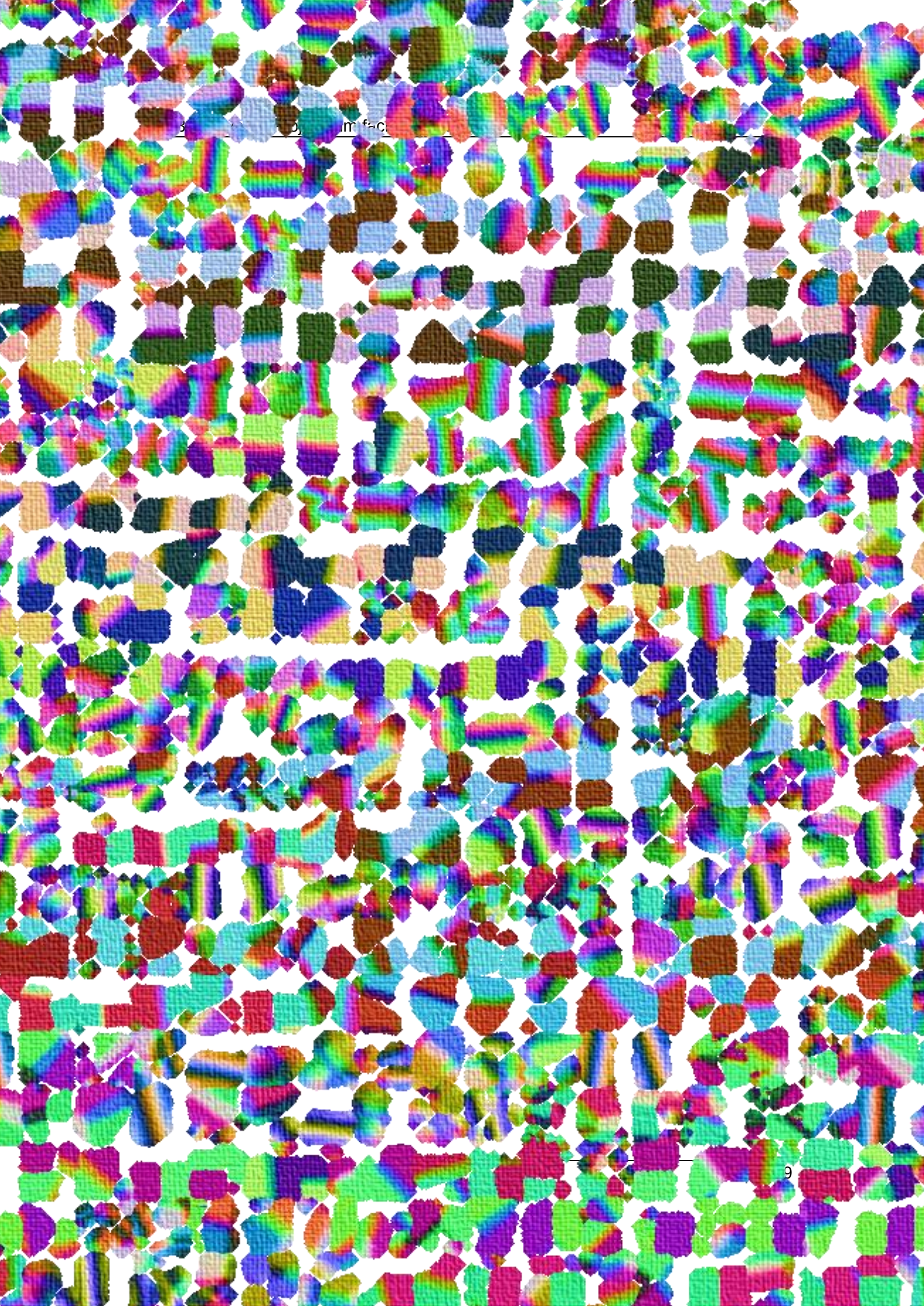


Abbildung 19 – Vortests „MegaFaces“



4 Der Entwicklungsweg

Dieses Kapitel beschäftigt sich mit der Dokumentation des Entwicklungsweges vom Beginn der Vision bis zur Umsetzbarkeit der fertigen Installation. Einzelne Entwicklungsschritte werden aufgelistet und auf deren Hintergründe eingegangen. Wie bereits in Abschnitt 2.6 behandelt werden hier die einzelnen „Artefakte“ vorgestellt. Der Entstehungsprozess spielt eine ebenso große Rolle wie die Reaktionen der externen Stakeholder. Die Ergebnisse jedes Durchlaufs werden festgehalten und mögliche Umsetzungen dokumentiert.

Insgesamt wurden bis zur Fertigstellung der Installation 13 „Artefakte“ kreiert und 10 „Engagement Sessions“ abgehalten. Insgesamt nahmen 26 verschiedene externe Stakeholder an den „Engagement Sessions“ teil. Unter den externen Stakeholdern waren Mitstudenten, Lektoren, Freunde und Familienmitglieder. Der Weg startet zu Beginn nur mit der Vision. Währenddessen kommt immer wieder Input von außen durch die „Engagement Sessions“ oder durch andere, bereits umgesetzte Projekte, hinzu. Die externen Stakeholder haben das Konzept in unterschiedlichste Richtungen getrieben und somit viele Möglichkeiten aufgezeigt. Ganz nach dem folgenden, bereits in Abschnitt 2.5.2 vorgestellten, Zitat.

> *„Without this “surveying of the land” we won’t know what other hills there are that could have been climbed“ (Frishberg & Lambdin, 2015, S. 73).*

Jeder der folgenden Abschnitte behandelt jeweils die Erstellung des „Artefakts“ und die Reaktionen der externen Stakeholder in den „Engagement Sessions“. Am Ende jedes Abschnittes ist eine farbige Tabelle aufgelistet, welche jeden Zyklus zusammengefasst darstellt.

4.1 Interaktives Nagelbrett

Das erste „Artefakt“ entstand als Handskizze auf Papier, noch lange vor dem Beginn dieser Arbeit. Abbildung 20 zeigt eine Kopie dieser Skizzierung.

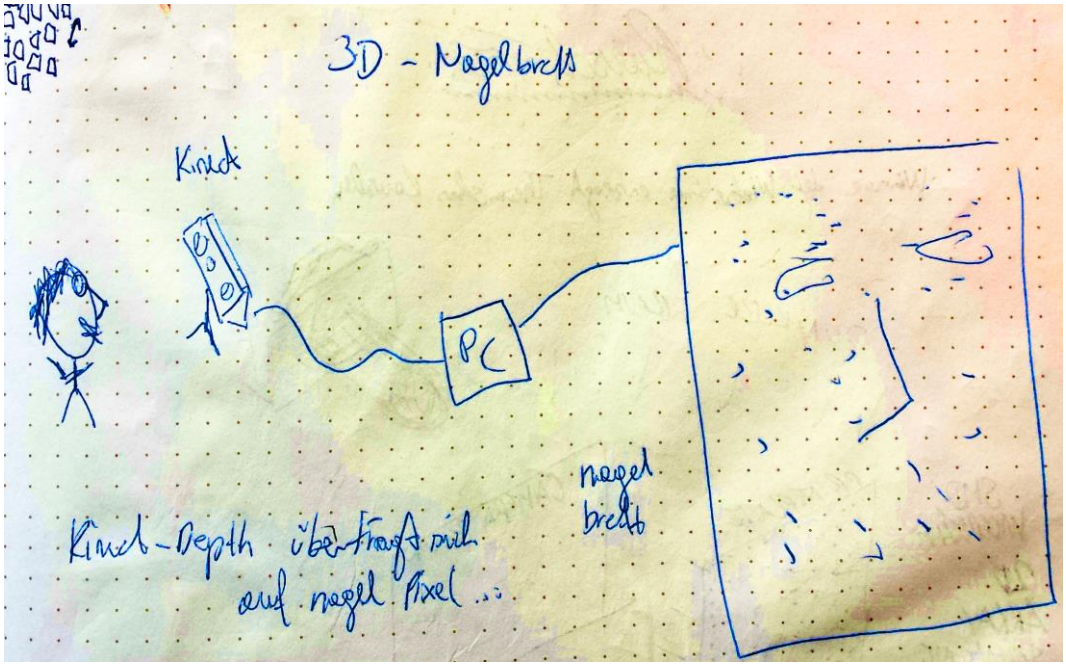


Abbildung 20 – Artefakt 1: Handskizze „Interaktives Nagelbrett“

Die Grundidee hinter dieser Skizze ist es mit Hilfe einer Kinect[4] die physische Umgebung in einem Raum abzuscannen. Die Kinect liefert unter anderem Informationen über die Tiefe des Bildes. Das bedeutet sie liefert Daten darüber wie weit Objekte in ihrer unmittelbaren Umgebung entfernt sind. Dadurch ist es möglich auf die Form von beispielsweise Personen rückzuschließen. In der linken Hälfte von Abbildung 21 ist die Information über die Tiefe auf ein Schwarz-Weiß-Bild gemappt. Schwarz wäre hier weit entfernt, Weiß hingegen sehr nah. Diese gewonnene Tiefeninformation soll nach Verarbeitung in einem PC auf eine Art Nagelbrett, wie rechts in Abbildung 21 ersichtlich, übertragen werden. Somit würde der Benutzer direkt durch Bewegungen das Aussehen und die Form des Nagelbretts steuern können.

Der Ablauf beim Benutzen eines klassischen Nagelbretts mit dem Gesicht ist meist folgender. Der Benutzer drückt sein Gesicht an der Rückseite ins Nagelbrett hinein, zieht es langsam ohne es zu verändern wieder heraus und betrachtet die entstandene Form auf der Vorderseite. Dabei vergehen einige Sekunden von der „Eingabe“ bis zur „Ausgabe“, sie sind also aufeinander aufbauend.

Die Feedbackschleife zwischen Machen und Betrachten beim interaktiven Nagelbrett hingegen wird kurzgeschlossen. Die Gesichts- und Körperform des

[4] https://de.wikipedia.org/wiki/Kinect

Benutzers wird hier unmittelbar auf das Nagelbrett übertragen. Der Anwender bekommt also sofort Feedback über seine Eingabe. Er tritt ins interaktive Zusammenspiel mit der Installation.

Abbildung 21 – Tiefenbild Kinect (Kinect, 2018) & Klassisches Nagelbrett (Nagelbrett, 2018)

Dieses grobe Blockschaltbild aus Abbildung 20 soll das ungefähre Zusammenspiel zwischen Benutzer und Installation verdeutlichen. Detailliertere Pläne zur Umsetzung des interaktiven Nagelbretts sind hier noch nicht ausgearbeitet. Der nach dem Anfertigen dieser Skizze entdeckte Pinscreen von Alexeieff und Parker würde hier Ansätze zur Lösung bieten. Wie das im Abschnitt 3.1 behandelte Patent (US2100148A, 1937) meint, könnten die einzelnen Nägel durch Elektromagneten und durch entsprechender Ansteuerung automatisch positioniert werden, vgl. Abbildung 9.

Nach bevor dieses erste „Artefakt“ externen Stakeholdern in „Engagement Sessions“ vorgelegt worden ist, wurde es weiterentwickelt. Die Weiterentwicklung entstand ebenfalls als Handskizze auf Papier. Sie versteht sich als Erweiterung des ersten in Abbildung 20 angeführten „Artefakts“. Abbildung 22 zeigt die Skizze dieser Erweiterung.

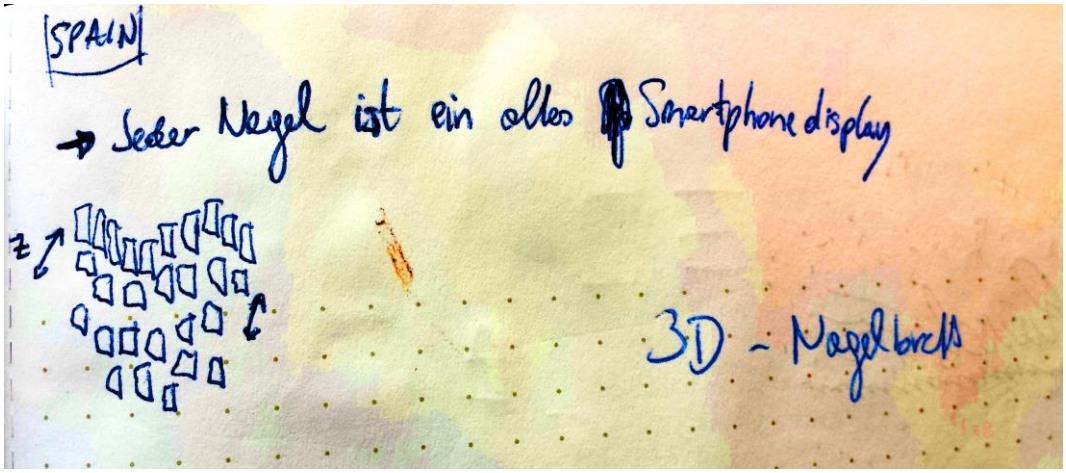


Abbildung 22 – Artefakt 2: Handskizze „Interaktives Displaybrett“

Die Idee dahinter ist es die zuvor besprochene Installation bestehend aus Kinect und Nagelbrett mit alten Smartphone Displays zu erweitern. An jeder Nagelspitze soll bei dieser Idee ein Display angebracht werden. Genauso wie bei der zuvor besprochenen Idee passt sich die Form des Nagelbretts an die erfasste Form des Benutzers an. Jedes Display kann sich innerhalb einer Achse vor und zurückbewegen. Außerdem kann Inhalt auf den Displays wiedergegeben werden. Durch die Erweiterung mit Displays gibt es eine enorme Änderung der Auflösung. Beim „Artefakt 1“ können die einzelnen Nägel aufgrund ihres kleinen Durchmessers eng aneinander positioniert werden und es ergibt sich eine relativ feine Auflösung. Beim „Artefakt 2“ hingegen verringert sich durch die Verwendung von Displays die Anzahl der Elemente pro Fläche, da jedes Display mehr Platz beansprucht als ein einzelner Nagel. Dadurch würde sich die Auflösung durch die Elemente verringern. Auf der anderen Seite ergibt sich durch die Möglichkeit der Wiedergabe von Bewegtbildern ein Mehrwert.

Diese beiden „Artefakte“ wurden als Grundlage für die erste „Engagement Session“ verwendet. Tabelle 1 listet die zusammengefassten Fakten dieser Session auf. Es wurden dabei die beiden „Artefakte“ einzeln zwei verschiedenen externen Stakeholdern vorgelegt. Außerdem wurden die obig beschriebenen Ideen dahinter zusätzlich erklärt. Deren Ziel war es sich aufgrund der Skizzen die fertige Installation und deren Interaktion vorzustellen. Dabei kamen einige Reaktionen der externen Stakeholder. Einerseits gab es mehr Zuspruch für das „Artefakt 2“ mit Displays, da sich hier ein größerer Gestaltungsspielraum ergibt als bei der Version ohne Displays. Andererseits war beiden externen Stakeholdern die Skizze in Abbildung 22 zu abstrakt um sich einen klaren Aufbau und eine klare Interaktion vorstellen zu können. Dafür entstanden aufgrund der Unklarheiten einigen Fragen, die richtungsgebend für den weiteren Entwicklungsweg sind: Wie

verhält sich diese Installation? Was ist auf den Displays zu sehen? Wie groß ist die gesamte Installation? Wie wird diese Installation gesteuert?

Engagement Session	1
Artefakt	1 – Interaktives Nagelbrett 2 – Interaktives Displaybrett
Artefakt Typ	Handskizzen, zusätzliche Erklärung
Datum	13.01.2018
Anzahl der externen Stakeholder	2
Annahmen	• Grundsätzliche Form • Aufbau • Art der Interaktion • Verhalten der Installation
Reaktionen	• Zuspruch mit Displays • Unzureichende Informationen über Aussehen der Installation • Fragen über Verhalten, Inhalt, Steuerung

Tabelle 1 – Engagement Session 1

Mit diesen gewonnen Informationen aus der ersten „Engagement Session“ wurde das nächste „Artefakt“ erschaffen. Dieses und die Reaktionen der nächsten „Engagement Sessions“ werden im folgenden Abschnitt 4.2 behandelt.

4.2 Morph-Spiegel

Die im vorherigen Abschnitt behandelten Reaktionen bilden eine Grundlage für dieses „Artefakt“. Daher konzentriert sich dieses auf die Verwendung von Displays und konkretisiert die Form und das Aussehen der Installation. Um das Gefühl der Interaktion und den Live-Charakter zu gewährleisten wurde dieses „Artefakt“ als 3D-Rendering in vvvv[5] umgesetzt. Abbildung 23 zeigt einen Ausschnitt davon.

[5] https://vvvv.org/

Abbildung 23 – Artefakt 3: 3D-Rendering „Morph-Spiegel“

Dieses „Artefakt“ ist als Simulation auf einem Laptop mit Webcam erlebbar. Diese visualisierte Installation besteht aus 25 einzelnen Displays welche Hochformat 5 waagrecht mal 5 senkrecht angeordnet sind. Jedes davon kann wie bei den vorangegangenen „Artefakten“ in einer Achse vor und zurückgeschoben werden. Die gesamte Installation kann dadurch verschiedene Formen annehmen. Beim Annähern des Benutzers an die Installation wird er von der Webcam erfasst und sein Spiegelbild erstreckt sich über alle Displays der Installation. Auch die Positionen der Displays nähern sich seiner Kopfform an. Bei der Visualisierung erfolgt die Positionierung der Displays manuell. Bei der fertigen Installation würde sie aber automatisch gesteuert werden. Bei diesem „Artefakt“ handelt es sich dadurch um einen digitalen Spiegel, der neben dem erfassten Bild auch die erfasste Form wiedergibt. Der Benutzer tritt in Interaktion mit der Installation. Sie spiegelt seine Bewegungen in Licht und Form wider.

Dieses „Artefakt 3“ nimmt starken Bezug zu das im Abschnitt 4.1 beschriebene „Interaktiven Nagelbrett“. Die Beziehung zwischen der Form des Benutzers und der Form der Installation spielen hier eine wesentliche Rolle. Durch die Erschaffung dieser Visualisierung wurde folgende Entdeckung gemacht. Wie bereits in Abschnitt 4.1 besprochen, ergibt sich durch die Verwendung von Displays anstatt von Nägeln eine geringere Auflösung der einzelnen Positionen.

Das heißt die Formänderung der Installation von „Artefakt 3“ unterstützt die Spiegelung des Benutzers, aber ohne Inhalt der Displays wäre eine Wiedererkennung der ursprünglichen Form nicht möglich. Abbildung 24 zeigt das Beispiel aus Abbildung 23 mit grauer Fläche anstatt des Spiegel-Bilds. Auf die Form des Kopfes könnte hier nicht geschlossen werden.

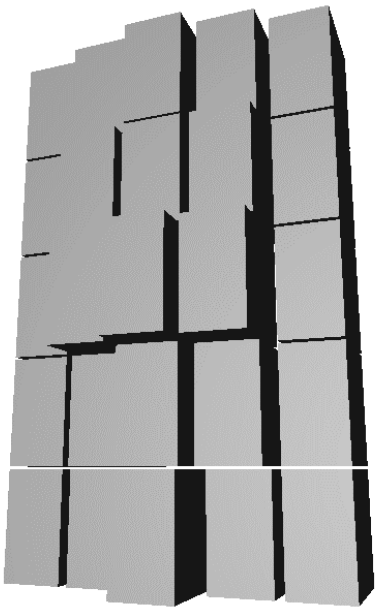


Abbildung 24 – Artefakt 3: Displays zeigen graue Fläche

Dieser „Morph-Spiegel“ hat große Ähnlichkeit zu „MegaFaces“ in Abschnitt 3.3. Auch dort geht es um die Wiedergabe von Gesichtern in dreidimensionaler Form. Um eine höhere Auflösung zu erreichen verwenden die Entwickler von „MegaFaces“ statt der hier besprochenen Displays RGB-LEDs. Insgesamt verwenden sie 11.000 Aktuatoren und ebenso viele RGB-LEDs (iart, 2014). Dadurch können die einzelnen Pixel näher aneinander positioniert werden und eine höhere Auflösung erreicht werden. Der hier besprochene „Morph-Spiegel“ hat im Gegensatz zu den 11.000 verschiedenen Punkten nur 20 Punkte, die im Raum positioniert werden können. Für die Wiedergabe von 3D-Gesichtern ist diese Installation dadurch weniger geeignet.

Auch in der zweiten „Engagement Session“ wurde das „Artefakt“ zwei externen Stakeholdern einzeln vorgelegt. Diesmal wurde des „Artefakt 3: Morph-Spiegel“ gezeigt. Ziel war es mit der Visualisierung der Installation zu interagieren. Daraus entwickelten sich einige gute und wichtige Reaktionen der externen Stakeholder. Tabelle 2 zeigt wieder die wichtigsten Fakten daraus.

Auch hier wurde das zuvor besprochene Problem der geringen Auflösung der Positionspunkte angesprochen. Deshalb kam der Vorschlag die Spiegelung der Form zwischen Benutzer und Installation wegzulassen, da dies wenig Mehrwert hat. Ein weiterer Vorschlag war die Installation im Querformat anzuordnen, anstatt Hochformat. Die Entscheidung für Hochformat beim „Artefakt 3“ hing vor allem mit der Beziehung zum Kopf des Benutzers zusammen. Außerdem wurde die Möglichkeit vorgebracht den Inhalt der Displays abstrakter zu gestalten. Beispielsweise könnte die Position der Displays mit der Zeit verknüpft werden. Verschiedene Zeitpunkte könnten so mit unterschiedlichen Positionen der Displays zusammenhängen. Dadurch würde sich ein riesiges Feld zum Experimentieren auftun. Der letzte wichtige Punkt der Reaktionen bezog sich auf den Aufbau und die Funktionsweise der Installation. Der zweite externe Stakeholder ging davon aus, dass die Displays mittels Servomotoren auch neigbar und schwenkbar wären. Diese Möglichkeit wurde bislang noch nicht bedacht. Sie würde aber die gesamte Installation noch flexibler machen und den Platz für mehrere verschiedene Verwendungen eröffnen.

Engagement Session	2
Artefakt	3 – Morph-Spiegel
Artefakt Typ	Interaktives 3D-Rendering
Datum	28.01.2018
Anzahl der externen Stakeholder	2
Annahmen	• Starke Formbeziehung zu Kopf • Spiegel der sich der Form des Benutzers anpasst • Hochformat • Inhalt über gesamte Installation gemappt
Reaktionen	• Bezugnahme der 3 Dimension auf Zeit • Querformat • Abstrakter Inhalt • Keine Körperbeziehung zum Benutzer • Bewegliche Displays, Neigbar

Tabelle 2 – Engagement Session 2

Einige dieser wertvollen Reaktionen und Vorschläge wurden im nächsten „Artefakt“ berücksichtigt. Der folgende Abschnitt geht näher darauf ein.

4.3 Raum-Zeit Installation

Dieses „Artefakt 4“ wurde ebenso wie das vorherige im Abschnitt 4.2 als 3D-Rendering in vvvv[6] umgesetzt. Die Änderungen aufgrund der Reaktionen aus der vorangegangenen „Engagement Session“ in Tabelle 2 wurden darin festgehalten. Diese Simulation ist im Gegensatz zur vorherigen nicht mehr interaktiv. Abbildung 25 zeigt vier verschiedene Ausschnitte von links nach rechts als Sequenz. Dabei handelt es sich um eine Installation bestehend aus 25 einzelnen Displayelementen, welche 5 waagrecht mal 5 senkrecht als Displaywand angeordnet sind. Jedes von denen kann wie beim „Artefakt 3“ in z-Richtung verschoben werden. Allerdings können hier die einzelnen Elemente zusätzlich geneigt und geschwenkt werden, was eine komplexere Verformung ermöglicht. Im 3D-Rendering dieses „Artefakts“ ist eine sphärische Verkrümmung von den Extrempositionen in Abbildung 25 links bis rechts und wieder zurück zu sehen. Diese Installation ist im Querformat angeordnet. Der Inhalt, der auf den Displays gezeigt wird, erstreckt sich über alle 25 Elemente. Darin sind schnelle sich verformende, abstrakte Farben in hellem und dunklem Blau zusehen. Diese erinnern an Aufnahmen in Filmen, in denen Zeitsprünge beziehungsweise Reisen durch die Zeitdimension dargestellt werden. Die sphärische Verformung der Installation ist eher langsam, wobei der Inhalt ein schnelleres Tempo mit sich bringt. Sie bilden zueinander einen großen Kontrast.

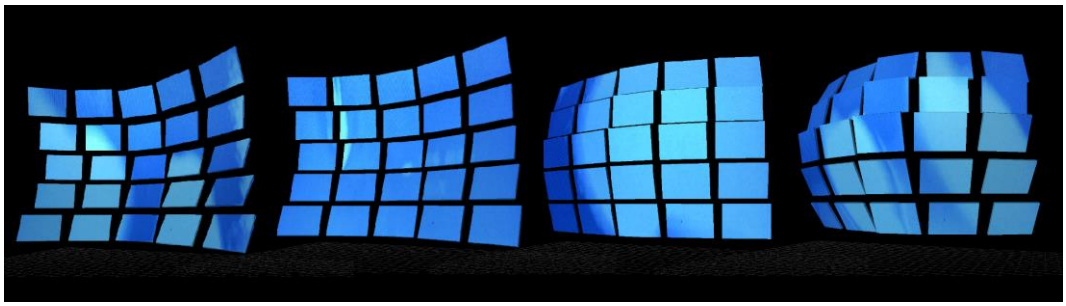


Abbildung 25 – Artefakt 4: 3D-Rendering „Raum-Zeit Installation“

Das „Artefakt 4 – Raum-Zeit Installation“ wurde in der dritten „Engagement Session“ sechs externen Stakeholdern vorgelegt. Es wurde das 3D-Rendering in einer Schleife abgespielt. Ziel der Stakeholder war es, sich diese Installation als Umsetzung im physikalischen Raum vorzustellen. Dabei entstanden einige Fragen und Assoziationen. Eine der wichtigsten Fragen war die Größe der Installation

[6] https://vvvv.org/

betreffend. Diese Simulation wurde auf einem 17-Zoll Laptopdisplay wiedergegeben. Deshalb bestand hier kein Vergleich zur finalen Größe der Installation. Einerseits wurde gefragt, wie groß die einzelnen Displays sind und andererseits welche Dimensionen die gesamte Displaywand annimmt.

Eine wirklich interessante Assoziation kam in dieser „Engagement Session“ auf. Dieses 3D-Rendering erinnert durch die einzelnen Elemente und die Form an ein Facettenauge wie es im Tierreich vorkommt. Die Facettenaugen von Gliederfüßern bestehen aus mehreren Dutzend bis mehreren 10.000 Einzelaugen (Nachtigall & Wisser, 2013, S. 164). Jedes dieser Einzelaugen nimmt einen winzigen Ausschnitt des Gesichtsfeldes wahr. Das Lebewesen kann sich dadurch ein Mosaikbild variierender Helligkeit bilden (Hickman, 2008, S. 1101). Im Gegensatz zu Linsenaugen, die nur das mittlere Sehfeld scharf sehen, wird beim Facettenauge das gesamte Sehfeld scharf abgebildet (Heldmaier & Neuweiler, 2013). Facettenaugen eignen sich daher sehr gut zur Wahrnehmung von Bewegungen (Hickman, 2008, S. 1101). Abbildung 26 zeigt die Facettenaugen einer Pferdebremse.

Abbildung 26 – Facettenaugen einer Pferdebremse („Facettenauge“, 2018)

Wie auch in vielen anderen Bereichen werden hier Naturphänomene in die Technik übertragen werden. Hierbei dient das Facettenauge als Inspirationsquelle zur Entwicklung verschiedenster Sensoren. Die Firma Lytro[7] entwickelte beispielsweise Lichtfeldkameras, die dieses Phänomen mit mehreren Sensorelementen nutzt. Zusätzlich zum zweidimensionalen Bild wird hier die

[7] https://support.lytro.com/hc/de

Information über den Lichteinfallswinkel gespeichert. Dadurch kann auf die Tiefe des Bildes geschlossen werden. Im Nachhinein kann dadurch die Tiefenschärfe eines aufgenommenen Bildes verändert werden oder ein Bild mit theoretisch unendlicher Tiefenschärfe aufgenommen werden (Lytro, 2018).

Das Thema und die Funktionsweise des Facettenauges wurden im folgenden Projekt auch künstlerisch aufgegriffen. Der Künstler Andreas Rimkus gestaltete zwei Facettenaugen in der die Besucher in die Rolle eines Insekts eintauchen können. Die Metallskulptur weist insgesamt 120 Facetten in Form von Linsen auf, durch die der Besucher die Umgebung in vereinfachter Weise wie ein Insekt wahrnehmen kann. Abbildung 27 zeigt diese Skulptur von der Innenseite. Sie steht im Park der Sinne in Laatzen in der Nähe von Hannover (Park der Sinne, 2018).

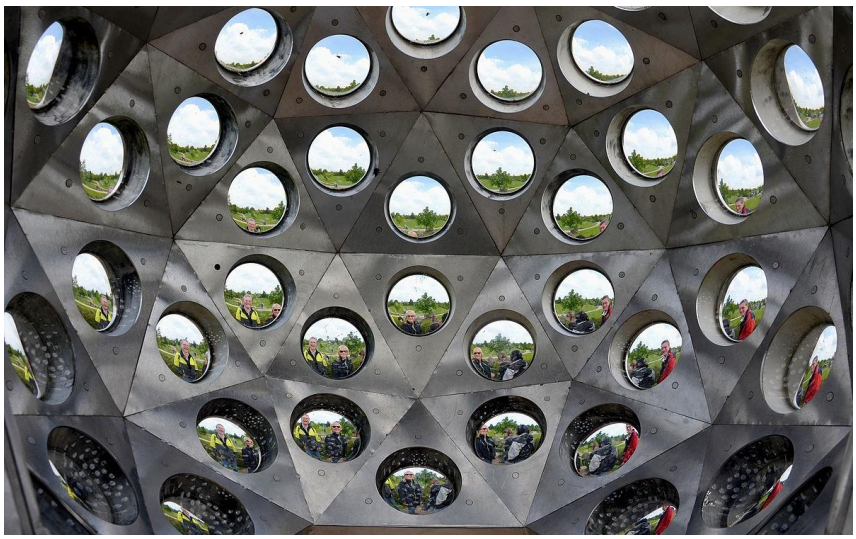


Abbildung 27 – Insektenauge, Andreas Rimkus (Hannover, 2014)

Engagement Session	3
Artefakt	4 – Raum-Zeit Installation
Artefakt Typ	3D-Rendering
Datum	07.02.2018
Anzahl der externen Stakeholder	6
Annahmen	• Querformat • Verformbare Fläche, linear, neigen, schwenken • z-Position hängt mit Zeit zusammen • Inhalt über gesamte Installation gemappt

Reaktionen	• Fragen zur Größe • Beziehung mit Facettenauge, Lichtfeldkamera

Tabelle 3 – Engagement Session 3

In der „Creation Session“ des vierten „Artefakts – Raum-Zeit Installation“ entstand eine Idee zur technischen Umsetzung dieser Installation. Diese wird im folgenden Abschnitt behandelt.

4.4 Modulskizzierung

Parallel zum „Artefakt 4“ in Abschnitt 4.3 entstand dieses „Artefakt“. Während der „Creation Session“ in der der Aufbau und die Bewegungen der „Raum-Zeit Installation“ durchdacht und entwickelt wurden, entstand nebenbei auf einem Blatt Papier eine Handskizze, die die technische Funktionsweise für die erforderlichen Bewegungen darstellt. Abbildung 28 zeigt einen Scan davon.

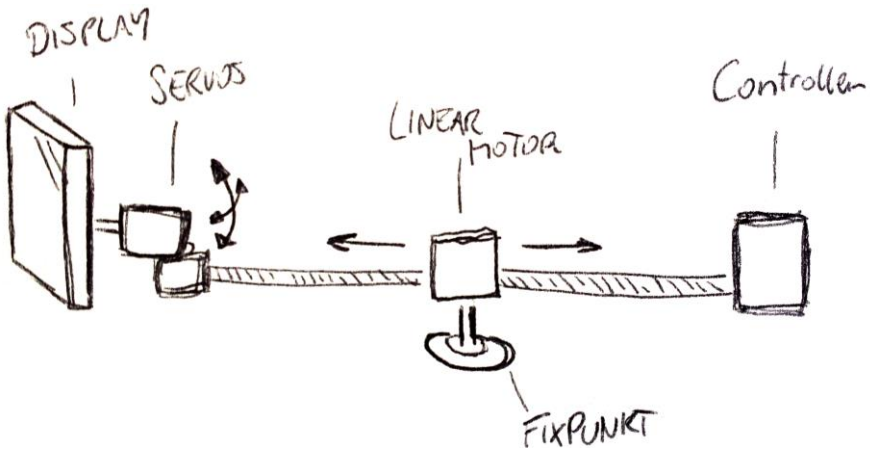


Abbildung 28 – Artefakt 5: Handskizze „Modulskizzierung“

Um die gewünschten Bewegungen, wie in Abbildung 25 dargestellt, zu ermöglichen, wurde obig skizzierter technischer Aufbau angedacht. Dieses Modul stellt ein Element aus den 25 Elementen der Installation dar. Jedes soll eine Bewegung in Bildrichtung hinein und heraus erlauben. Dafür ist der Linearmotor zuständig. Um das Display neigen und schwenken zu können, ist es auf einem Pan-Tilt-Servo montiert. Diese Servos sollen eben eine Rotation im Raum ermöglichen. Um alles regeln zu können, ist auf der Rückseite ein Controller angebracht. Dieses Modul kann am Linearmotor an einem Rahmen oder ähnlicher Konstruktion fixiert werden. Somit ist es möglich, dass es sich frei in einer Achse bewegen kann.

Die Grundüberlegung dabei ist es, dass jedes Modul bestehend aus Display, Servos, Linearmotor und Controller sich eigenständig im Raum bewegen kann und ein eigenes Bild wiedergeben kann. Die Installation kann dann aus beliebig vielen Modulen zusammengesetzt werden. Von einem Punkt werden Bild- und Steuerdaten an die einzelnen Module gesendet, die die Installation betreiben und zum Leben erwecken.

Dieses „Artefakt 5 – Modulskizzierung“ wurde einem externen Stakeholder vorgezeigt, der sehr in elektronische und technische Produktentwicklung vertieft ist. Ziel in dieser „Engagement Session“ war es die technische Umsetzbarkeit zu überdenken und aus unterschiedlichen Blickwinkeln zu betrachten.

Die Funktionsweise war aus der Modulskizzierung aus Abbildung 28 für den externen Stakeholder schnell ersichtlich. Erste Reaktionen äußerten sich bezüglich Verkabelung und Positionierung der einzelnen Bauteile. Je nach Bauform und Funktionsweise der Servos, Linearmotoren, Displays und Controllern ist auf sinnvolle Positionierung und Verbindung dieser zueinander zu achten. Die Wege der Kabel sowie Knicke durch Bewegungen sind ebenfalls wichtige Punkte, die nicht außer Acht gelassen werden sollen.

Eine interessante Idee entstand ebenfalls in dieser „Engagement Session“, die sich als Skizze manifestiert hat. Beim „Artefakt 5“ ist davon ausgegangen worden, dass es einen zentralen Punkt gibt, von dem aus die Bild- und Steuerdaten für die Installation gesendet werden. Die Idee hierbei ist es, dass an jedem Modul eine eigene Kamera befestigt ist, welche das Bild für das Display liefert. Wie aus der entstandenen Skizze in Abbildung 29 links zu sehen ist, ist die Kamera auf der Seite gegenüber dem Display ebenfalls am Linearmotor angebracht. Sowohl das Display als auch die Kamera sind auf Pan-Tilt-Servos montiert um eine Rotation im Raum zu ermöglichen. Somit wäre es jedem Modul möglich verschiedene Positionen und Rotationen im Raum einzunehmen, welche verschiedenste Effekte erzielen.

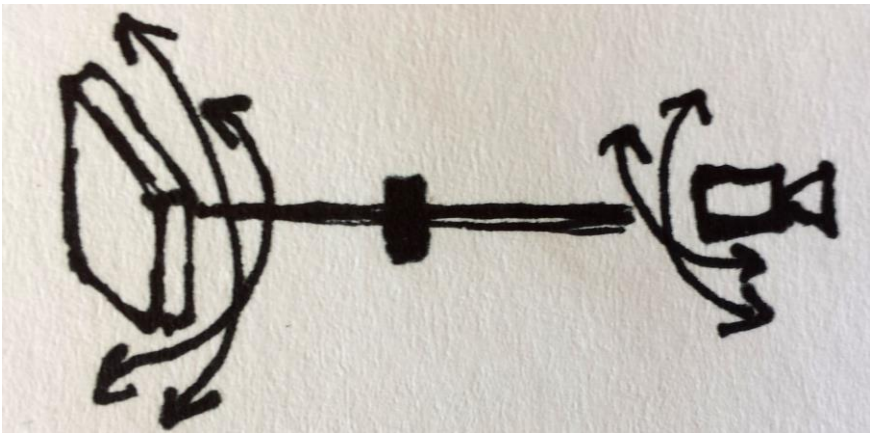


Abbildung 29 – Idee: Modul mit Kamera und Display

Dies bedeutet, dass bei dieser Idee die Bilddaten nicht von einem zentralen Punkt aus an die Module gesendet werden, sondern dass jedes Modul seine eigenen Bilddaten erzeugt. Die Steuerdaten für den Linearmotor sowie für die Servos könnten von einem zentralen Punkt aus oder auch direkt von den Controllern am Modul kommen.

Werden diese Module mit Kamera und Display nebeneinander angeordnet und als Installation betrieben, hätte diese große Ähnlichkeit zu dem im Abschnitt 4.3 besprochenem Facettenauge. Jedes Modul würde einem Einzelauge entsprechen. Es nimmt einen bestimmten Ausschnitt des gesamten Gesichtsfeldes wahr und bildet daraus ein gemeinsames Mosaikbild, das aus Teilen unterschiedlicher Perspektiven zusammengesetzt ist. Es hätte vermutlich eine ähnliche Wirkung wie das Insektenauge von Andreas Rimkus in Abbildung 27.

Engagement Session	4
Artefakt	5 – Modulskizzierung
Artefakt Typ	Handskizze
Datum	16.02.2018
Anzahl der externen Stakeholder	1
Annahmen	• Verformbare Fläche, linear, neigen, schwenken • Zentrale Steuereinheit, Controller je Modul • Inhalt über gesamte Installation gemappt

Reaktionen	• Wichtigkeit Hardware, Positionierung, Verbindung zueinander • Idee: Jedes Display wird von einer Kamera versorgt

Tabelle 4 – Engagement Session 4

Aus dieser „Engagement Session“ ist ein neuer Weg entstanden, der weitere Möglichkeit auftut. Die definitive Entscheidung über den Inhalt, der auf den Displays gezeigt wird, soll getroffen werden, wenn ein Teil der Installation physisch existiert, da hier besser die Wirkung ermittelt werden kann. Das „Artefakt“ im nächsten Abschnitt geht näher auf die Größe und Form der Installation ein.

4.5 Die Papertypen

Eine der Reaktionen aus der dritten „Engagement Session“ in Tabelle 3 war eine Frage bezüglich der Größe der Installation. Die „Artefakte 6, 7 und 8 – Die Papertypen“ in diesem Abschnitt beschäftigen sich näher mit dieser Kernfrage. Einerseits sollen die einzelnen Displays leicht und klein genug sein um schnell im Raum bewegt zu werden, andererseits sollen sie groß genug sein um einen immersiven Eindruck für den Benutzer zu gewährleisten. Für diese Fragestellung wurden drei unterschiedlich große Videowände mit Hilfe von Papierzetteln nachgebildet. Jede dieser drei Nachbildungen besteht aus 25 einzelnen Displays, welche 5 waagrecht mal 5 senkrecht angeordnet sind.

Der erste „Papertyp 3.5“ in Abbildung 30 besteht aus Elementen mit einer Diagonale von 3,5 Zoll. Das entspricht einer Diagonalen von knapp 9cm. Im Falle einer 5x5 Anordnung wie in Abbildung 30 zu sehen, ergibt sich für die gesamte Installation eine Diagonale von etwa 50cm, je nach Positionierung der Displays zueinander.

Abbildung 30 – Artefakt 6: Papierzetteln „Papertyp 3.5“

Der zweite „Papertyp 5“ ist aus Zetteln mit einer Diagonale von 5 Zoll beziehungsweise 12,7cm gebaut. Daraus ergibt sich für die gesamte Videowand in einer 5x5 Anordnung wie in Abbildung 31 eine Diagonale von zirka 72cm. Falls die Installation nur aus 20 Displayelementen geformt wird, dann ergibt sich bei einer Anordnung mit 5 waagrechten Mal 4 senkrechten Elementen eine Diagonale von etwa 66cm.

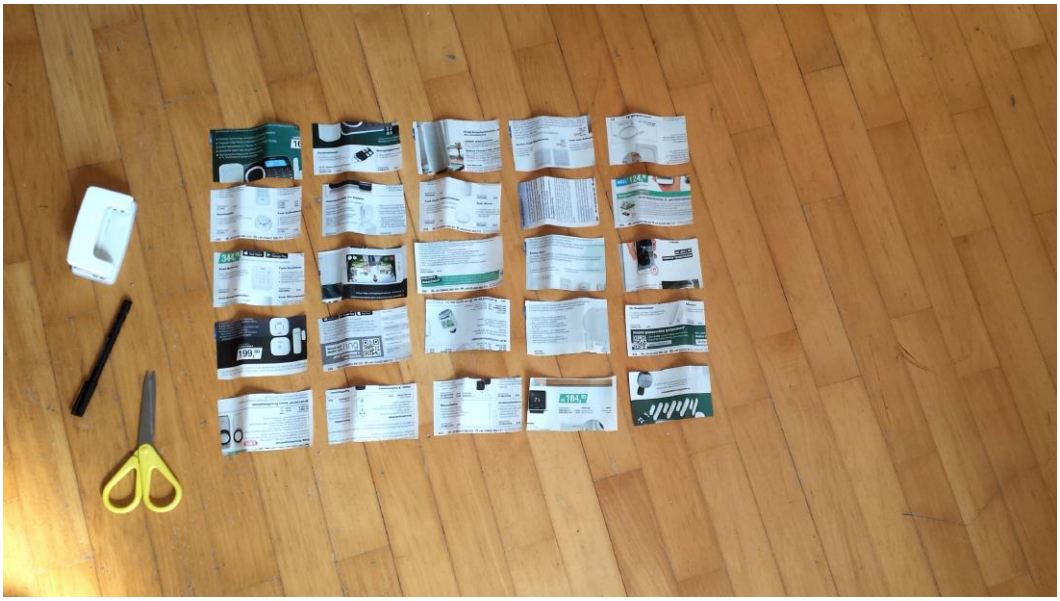


Abbildung 31 – Artefakt 7: Papierzetteln „Papertyp 5“

Der „Papertyp 7“ in Abbildung 32 ist aus Papierzetteln in der Größe von 7 Zoll Displays zusammengesetzt. 7 Zoll entsprechen 17,78cm. Dadurch ergibt sich für die gesamte Installation bei einer Anordnung von 5x5 Displays eine Diagonale von etwa 90cm. Im Falle einer Anordnung von 5 waagrechten Mal 4 senkrechten Displays ergibt sich eine Diagonale von etwa 85cm.

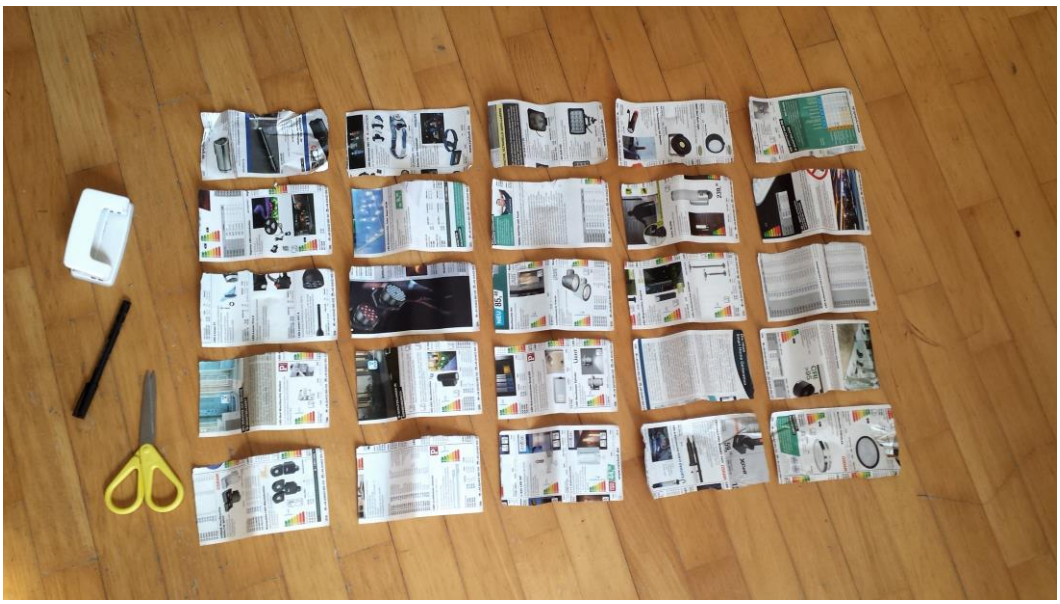


Abbildung 32 – Artefakt 8: Papierzetteln „Papertyp 7“

In der fünften „Engagement Session“ nahmen vier externe Stakeholder teil. In dieser Session wurden die drei „Papertypen“ unterschiedlicher Größe

nebeneinander aufgelegt. Das Ziel für die externen Stakeholder war es die Wirkung der unterschiedlichen „Artefakte“ aufgrund der Größe zu beurteilen. Dadurch entstanden einige wichtige Reaktionen.

Einerseits fiel die Entscheidung schnell auf die Displays mit einer Diagonalen von 7 Zoll. Die externen Stakeholder hielten eine ausreichende Größe für ein immersives Erlebnis notwendig. Mit einer Diagonalen von 90cm bei einer 5x5 Anordnung beziehungsweise 85cm bei einer 5x4 Anordnung wäre die Installation groß genug um den Großteil des Sichtfeldes des Benutzers bei näherer Betrachtung abzudecken. Bei einer Verwendung der 3,5 Zoll Displays wie im „Artefakt 6“ hätte die Installation eine Diagonale von nur 50cm. Selbst bei näherer Betrachtung wäre nicht das gesamte Sichtfeld des Benutzers ausgefüllt. Deshalb viel die Entscheidung auf die 7 Zoll Displays.

Andererseits kamen Reaktionen bezüglich der Anordnung und der Form der Installation. Da die einzelnen Papierzettel lose auf dem Tisch lagen, konnten sie leicht herumgeschoben werden und die Installation konnte schnell unterschiedliche Formen annehmen. Das Fazit daraus war, dass es unbegrenzte Möglichkeiten zur Anordnung gibt und dass die Anordnung als klassische Videowand nicht unbedingt fixiert werden muss. Zusätzlich kam noch ein weiterer Input bezüglich des Werkzeugs zum Erstellen von „Artefakten“. Durch das Aufkommen von Virtual Reality Brillen mit Controllern wie beispielsweise der Oculus Rift[8] ist es mit Hilfe von unterschiedlichen Apps möglich Skizzen im dreidimensionalen Raum anzufertigen. Dadurch ist es möglich die Vorstellungen noch detaillierter festzuhalten, um diese „Artefakte“ und Ideen mit anderen teilen zu können. Abschnitt 4.10 beschäftigt sich näher damit.

Engagement Session	5
Artefakt	6 – Papertyp 3.5 7 – Papertyp 5 8 – Papertyp 7
Artefakt Typ	Nachbildungen aus Papier
Datum	01.03.2018
Anzahl der externen Stakeholder	4
Annahmen	• Displays unterschiedlichster Größe

[8] https://www.oculus.com/rift/

	• Anordnung als Displaywand
Reaktionen	• Größe: 7-Zoll Displays • Andere Anordnung als Displaywand • Ideenfindung der Anordnung im virtuellen Raum

Tabelle 5 – Engagement Session 5

Die „Engagement Session“ in diesem Abschnitt hat sich detaillierter mit den Fragen bezüglich Größe und Anordnung der Installation auseinandergesetzt. Diese Fragen werden mit den gewonnenen Reaktionen in Abschnitt 4.10 weiter behandelt. Der nächste Abschnitt beschäftigt sich näher mit den Möglichkeiten der Signalverarbeitung für die Bild- und Steuerdaten.

4.6 Signalverarbeitung der Installation

Für die Umsetzung dieser Installation ergibt sich folgende Problemstellung. Die Installation besteht aus etwa 20 bis 25 beweglichen Displays. Jedes davon soll individuelle Bild- und Steuerdaten bekommen, um daraus eine Videowall werden zu lassen, damit ein einziges Bild über die gesamte Installation wahrgenommen werden kann. (Bundulis & Arnicans, 2013) listen in ihrem Paper verschiedene Ansätze zur Ansteuerung solcher Videowalls auf und besprechen die Vor- und Nachteile dieser Möglichkeiten.

Eine Möglichkeit zur Umsetzung von Videowalls ist die Verwendung von Video Splittern wie beispielsweise den TripleHead2Go[9] von Matrox. Dabei kann ein Laptop an diesen Splitter angehängt werden. Das Betriebssystem des Laptops erkennt ein großes Display. Der Splitter gibt dann daraus mehrere Videosignale mit verschiedenen Inhalten aus. Dadurch können Videowalls mit einer 2x2 beziehungsweise 3x3 Matrix einfach ohne zusätzlicher Verwendung von anderer Hardware betrieben werden (Bundulis & Arnicans, 2013). Ein solcher Splitter für den Betrieb von drei Bildschirmen bewegt sich im Jahr 2018 in einem Preisbereich von 200 – 300 €. Andere Splitter wie beispielsweise der MST-Hub von Club-3D[10] verwenden Multi Stream Transport. Dadurch sind für das Betriebssystem des Laptops mehrere Bildschirme sichtbar, obwohl nur ein Videokabel angeschlossen

[9] http://www.matrox.com/graphics/de/products/gxm/th2go/displayport/

[10] http://www.club-3d.com/de/detail/2411/multi_stream_transport_(mst)_hub_displayportt_1.2_quad_monitor/

ist. Der Vorteil dabei ist, dass am Laptop verschiedene Einstellungen wie Auflösung und Bildwiederholungsrate für jeden Bildschirm einzeln eingestellt werden können (Club-3D, 2018).

Eine weitere Möglichkeit für den Betrieb von Videowalls ist die Verwendung von mehreren Grafikkarten. Abhängig von der Konfiguration von Motherboard und den Grafikkarten können etwa acht verschiedene Videoausgänge gesteuert werden. Diese Lösung ist aber weniger mobil als erstere, da eine solche Konfiguration nur mit Standrechnern möglich ist (Bundulis & Arnicans, 2013).

Die ersten beiden Möglichkeiten sind an die Hardware gebunden. Je nach Größe der Videowall steigt die Notwendigkeit für Adapter, Splitter, Kabel, Grafikkarten und so weiter. Die dritte angeführte Möglichkeit virtualisiert die Verbindungen von der Steuereinheit zu den Displays. Dabei wird die Videowall von einem Master und mehreren Clients betrieben. Vom Master können über Netzwerk entweder Videodaten oder logische Repräsentationen über das Bild, wie beispielsweise 3D-Mesh Daten oder OpenGL Befehle, an die Clients gesendet werden, die sich daraus Teile herausnehmen oder eigene Bilder aufgrund der empfangenen Daten rendern. Dadurch kann die Anzahl der Videowall-Displays einfach durch die Verwendung von weiteren Clients hochskaliert werden. Diese müssen lediglich in das gleiche Netzwerk eingebunden werden (Rudolfs Bundulis & Arnicans, 2014).

Da die Videowall in dieser Arbeit aus etwa 20 – 25 Displays bestehen soll, sind die beiden ersten angeführten Möglichkeiten mit Splittern und Grafikkarten nicht sinnvoll. Da für 20 Displays einige solcher Videosplitter notwendig wären, würde dies kostenineffizient werden. Deshalb und aufgrund der leichteren Skalierbarkeit wird in dieser Arbeit auf die dritte Möglichkeit gesetzt. Abbildung 33 zeigt das Blockschaltbild.

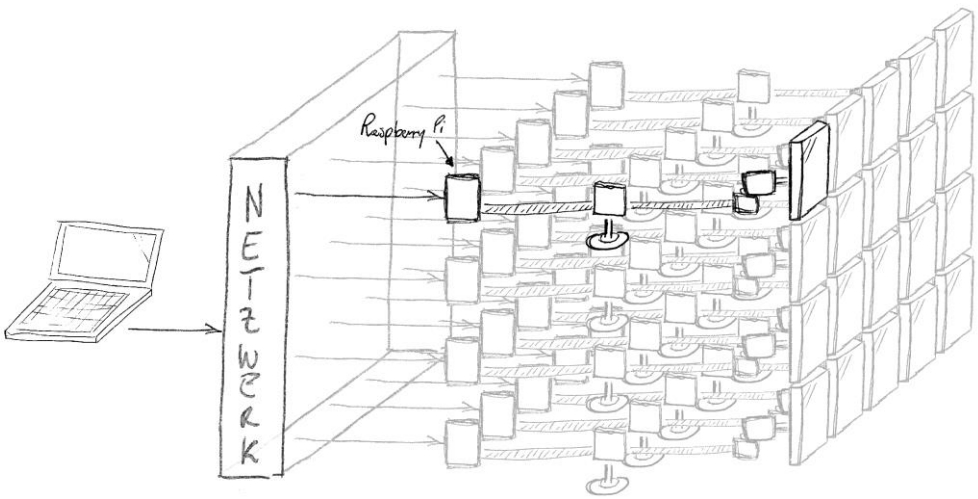


Abbildung 33 – Blockschaltbild Installation

Als Master wird in dieser Arbeit ein PC oder ein Raspberry Pi angedacht, welcher Bild- und/oder Steuerdaten an die Clients der Installation sendet. Diese Clients werden mit Hilfe von Raspberry Pis umgesetzt. Am Master werden Daten für die gesamte Installation errechnet. Diese Daten können einerseits die Positionen und Rotationen der einzelnen Displays repräsentieren und andererseits den Content der Displays wiederspiegeln. Diese Daten gelangen übers Netzwerk an die Raspberry Pis, welche diese empfangen und für jedes Modul weiterverarbeiten. Die erzeugten Videobilder gelangen über HDMI an die Displays. Die aufbereiteten Rotations- und Positionsdaten werden über die GPIO Pins des Raspberrys an die Servos und den Linearmotor weitergegeben. Hiermit kann von einem zentralen Punkt aus, die gesamte Installation gesteuert werden.

Die in diesem Abschnitt behandelte Überlegung bietet eine gute Grundlage für weitere Entwicklungen der einzelnen Module und der gesamten Installation. Die Module werden alle ident aufgebaut. Sie unterscheiden sich lediglich durch ihre ID und der Position in der Installation. Dadurch ist es ohne viel Verkabelungsaufwand durch Adapter oder Converter möglich diese Installation einfach zu skalieren. Der folgende Abschnitt beschäftigt sich näher mit den Komponenten und der Signalverarbeitung des Moduls.

4.7 Signalverarbeitung des Moduls

Jedes Modul setzt sich aus jetziger Sicht, wie in Abbildung 28 ersichtlich, aus einem Display, welches auf Servos montiert ist, einem Linear-Aktuator und einem Raspberry Pi zusammen. Folgende Abschnitte stellen die verwendeten Komponenten des Moduls vor und anschließend wird näher auf die Signalverarbeitung dieser eingegangen.

4.7.1 Display

In der fünften „Engagement Session“ in Tabelle 5 wurde die Größe der Displays auf sieben Zoll festgelegt. Für den Bau dieser Installation werden 7-Zoll LCDs mit einer Auflösung von 1024x600 Pixeln verwendet. Die Kosten waren Hauptursache für diese Auswahl. Wie in Abbildung 34 ersichtlich besteht die Einheit aus dem Display, einer Treiberplatine und einer Eingabeplatine. Das Display misst 165 x 100 x 5 mm, wobei die Anzeigefläche 154.08 x 85.92 mm groß ist. Es weist eine Leuchtdichte von 220cd/m² auf und hat einen Betrachtungswinkel von 70° links und rechts, 50° oben und 60° unten. Es wird mit einer 50 Pin Digital (TTL) TCON Schnittstelle durch die Treiberplatine mit Strom und Daten versorgt. Insgesamt hat diese Einheit eine Leistung von etwa 6 bis 7 Watt und eine Betriebsspannung von

5V. Dadurch ist es möglich das Display über den USB-Port eines Raspberrys mit Strom zu versorgen. Für die Videoansteuerung stehen die Schnittstellen HDMI 1.2, VGA und Composite Video zur Verfügung. Mittels Eingabeplatine können Einstellungen wie Helligkeit, Farbe, Quelle und Bildtransformationen getroffen werden. Die Verbindung von Eingabeplatine und Treiberplatine ist für den Betrieb nicht notwendig (52Pi, 2018).

Abbildung 34 – 7-Zoll LCD inkl. Treiberplatine und Eingabeplatine (52Pi, 2018)

4.7.2 Servos

Um die Neigung und Drehung der Displays zu ermöglichen, werden Pan-Tilt-Servos vom Typ AS3103PG verwendet. Dieses Set besteht, wie in Abbildung 35 ersichtlich, aus zwei verschiedenen Servos die mittels zweier Metallbügel verbunden werden können. Die Ansteuerung der Servos erfolgt über Pulsweitenmodulation (PWM). Abschnitt 7.2.1 geht näher auf dieses Thema ein. Die Servos messen 40 x 20 x 37,5 mm. Bei 4,8V ist ein Stell-Moment von 34 Ncm und eine Stell-Zeit von 0,19 sec (60°) angegeben, bei 6V ein Stell-Moment von 41 Ncm und eine Stell-Zeit von 0,22 sec (60°) (ali, 2018).

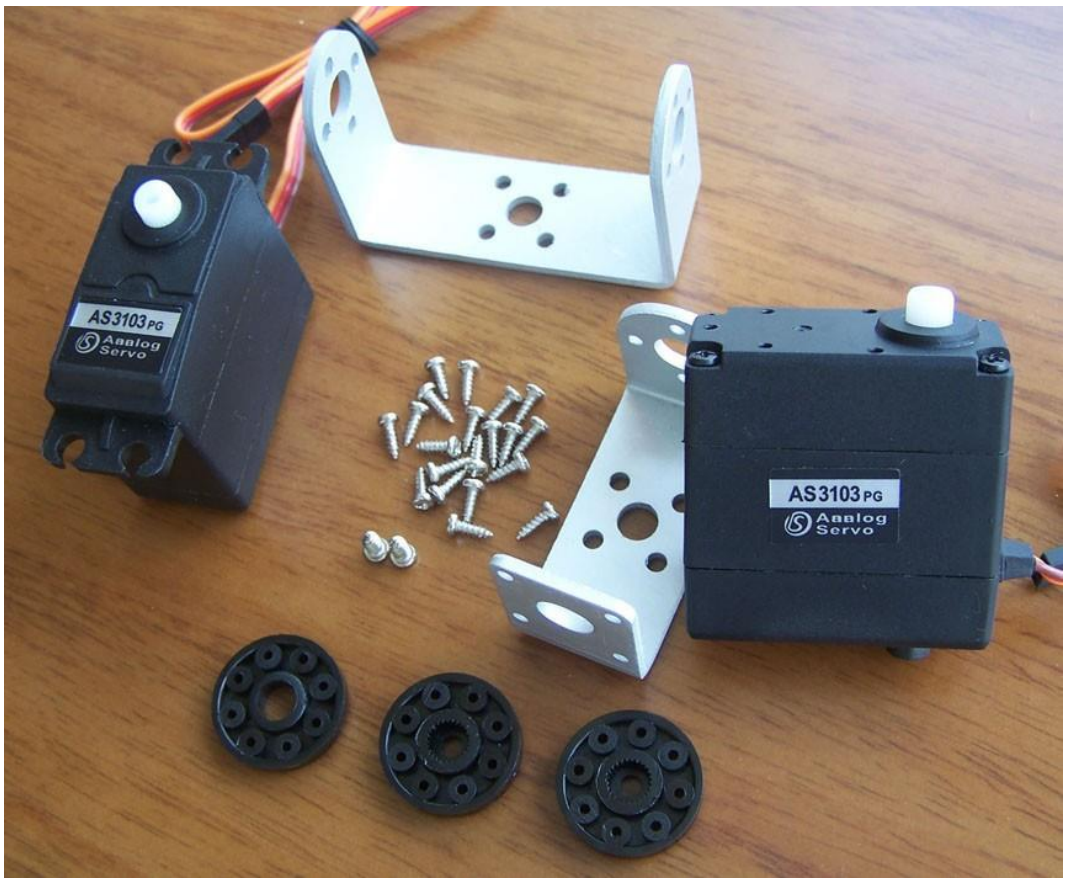


Abbildung 35 – Pan-Tilt-Servos AS3103PG (ali, 2018)

4.7.3 Linear-Aktuator

Für die Gewährleistung der Linearbewegung der Displays wird ein Linear-Aktuator, der mittels Stepper-Motor gesteuert wird, verwendet. Wie in Abbildung 36 ersichtlich, ist der Linear-Aktuator aus schwarzem ABS (Acrylnitril-Butadien-Styrol) 3D gedruckt. Er besteht aus der Basis und einer Zahnstange. An der Basis befindet sich der Stepper-Motor des Typs 28BYJ-48. An seiner Welle ist ein Zahnrad

angebracht, das durch Drehung die Zahnstange vor und zurück schieben kann. Daraus ergibt sich ein Aussteuerbereich von 248mm. An der Basis ist ein Taster angebracht, der zur Referenzierung der Position verwendet werden kann. Der Stepper-Motor kann unipolar oder bipolar betrieben werden. Damit dieser Motor mittels Raspberry Pi gesteuert werden kann, ist eine ULN2003a Treiberplatine, wie links unten in Abbildung 36 zu sehen, notwendig. Daraus ergibt sich eine Linear-Geschwindigkeit von 7mm pro Sekunde und eine Aktuator-Kraft von 6,8N (Otamat, 2017).

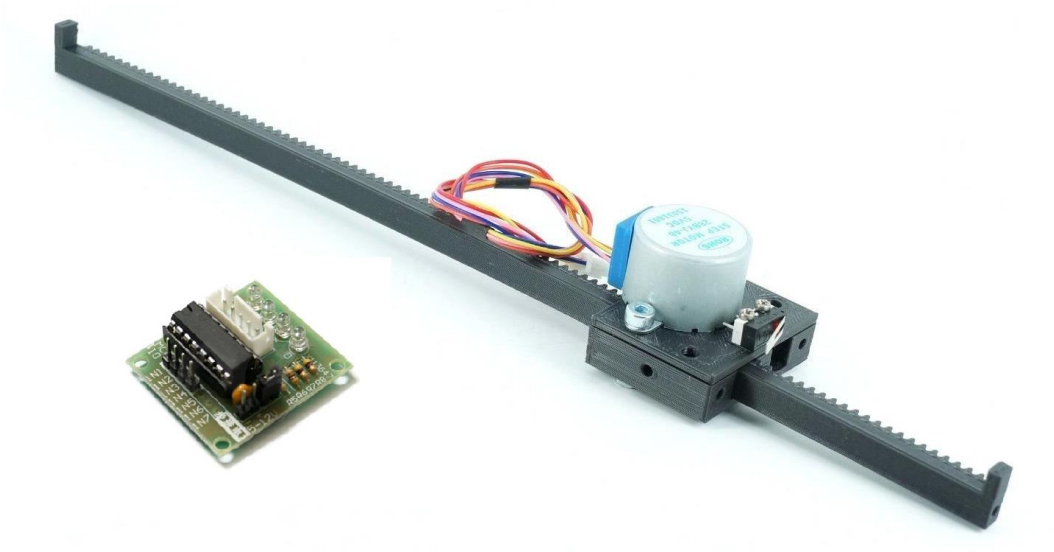


Abbildung 36 – Linear-Aktuator mit Stepper-Motor 28BYJ-48 und Treiberplatine ULN2003a (Otamat, 2018)

4.7.4 Raspberry Pi

Wie in Abschnitt 4.6 beschrieben, werden für die Clients Raspberry Pis verwendet. Für die ersten Tests in dieser Arbeit wird das Modell Raspberry Pi 2 Model B verwendet, da diese an der Fachhochschule bereits zur Verfügung stehen.

Raspberry Pis sind winzige Linux-basierte Computer, die mit Gehäuse etwa so groß wie zwei Smartphones sind. Sie sind für ihre Leistung sehr leistbar, eine Platine kostet in etwa 35€. Durch die kleine Bauweise und den kleinen Preis eröffnen sie ein riesen Feld für neue Projekte. Sie können beispielsweise als Mini-Server oder als Multimedia-Center betrieben werden (Kofler, Kuhnast, & Scherbeck, 2015, S. 21). Durch ihre integrierten GPIOs (General Purpose Input/Output) eröffnen sie einen weiteren riesigen Bereich für verschiedene

Elektronikprojekte, indem damit Daten verarbeitet und unterschiedlichste Funktionen gesteuert werden können (Kofler u. a., 2015, S. 15).

Der Raspberry Pi 2 Model B ist seit Februar 2015 auf dem Markt. Er hat eine vier Core CPU mit 900 MHz Taktfrequenz und einen Arbeitsspeicher von einem Gigabyte. Er verfügt über vier USB-2.0-Anschlüsse und einem 100-MBit-Netzwerkanschluss. Auf der Steckerleiste stehen 40 GPIO-Pins zur Verfügung. Außerdem hat er einen Broadcom Video-Core IV mit H.264 Encoder/Decoder, was ihn für dieses Projekt sehr eignet (Kofler u. a., 2015, S. 22). Abbildung 37 zeigt dieses Modell. Als Videoschnittstelle steht HDMI zur Verfügung. Außerdem gibt es eine 3.5 mm Klinkenbuchse für Audio und Video. Für den Betrieb mit einer Kamera existiert ein Camera Serial Interface (CSI) und ein Display Serial Interface (DSI). Über das microSD Interface kann eine microSD-Karte eingelegt werden, die als Speicher für Betriebssystem und Daten dient. Der Raspberry Pi wird über microUSB mit Strom versorgt.

Abbildung 37 – Raspberry Pi 2 Model B (Raspberry, 2018)

Wie im Abschnitt 4.6 behandelt, funktioniert der Raspberry Pi als Client. Er empfängt einen Stream bestehend aus Daten, nimmt sich die für ihn relevanten heraus und übergibt diese aufbereitet an seine Ausgabegeräte weiter. Das Blockschaltbild in Abbildung 38 veranschaulicht die Signalverarbeitung des Moduls. Der Raspberry Pi wird über ein LAN-Kabel ans Netzwerk der Installation abgebunden. Darüber empfängt er den Stream, der aus Bilddaten oder aus Daten, die das Bild repräsentieren, wie beispielsweise OpenGL Befehle, bestehen kann. Außerdem empfängt er über den Stream Daten über die Rotation und die Position

der Displays. Die decodierten oder gerenderten Bilder werden über HDMI an das Display weitergegeben. Die aufbereiteten Steuerdaten für Stepper-Motor und Servos können über die GPIO Pins ausgegeben werden. Der Taster, der an der Basis des Linear-Aktuators befestigt ist, gibt Rückmeldung über dessen Position. Diese Daten gelangen ebenfalls über die GPIO Pins in den Raspberry.

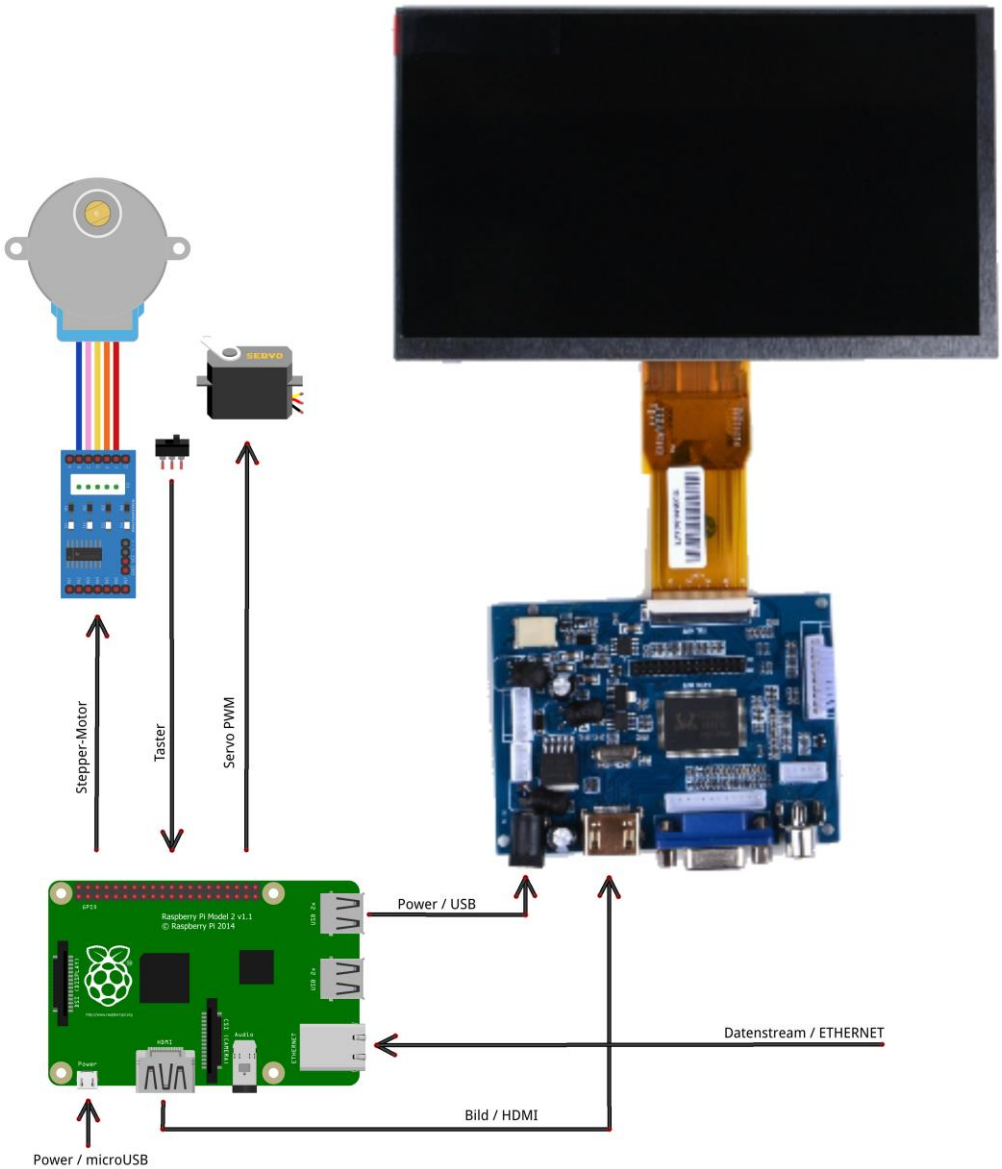


Abbildung 38 – Blockschaltbild Modul

Durch die in diesem Abschnitt besprochene Verwendung des Raspberry Pis als Client, ist es möglich von einem zentralen Punkt die gesamte Installation zu steuern. Dabei ist jedes Modul gleich aufgebaut, sie unterscheiden sich lediglich durch ihre ID. Somit ist es möglich die zentral generierten Steuer- und

Bildinformationen auf viele Punkte zu verteilen. Außerdem kann leicht die Anzahl der Module und somit die Komplexität der Installation skaliert werden.

Der nächste Abschnitt beschäftigt sich näher mit dem Zusammenbau dieser Elemente und geht auf die Reaktionen der externen Stakeholder ein.

4.8 Prototyp 1

Das Ziel des „Artefakt 9 – Prototyp 1“ ist es, die Anforderungen an die Konstruktion des Moduls zu untersuchen. Die im vorherigen Abschnitt 4.7 vorgestellten Komponenten – Display, Linear-Aktuator, Servos und Raspberry Pi – werden in diesem „Artefakt“ zusammengebracht und untersucht.

In der „Creation Session“ dieses „Artefakts“ werden für den Zusammenbau dieser Komponenten Materialien wie Karton, Kabelbinder und Metallteile von Meccano[11] Spielzeugen verwendet, um rasch Verbindungen herstellen zu können und um schnell verschiedene Positionierungen der Komponenten testen zu können. Abbildung 39 und Abbildung 40 zeigen den entstandenen „Prototyp 1“.

Die Anordnung hält sich sehr an die Skizze des „Artefakt 5“ in Abbildung 28. An der Basis des Linear-Aktuators, wo sich der Stepper-Motor befindet, wird das Modul an der Installation befestigt. Dieser verschiebt die Servos und das Display entlang einer Achse. An einer Spitze des Aktuators, in Abbildung 39 links zu sehen, sind die Servos montiert. Der erste Servo sorgt für die horizontale Schwenkbewegung. Darauf ist der zweite Servo montiert, der das Display vertikal neigt. Das Display inklusive Treiberplatine ist mittels Karton und Kabelbinder am zweiten Servo angebracht. Der Raspberry Pi würde auf der rechten Seite des Linear-Aktuators montiert werden. In diesem „Artefakt“ wurde er allerdings nicht angebracht.

[11] http://www.meccano.com/products

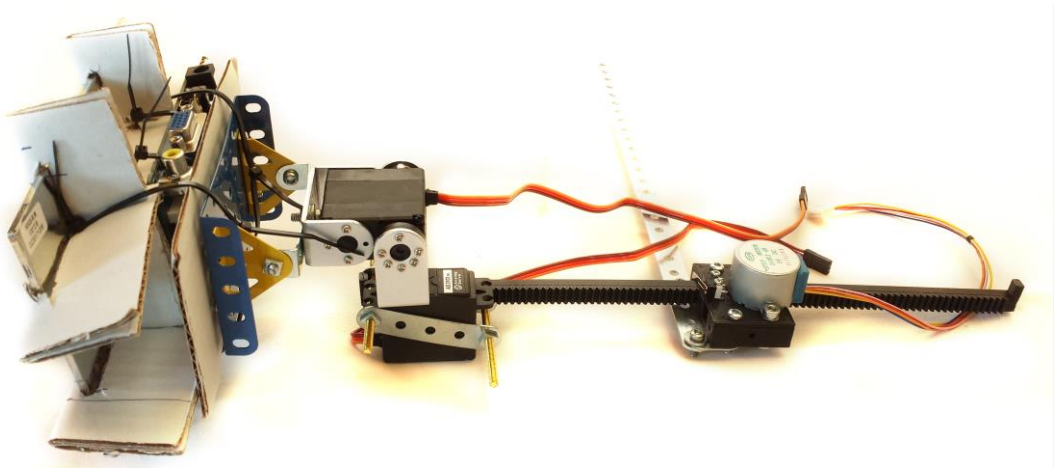


Abbildung 39 – Artefakt 9: Zusammenbau „Prototyp 1“ – Seitenansicht

In der sechsten „Engagement Session“ wurden erste Bewegungen mit diesem „Prototyp 1“ durchgeführt. Es nahm ein externer Stakeholder daran teil. Ziel dieser „Session“ war es Problemfelder dieser Konstruktion zu identifizieren und zu untersuchen. Dadurch entstanden einige wichtige Reaktionen.

Diese Konstruktion ermöglicht es dem Display verschiedene Positionen und Rotationen im Raum anzunehmen. Die Servomotoren sind stark genug, um das Display schnell rotieren zu lassen. Abbildung 40 zeigt diese Konstruktion mit nach oben schauendem Display. Obwohl die Grundfunktionen bei diesem Modul gegeben sind, ergeben sich einige Verbesserungsvorschläge. Eine der größten Problematiken dieses „Prototyp 1“ ist das große Gewicht und der lange Hebelarm der Displayeinheit. In diesem „Artefakt“ ist das Display und die dazugehörige Treiberplatine am Servo angebracht, daraus ergibt sich ein großes Gewicht. Außerdem entsteht durch die derzeitige Befestigung der Displayeinheit am Servo ein langer Hebelarm von etwa 10cm, wie in Abbildung 39 links ersichtlich.

Dadurch ergeben sich einige Nachteile: Beim abrupten Starten oder Stoppen einer Drehbewegung beginnt das ganze Modul zu schwingen und benötigt einige Sekunden um sich wieder zu beruhigen. Außerdem entsteht durch den langen Hebelarm ein riesiger Bewegungsradius für das Display und riesige Kräfte an den Servos. Würden für den Bau der Installation diese Module verwendet werden, könnten diese nicht eng nebeneinander positioniert werden, da jedes aufgrund der großen Bewegungsradien enormen Platzbedarf hätte.

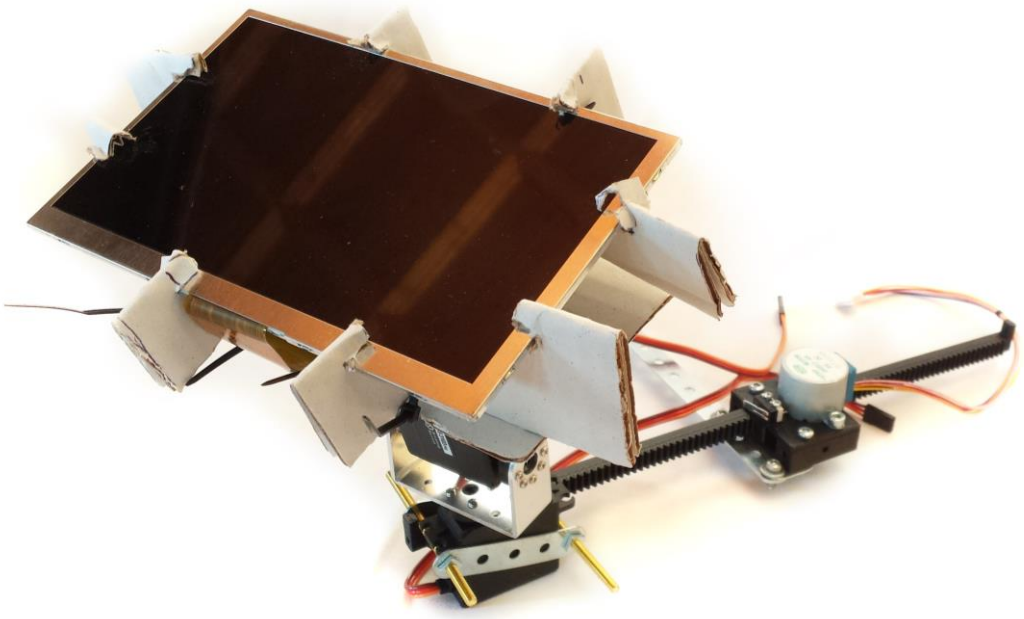


Abbildung 40 – Artefakt 9: Zusammenbau „Prototyp 1“ – Vorderansicht

Ein weiterer Nachteil dieser Konstruktion sind die unsauberen Drehbewegungen, die sich durch den Aufbau des Pan-Tilt-Servos ergeben. Theoretisch müssten die beiden Drehachsen der Servos, wie bei einem Kugelgelenk, ineinander liegen. Da das Schwenken und das Neigen von zwei verschiedenen Servos ausgeführt wird, ergibt sich immer ein bestimmter Versatz einer Achse, was wiederum immer zu einem gewissen Bewegungsradius führt.

Durch diese Bewegungstests entstanden in der sechsten „Engagement Session“ mit Hilfe eines externen Stakeholders einige Verbesserungsvorschläge. Einerseits soll für saubere Bewegungsverläufe die Treiberplatine des Displays nach hinten verlegt werden, um das zu bewegende Gewicht so weit wie möglich zu reduzieren. Andererseits soll die Drehbewegung auf nur eine Achse reduziert werden, um die Komplexität der Konstruktion zu reduzieren. Dadurch wird es möglich die Drehachse auf das Display zu legen, um den kürzest möglichen Hebelarm zu bekommen. Dadurch soll die Drehbewegung sauberer ausgeführt werden können.

Engagement Session	6
Artefakt	9 – Prototyp 1
Artefakt Typ	Elektronikkomponenten, Karton, Metallkonstruktion
Datum	04.04.2018

Anzahl der externen Stakeholder	1
Annahmen	• Grundsätzliche Anordnung und Positionierung zueinander • Schwenken und Neigen des Displays
Reaktionen	• Zu großer Abstand, Hebelwirkung • Zu viel Gewicht • Andere Position: Treiberplatine des Displays • Komplexität Pan und Tilt zu hoch

Tabelle 6 – Engagement Session 6

Die durch dieses „Artefakt 9“ entstandenen Reaktionen fließen in den Entstehungsprozess des nächsten „Artefakts 10 – Prototyp 2“ ein. Der folgende Abschnitt beschäftigt sich näher damit.

4.9 Prototyp 2

In diesem Durchlauf werden die gewonnenen Reaktionen aus dem vorherigen Abschnitt in Tabelle 6 in Form eines neuen Prototyps verarbeitet. Dieser „Prototyp 2“ wird in diesem Abschnitt vorgestellt.

Die Ursachen für die entstandenen Probleme beim „Prototyp 1“ waren vor allem der lange Hebelarm zwischen Drehachse und Display und das große Gewicht der Displayeinheit. Diese Ursachen wurden hier so weit wie möglich minimiert. Abbildung 41 und Abbildung 42 zeigen den „Prototyp 2“ von der Seite und von vorne.

Um die Komplexität der Bewegungen und den damit verbundenen Aufwand für Konstruktion jedes Moduls zu reduzieren, wird hier nur ein Servo für die Drehung des Displays verwendet. Dieser ist direkt am Display montiert, um den Hebelarm und die daraus resultierenden Kräfte zu minimieren.

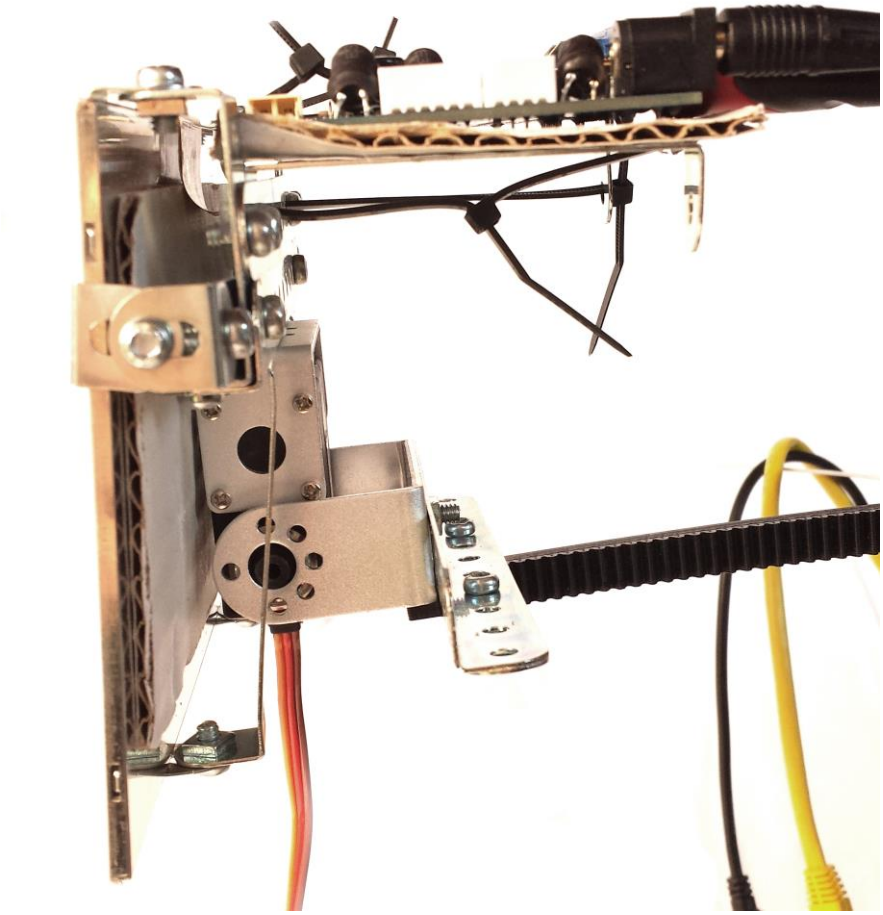


Abbildung 41 – Artefakt 10: Zusammenbau „Prototyp 2“ – Seitenansicht

Auf der linken Seite in Abbildung 41 ist das Display zu sehen. Oben waagrecht ist die Treiberplatine des Displays befestigt, die über ein Flachbandkabel das Display versorgt. Da zu diesem Zeitpunkt noch keine Verlängerung für das Flachbandkabel vorhanden ist, ist die Treiberplatine noch an der Displayeinheit befestigt. Zukünftig soll diese aber an einer anderen Position angebracht sein, damit die Displayeinheit lediglich aus dem Display und dessen Halterung besteht. Dadurch wird das Gewicht noch mehr reduziert, um die Rotationen der Displays zu begünstigen.

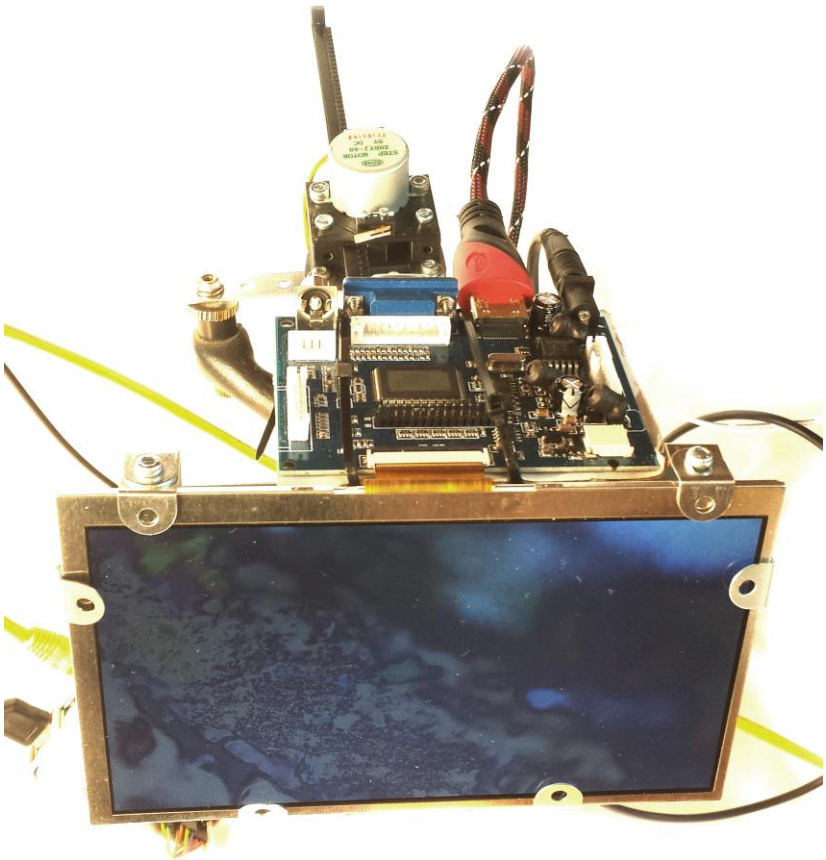


Abbildung 42 – Artefakt 10: Zusammenbau „Prototyp 2“ – Vorderansicht

Im Gegensatz zum „Prototyp 1“ ist der „Prototyp 2“ nur aus Metallteilen gefertigt. Dadurch ist dieser stabiler und weniger anfällig für Schwingungen. In der „Engagement Session 7“ nahmen zwei externe Stakeholder teil. Ihnen wurde der „Prototyp 2“ vorgestellt und daraus entstanden einige Reaktionen. Bei dieser „Session“ wurde auf dem Display ein gerendertes Video, auf dem Ameisen zu sehen sind, abgespielt. Dabei neigte sich das Display rhythmisch auf und ab. Dadurch entstand ein erstes Gefühl für den Inhalt und das Verhalten der Installation. Die Ausführung der Rotationsbewegung war zwar sehr zufriedenstellend, es kam aber ein weiteres Problem zum Vorschein. An den Maxima der Displayneigung, wie in Abbildung 42 erkennbar, erscheint der Inhalt des Displays dunkler als in Neutralstellung. Dies liegt an dem in Abschnitt 4.7.1 besprochenen Betrachtungswinkel des Displays von 70° links und rechts, 50° oben und 60° unten. In der nächsten Ausführung des Moduls soll deshalb der Servo am Display um 90° gedreht angebracht werden, sodass es nicht geneigt, sondern geschwenkt wird. Dadurch steht dem User ein größerer Betrachtungswinkel zur

Verfügung. Statt den 50° oben und 60° unten durch die vertikale Rotation entstehen durch die horizontale Rotation 70° links und rechts.

Eine weitere Reaktion entstand durch den Testlauf. Wie in Abbildung 41 zu sehen, befindet sich die Drehachse des Servos etwas unterhalb der Displaymitte. Daraus ergibt sich eine ungleiche Drehbewegung. Um die Rotation gleichmäßiger wirken zu lassen, soll in der nächsten Ausführung diese Achse in der Mitte des Displays positioniert werden. Außerdem soll, wie schon im vorherigen Abschnitt besprochen, die Treiberplatine des Displays an einer anderen Position befestigt werden, damit sich das zu rotierende Gewicht noch mehr verringert.

Engagement Session	7
Artefakt	10 – Prototyp 2
Artefakt Typ	Elektronikkomponenten, Metallkonstruktion
Datum	28.04.2018
Anzahl der externen Stakeholder	2
Annahmen	• Rotation einer Achse • Drehachse direkt am Display
Reaktionen	• Konstruktion gut • Drehbewegung: Schwenken statt Neigen • Zentrierung der Drehachse am Display • Andere Position: Treiberplatine des Displays

Tabelle 7 – Engagement Session 7

Die letzten beiden Abschnitte 4.8 und 4.9 haben die Anforderungen an die Konstruktion des Moduls untersucht. Durch die Reaktionen der externen Stakeholder wurden diese Anforderungen an die Modulkonstruktion vertieft. Abschnitt 4.11 geht näher auf die weitere Entwicklung des Moduls ein. Der nächste Abschnitt 4.10 beschäftigt sich näher mit der Ideenfindung für den Aufbau und die Konstruktion der gesamten Installation.

4.10 Virtuelle Manifestation

Der Abschnitt 4.5 setzte sich näher mit der Größe der Displays und der Installation auseinander. In dieser fünften „Engagement Session“ in Tabelle 5 kam auch die Idee auf, Virtual Reality als Werkzeug zur Ideenfindung zu nutzen. Dieser Abschnitt geht näher auf diese Idee ein und beschäftigt sich mit den Reaktionen der externen Stakeholder.

Das „Artefakt 11 und 12“ wurde mit Hilfe der Oculus Rift[12] CV1 und den dazugehörigen Touch Controllern erstellt. Dieses verwendete Set besteht aus der Virtual-Reality-Brille, den beiden Touch Controllern und den beiden Sensoren. Abbildung 43 zeigt diese Komponenten. Die Sensoren werden über USB 3.0 an einen Rechner angeschlossen. Die beiden Touch Controller verfügen über eine kabellose Verbindung. Die Brille wird mit USB 3.0 und HDMI angeschlossen. Alle notwendigen Daten werden im Rechner verarbeitet. Dadurch ist es möglich sich in einem Bereich von etwa 3 x 3 Metern, abhängig von der physischen Raumgröße und der Kabellänge, umher zu bewegen. Die Sensoren tracken den Kopf und die Hände des Benutzers. Dadurch können die physischen Bewegungen in den virtuellen Raum übertragen werden und der Benutzer kann sich somit in den dreidimensionalen, virtuellen Raum begeben. Auf dem Rechner läuft die Oculus Desktop-App[13], mit der die Geräte kalibriert werden können und unzählige Virtual-Reality Applikationen gestartet werden können.

Abbildung 43 – Oculus Rift (Spinger, 2016)

[12] https://www.oculus.com/rift/

[13] https://www.oculus.com/setup/

Für das „Artefakt 11 und 12“ wurde Oculus Medium[14] verwendet. Diese App stammt direkt von den Herstellern der Oculus Rift. Mit ihr ist es möglich sich im virtuellen Raum zu bewegen und verschiedenste Objekte zu erschaffen, zu verformen und zu bemalen. Dafür stehen verschiedene Werkzeuge, wie beispielsweise Modelliermasse oder Pinsel, zur Verfügung. Objekte oder Gruppen können wie bei klassischen Fotobearbeitungsprogrammen sichtbar, unsichtbar, dupliziert, gelöscht und so weiter werden. Außerdem können in die Szene 3D-Modelle anderer Programme importiert werden. Ebenso können erstellte Objekte als 3D-Modell exportiert werden. Weiters können Lichter im Raum positioniert werden, die die Szene beleuchten. Ein weiteres wichtiges Feature sind die Foto- und Videoaufnahmefunktionen, mit der die Arbeiten in den üblichen 2D-Raum übersetzt werden können. Es ist möglich die Kamera auf dem Kopf, auf der Hand oder irgendwo im Raum zu positionieren. Dadurch können Kamerafahrten und Aufnahmen unterschiedlichster Einstellungen angefertigt werden. Die folgenden Abbildungen wurden ebenfalls mit dieser Funktion erstellt.

Ziel dieser „Creation Session“ ist es grundsätzlich Ideen für die Anordnung der Module zu sammeln und eine Vorstellung über die Wirkung der Installation zu bekommen. Dafür wurde zu Beginn die Form des Moduls mit den ungefähren Maßen des realen Prototyps modelliert und bemalt. Abbildung 44 zeigt einen Ausschnitt dieses Prozesses. Mit dem Pinsel kann das modellierte Modul farblich gestaltet werden. Hier wurde farblich der Inhalt des Displays nachgebildet.

[14] https://www.oculus.com/medium/

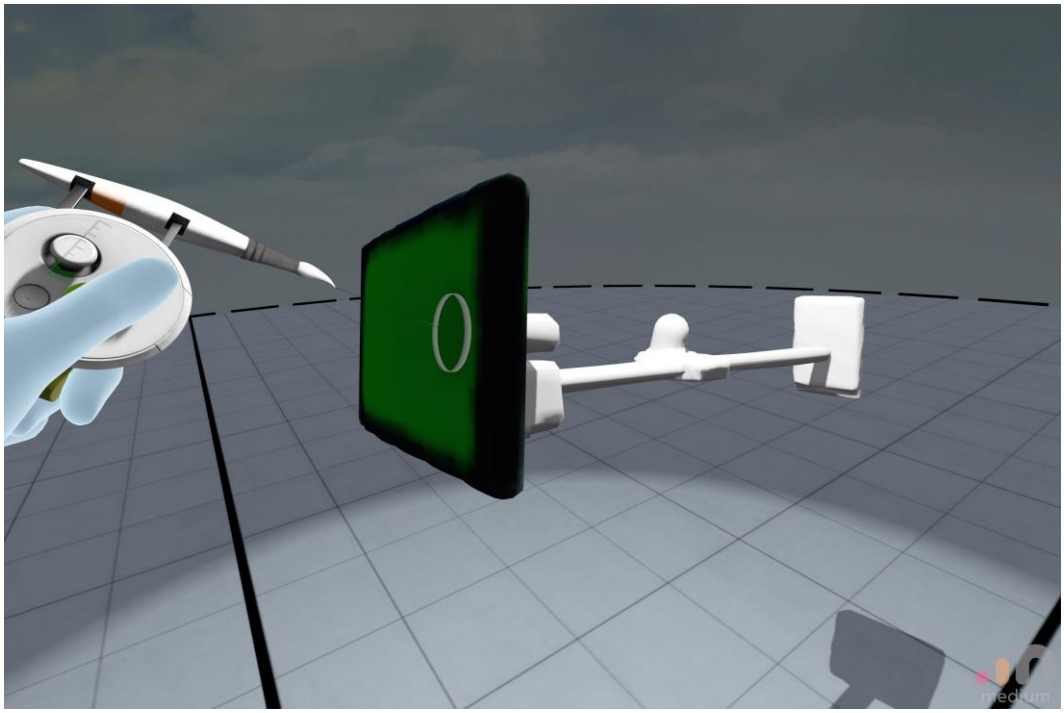


Abbildung 44 – Bemalung des Moduls in Oculus Medium

Anschließend konnte das fertig gestaltete Modul 20-mal dupliziert und auf einzelne Ebenen gelegt werden. Dadurch konnten die einzelnen Module frei im virtuellen Raum bewegt werden. Zu Beginn entstand das „Artefakt 11“ als plane Videowand. Abbildung 45 und Abbildung 46 zeigen dieses Modell von der Vorder- und Rückseite. Dadurch entstand ein erster Eindruck über die Anordnung und die Größe der Installation. Außerdem bildet dieses „Artefakt“ eine für den externen Stakeholder enorm immersive Basis für die achte „Engagement Session“ in Tabelle 8. In diesem Modell zeigen die Displays alle den gleichen Inhalt. In der physikalischen Umsetzung könnte sich der Inhalt ebenso über die gesamte Fläche erstrecken.

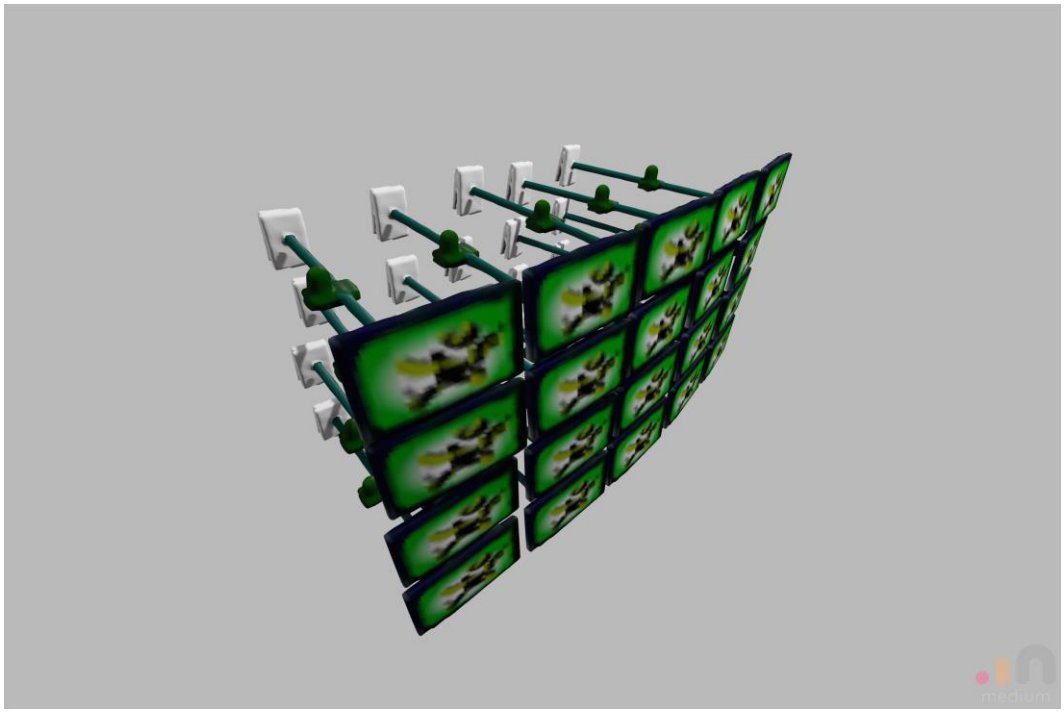


Abbildung 45 – Artefakt 11: Virtual-Reality Visualisierung „Virtuelle Manifestation 1“ – Vorderansicht

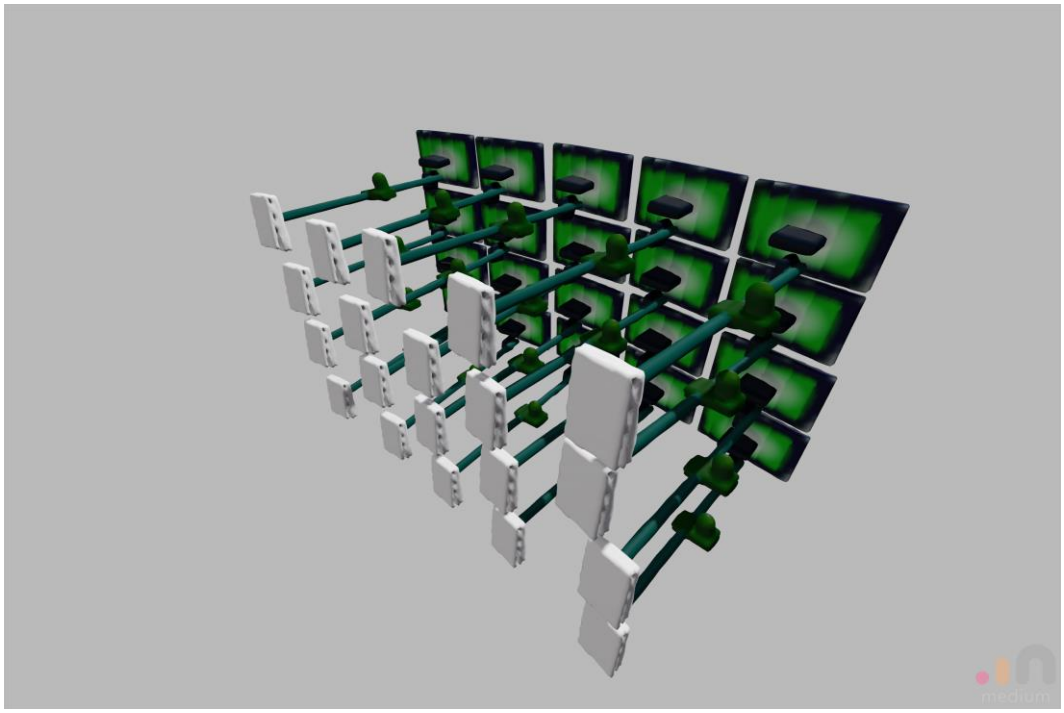


Abbildung 46 – Artefakt 11: Virtual-Reality Visualisierung „Virtuelle Manifestation 1“ – Rückansicht

Nach erster Begutachtung des „Artefakts 11“ wurde dieser Gestaltungsprozess wiederholt und es entstand eine weitere mögliche Anordnung der Installation. Abbildung 47 zeigt einen Ausschnitt aus dem Entstehungsprozess des „Artefakt 12“. Abbildung 48 zeigt das fertige „Artefakt 12“ von der Rückseite. Diese Form der Anordnung erinnert mehr an die Innenseite einer Halbkugel. Dabei sind die einzelnen Module zueinander gedreht. Sie sind sternförmig von innen nach außen angeordnet, sie weisen deshalb unterschiedliche Neigungen auf. Durch die dreidimensionale, tunnelförmige Anordnung entsteht ein sehr immersives Gefühl und ein leichter Sog nach innen. Nach Fertigstellung dieses „Artefakts“ konnte um die Installation herumgegangen werden und die gesamte Anordnung begutachtet werden. Auf der Rückseite, wie in Abbildung 48 zu sehen, erkennt man die konfuse Anordnung der Linear-Aktuatoren. Hier können die Module frei im virtuellen Raum schweben. In der physikalischen Umsetzung dieser Installation ist aber eine Konstruktion erforderlich, die diese Module an den richtigen Stellen im Raum fixiert. Diese Virtual-Reality Visualisierung gibt eine gute Vorstellung über die Notwendigkeit und die Komplexität dieser notwendigen Konstruktion.

Abbildung 47 – Artefakt 12: Virtual-Reality Visualisierung „Virtuelle Manifestation 2“ – Vorderansicht

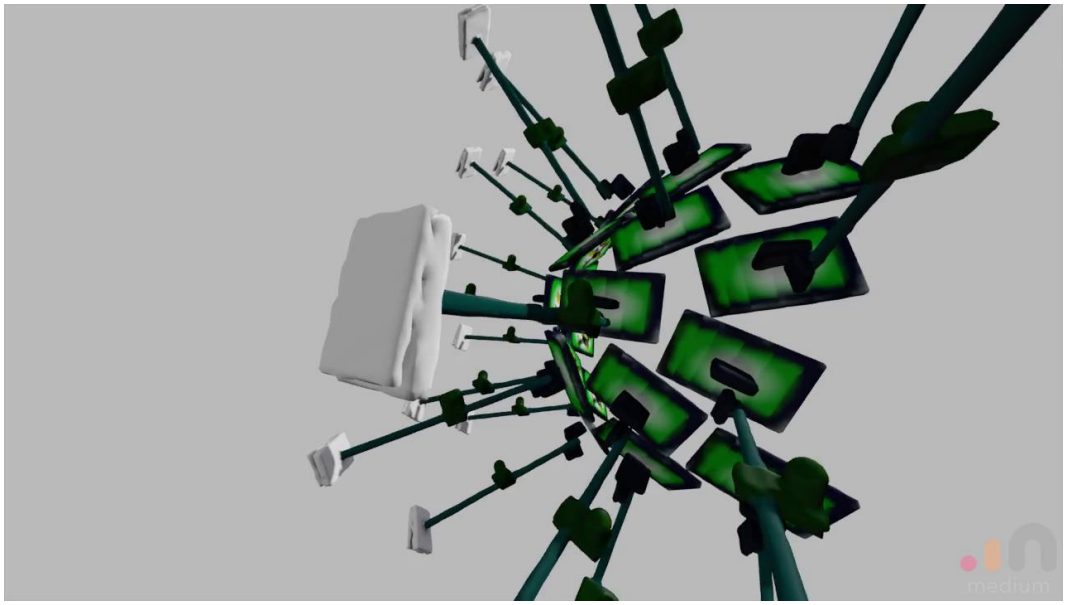


Abbildung 48 – Artefakt 12: Virtual-Reality Visualisierung „Virtuelle Manifestation 2“ – Rückansicht

Wie obig angeschnitten bietet Oculus Medium die Möglichkeit die Szene zu beleuchten. Einerseits kann die umgebende Sphäre farblich verändert werden und andererseits können Scheinwerfer im Raum positioniert werden. Dadurch ist es möglich schnell verschiedene Stimmungen durch unterschiedliche Lichtsetzungen auszuprobieren. Abbildung 49 zeigt eine mögliche Lichtstimmung, in der die Installation präsentiert werden kann. In dieser Abbildung wurde die halbkugelförmige Installation dupliziert, sie würde hier aus 40 Modulen bestehen. Durch die parallele Anordnung entsteht der Eindruck zweier Augen.

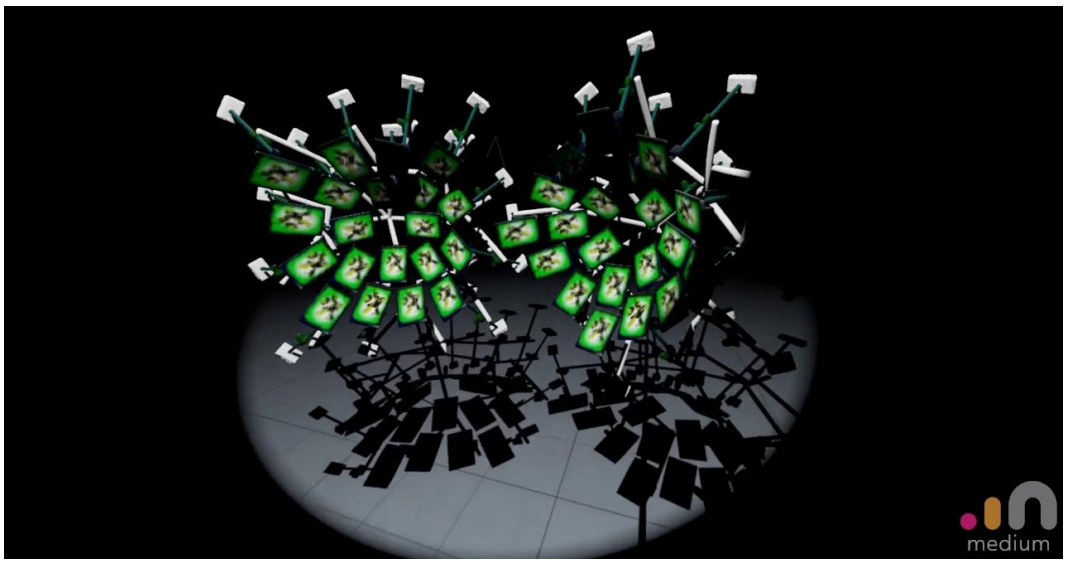


Abbildung 49 – Virtuelle Ausleuchtung in Oculus Medium

In der achten „Engagement Session“ wurden die beiden Virtual-Reality Visualisierungen „Virtuelle Manifestation 1 und 2“ drei externen Stakeholdern gezeigt. Sie tauchten mit Hilfe der Oculus Rift in die virtuelle Welt ein und konnten darin die beiden „Artefakte 11 und 12“ von allen Seiten begutachten. Durch deren Erlebnisse entstanden einige Reaktionen und gute Ideen.

Der Einsatz von Virtual-Reality als Werkzeug zur Ideenfindung stellte sich als sehr hilfreich heraus. Die externen Stakeholder bekamen so einen immersiven Eindruck über die Vorstellungen des Autors dieser Arbeit und konnten so produktives Feedback geben. Eine weitere Reaktion auf die beiden „Artefakte“ war die Notwendigkeit der Konstruktion der Installation. Die Positionierung der Module gab enormen Aufschluss über die Wichtigkeit und die Anforderungen der Installation. Bei einer Anordnung als Displaywand wie in Abbildung 46 würde die Konstruktion einfacher ausfallen als bei einer konfuseren Anordnung wie in Abbildung 48, da die Module hier unterschiedliche Rotationen aufweisen. Die Planung der Konstruktion im virtuellen Raum kann also für eine erfolgreiche Umsetzung enorm hilfreich sein.

Für die Anfertigung einer Konstruktion, kam in dieser „Engagement Session“ von einem externen Stakeholder die Idee ein Kellerschachtgitter wie in Abbildung 50 beispielhaft abgebildet zu verwenden. Der Vorteil hierbei wäre die gleichmäßige Rasterung. Dadurch könnten die Linear-Aktuatoren der einzelnen Module je nach gewünschtem Abstand zueinander positioniert werden. Dadurch wäre aber nur eine planare Anordnung wie im „Artefakt 11“ möglich. Außerdem beträgt üblicherweise der Maschenabstand dieser Gitter 30mm, was für die verwendeten Aktuatoren zu klein wäre.

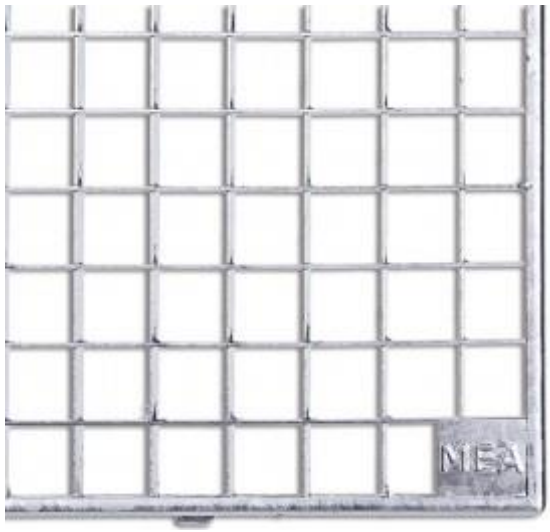


Abbildung 50 – Kellerschachtgitter (Baustoff, 2018)

In dieser „Engagement Session“ entstand eine weitere Idee als Skizze bezüglich Anordnung und Konstruktion. Der zweite externe Stakeholder brachte die Idee ein die Installation etwa in der Größe einer Softbox zu gestalten, um diese an einem C-Stand befestigen zu können. Die entstandene Skizze ist in Abbildung 51 zu sehen. Durch diese Befestigung wäre es möglich, die Installation für Beleuchtungszwecke in Table-Top-Shootings zu verwenden. Hierbei werden kleine Objekte auf einem Tisch von verschiedenen Seiten ausgeleuchtet, um das Objekt in das perfekte Licht zu rücken. Durch eine Krümmung der Fläche nach innen wie in „Artefakt 12“ könnte die Installation über das zu beleuchtende Objekt gestülpt werden. Dadurch wäre es möglich, von einem zentralen Punkt aus, die gesamte Beleuchtung zu steuern ohne die Position der Leuchtmittel verschieben zu müssen.

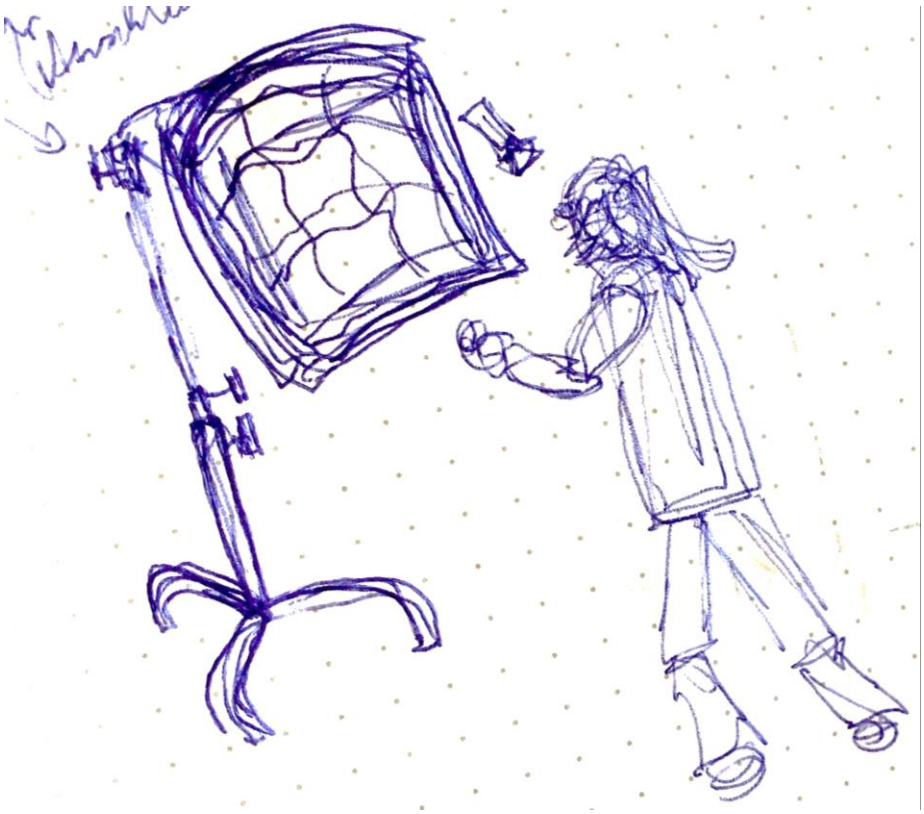


Abbildung 51 – Befestigung der Installation auf C-Stand

Im Vergleich der beiden „Artefakte“ fiel die Entscheidung der externen Stakeholder auf die komplexere Form des „Artefakt 12“. Alle Drei beschrieben diese Form als interessanter. Durch ihre dreidimensionale Anordnung entsteht ein immersiveres Gefühl. Sie beschrieben es so, als ob es sie in das Loch hineinziehen würde. Die zukünftigen Entwicklungen fokussieren sich daher auf eine nicht planare Anordnung der Installation.

Durch diese konfusere Anordnung entsteht auch die Notwendigkeit einer aufwendigeren Konstruktion, die alle Module in Position hält. Erste Überlegungen bezüglich dieser Konstruktion wurden ebenfalls im virtuellen Raum gemacht. Wie in Abbildung 52 ersichtlich, wurden alle Module mit Hilfe von Stangen verbunden. Durch diesen Prozess wurden die Komplexität dieser Konstruktion und die Notwendigkeit auf eine vertiefte Überlegung hinsichtlich Materials, Form und Anordnung der Konstruktion ersichtlich.

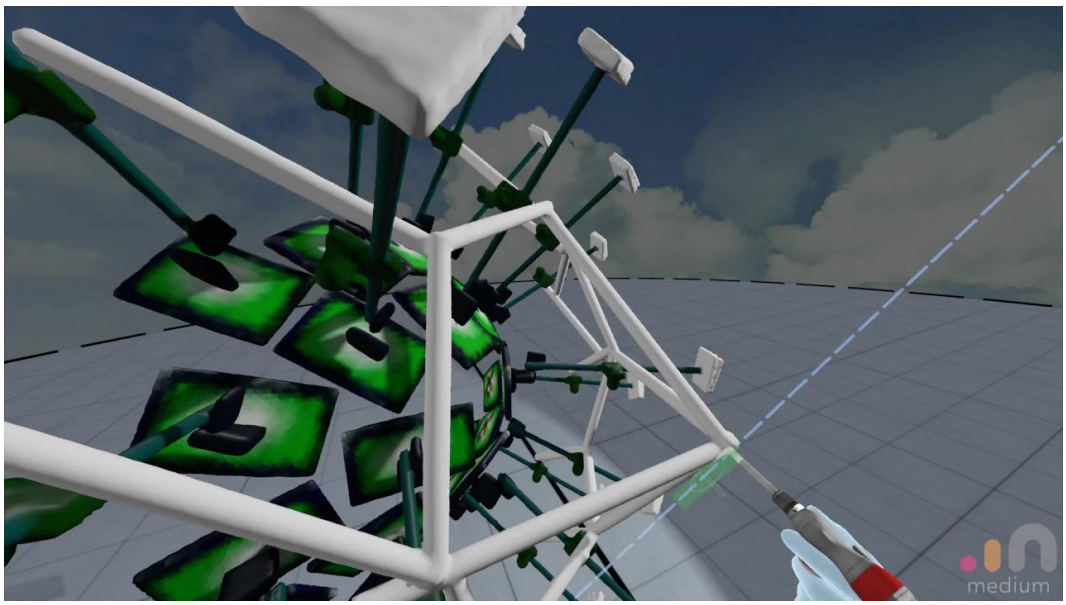


Abbildung 52 – Erste Überlegungen bezüglich Konstruktion der Installation

Die letzten Reaktionen entstanden bei der Betrachtung der beleuchteten Szene wie in Abbildung 49 ersichtlich. Diese Szene ist allgemein sehr dunkel gehalten. Die Installation wird über die Displays bespielt und nur von einem einzigen Spot beleuchtet um die Installation in den Fokus zu rücken. Alle drei externen Stakeholder unterstützten diese Art der Präsentation. Die Umgebung der Installation soll daher schwarz gehalten werden und durch dezente Lichtsetzung unterstützt werden.

Engagement Session	8
Artefakt	11 – Virtuelle Manifestation 1 12 – Virtuelle Manifestation 2
Artefakt Typ	Virtual-Reality Visualisierungen
Datum	16.05.2018

Anzahl der externen Stakeholder	3
Annahmen	• Unterschiedliche Anordnungen
Reaktionen	• Einsatz von VR zur Ideenfindung sehr gut • Art der Umgebung: Dunkel, mit dezenter Lichtsetzung • Nicht-planare Anordnung der Installation • Konstruktion der Installation komplex und daher enorm wichtig

Tabelle 8 – Engagement Session 8

4.11 Prototyp 3

Im vorherigen Abschnitt 4.10 wurden die ersten Ideen für die Form der gesamten Installation gesammelt. Es wurde sich auf eine nicht-planare Anordnung festgelegt und die Komplexität und Wichtigkeit der Konstruktion begriffen. Dieser Abschnitt greift die Reaktionen und Ergebnisse vom Prototyp 2 aus der Engagement Session 7 auf und versucht Lösungen dafür zu finden.

Wie in Tabelle 7 ersichtlich, waren die Kernthemen des Prototyp 2 die Änderung von Neigen auf Schwenken, die Zentrierung des Servos sowie die andere Positionierung der Displaytreiberplatine. Für den Bau des Prototyp 3 wurden eigene Displayhalterungen 3D gedruckt. Diese Halterungen werden auch bei der finalen Installation verwendet. Der Abschnitt 6.1 geht näher auf die Fertigung der Halterungen und die Behebung der Problemstellungen ein. Dieser Abschnitt fokussiert sich mehr auf die Machbarkeit der Umsetzung des Moduls.

Wie in Abbildung 53 ersichtlich wurde der Servomotor so angebracht, dass sich seine Drehachse in der Displaymitte befindet. Um die Displaytreiberplatine vom Display trennen zu können, wurde am Rahmen ein Verbinder für das Flachbandkabel angebracht. Die Verlängerung des Flachbandkabels wurde unter dem Servomotor verlegt. Es macht darunter einen 90 Grad Knick. Dadurch kann die Displaytreiberplatine weiter entfernt angebracht werden. Das, durch den Servo, zu drehende Gewicht wird dadurch reduziert, was in besserer Performance resultiert.

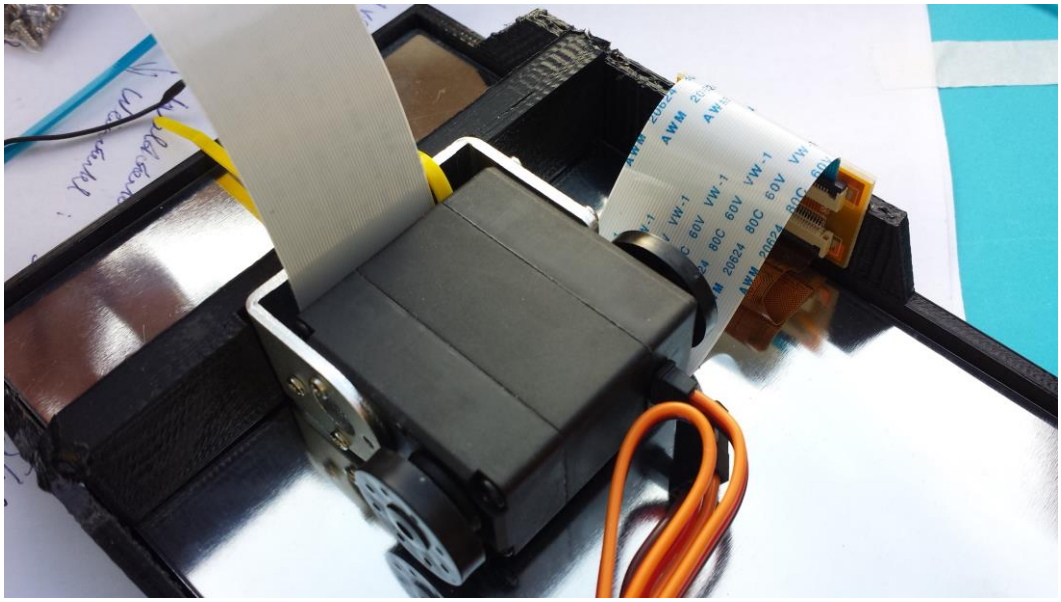


Abbildung 53 – Verlängerung Flachbandkabel

Diese Servo-Display-Kombination wird durch das, in Abschnitt 6.1.1.4 beschriebene, Verbindungselement an der Zahnstange des Linear-Aktuators angebracht. An diesem Verbindungselement befindet sich seitlich nochmals ein Verbinder, mit dem das Flachbandkabel verlängert werden kann. Wie in Abbildung 54 links ersichtlich, verläuft dieses Flachbandkabel bis nach hinten. An der Rückseite des Moduls befindet sich die Displaytreiberplatine, sowie das Raspberry Pi. Die Basis des Linear-Aktuators ist in der Mitte an einer provisorischen Halterung montiert. Dieser funktionierende, bewegliche Prototyp 3 bildet das „Artefakt“ in der neunten Engagement Session.

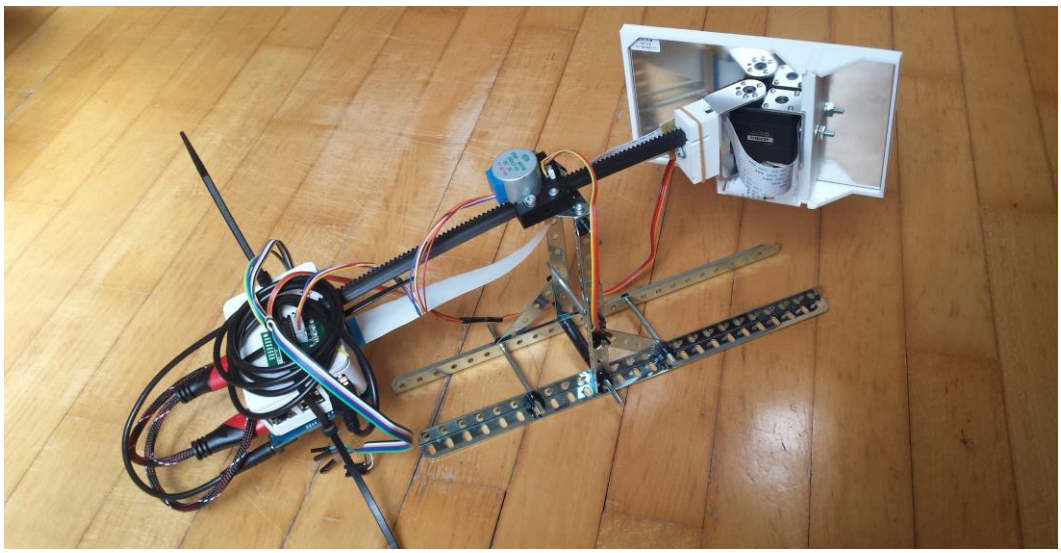


Abbildung 54 – Artefakt 13: „Prototyp 3“

In dieser neunten Engagement Session nahmen zwei externe Stakeholder teil. Dabei wurde das „Artefakt“ 13 vorgeführt und alle Bewegungen getestet. Die Reaktionen der externen Stakeholder waren gegenüber der Machbarkeit optimistisch. Der einzige Punkt für Verbesserungen wurde in der anderen Positionierung der Displaytreiberplatine und des Raspberry Pis gesehen. Dadurch würde das Gewicht am Linear-Aktuator reduziert werden, was wiederum in stabilerer Performance resultiert.

Engagement Session	9
Artefakt	13 – Prototyp 3
Artefakt Typ	Funktionierender, beweglicher Prototyp
Datum	28.06.2018
Anzahl der externen Stakeholder	2
Annahmen	• Positionierung der Komponenten • Verkabelung
Reaktionen	• Gewicht am Linear-Aktuator reduzieren

Tabelle 9 – Engagement Session 9

4.12 Das letzte Artefakt

Der vorherige Abschnitt beschäftigte sich mit der Machbarkeit des Moduls. Dieser Abschnitt beschäftigt sich mit der Machbarkeit der gesamten Installation. Technische Problemstellungen sollen hier erkannt und Lösungen dafür gefunden werden.

Obwohl die finale Installation nicht die Form einer Fläche aufweisen soll, wird das letzte „Artefakt“, aufgrund geringerer Komplexität, aus drei Modulen, planar angeordnet, zusammengesetzt. Durch den Bau dieses „Artefakts“ sollen Platzprobleme erkannt und besondere Eigenheiten, die aufgrund der Bewegung auftreten, festgestellt werden. Außerdem soll ein erster Eindruck durch das Zusammenspiel mehrerer Module entstehen.

Als Konstruktionsmittel, für den Bau des letzten „Artefakts“, wird das Holz eines alten Lattenrosts verwendet. Dieses zeichnet sich durch seine geringe Dicke und hohe Stabilität aus. Wie in Abbildung 55 ersichtlich, wird die Basis des Linear-

Aktuators so am Holzbrett montiert, dass das Modul normal dazu steht. Dadurch kann das Holzbrett an einer Halterung fixiert werden und das Display kann sich frei im Raum bewegen.

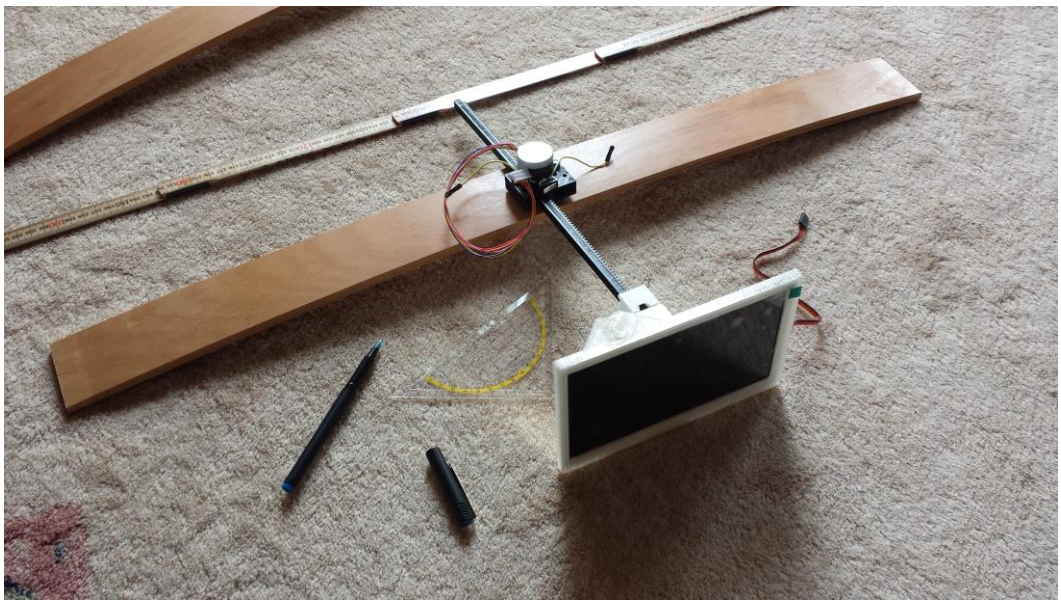


Abbildung 55 – Artefakt 13: „Das letzte Artefakt“ Zusammenbau

Nach der Befestigung aller drei Module wurden diese verkabelt. Wie in Abbildung 56 zu sehen, sind die Raspberry Pis zwischen den Linear-Aktuatoren montiert. Die Displaytreiberplatinen befinden sich an der Unterseite des Holzbretts.

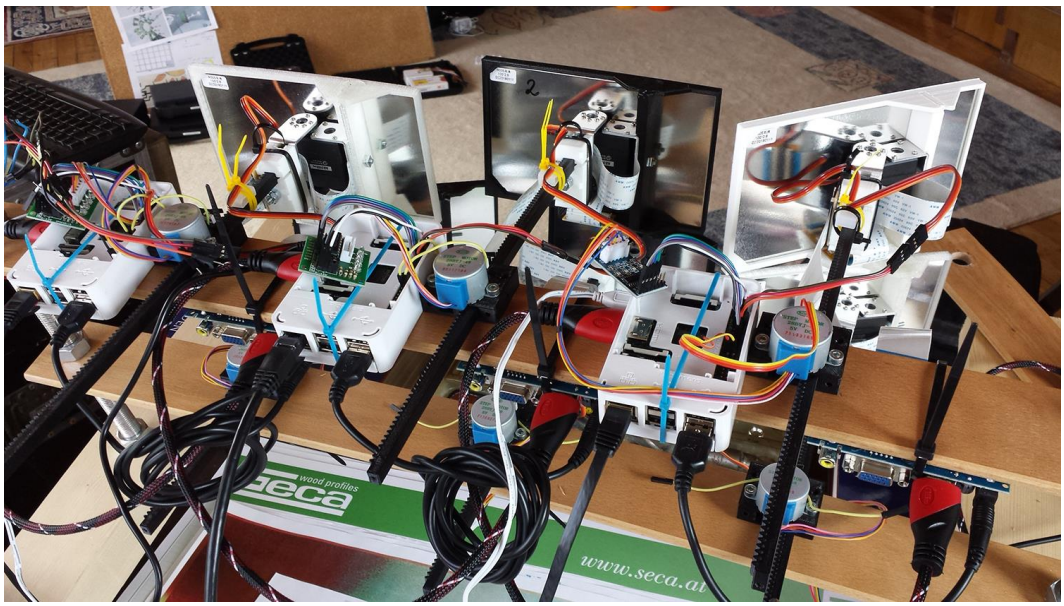


Abbildung 56 – Artefakt 13: „Das letzte Artefakt“ Verkabelung

Dieser Aufbau wurde in Aktion getestet. Dabei wurde von einem vierten Raspberry Pi aus, ein Video mit dem „BigBuckBunny“ Testvideo gestreamt. Jedes von den drei Modulen empfing diesen Stream und schnitt sich seinen Teil heraus, sodass, wie in Abbildung 57 ersichtlich, ein gesamtes Bild über die drei Displays zu sehen war. Für diese Aufteilung wurde PiWall[15] verwendet. Die Bewegungsdaten wurden von einem externen Laptop aus über vvvv[16] über Netzwerk an die Module gesendet.

Abbildung 57 - Artefakt 13: „Das letzte Artefakt“ Engagement Session

An der zehnten Engagement Session nahmen drei externe Stakeholder teil. Grundlage dafür war die, in Abbildung 57 zu sehende, Installation. Es entstand ein erster Eindruck über die Wirkung von bewegten Displays. Durch die gestreamten Bilddaten ergaben sich leichte zeitliche Verschiebungen zwischen den Displays. Die Qualität des Bildes war akzeptabel aber nicht überzeugend. Selten aber doch waren Codierungsartefakte zu sehen. Es ergab sich eine Latenz von etwa ein bis zwei Sekunden zwischen Senden und Empfangen des Videostreams. Im Gespräch der zehnten Engagement Session kam heraus, das Videostreaming keine Option, für eine qualitativ hochwertige, immersive Installation, darstellen kann. Wenn die Installation durch Interaktivität auf den Benutzer reagieren soll, ist eine Latenz von über einer Sekunde nicht akzeptabel. Deshalb soll der Content

[15] http://www.piwall.co.uk/

[16] https://vvvv.org/

der finalen Installation auf den Raspberry Pis erzeugt werden, sodass sich keine spürbare Latenz ergibt.

Die zweite wichtige Reaktion der zehnten Engagement Session bezog sich auf die hohe Komplexität und geringe Stabilität, die die Linear-Aktuatoren mit sich ziehen. Wie in Abbildung 56 ersichtlich entsteht ein enormer Verkabelungsaufwand, der eine geringe Ausfallsicherheit mit sich zieht. Aufgrund der Vielzahl an Kabeln ergeben sich ebenso viele Stellen, an denen sich eine Verbindung trennen kann. Deshalb sollen im Zuge des Baus der finalen Installation zwei verschiedene Module entwickelt werden. Ein Modul soll die gleichen Eigenschaften besitzen wie die hier verwendeten Module, nämlich lineare Bewegung und Rotation. Das zweite Modul soll dagegen nur rotierbar sein. Dadurch fällt die Komplexität des Linear-Aktuators und die damit verbundenen Verkabelungen weg. So soll die Performance der gesamten Installation mehr Stabilität erlangen und der Platzbedarf pro Modul geringer werden, sodass die finale Installation kompakter umgesetzt werden kann.

Engagement Session	10
Artefakt	13 – Das letzte Artefakt
Artefakt Typ	Bespielbare, bewegliche Installation aus 3 Modulen
Datum	15.07.2018
Anzahl der externen Stakeholder	3
Annahmen	• Positionierung der Komponenten • Verkabelung • Linear-Aktuatoren und Servos • Streaming
Reaktionen	• Streaming zu langsam • Entwicklung von 2 verschiedenen Modulen

Tabelle 10 – Engagement Session 10

5 Technische Umsetzung & Tools

Im vorherigen Kapitel 4 wurden in enger Zusammenarbeit mit den externen Stakeholdern die wichtigsten Punkte der Installation ausgearbeitet und aus verschiedenen Perspektiven betrachtet. Unterschiedliche Ideen wurden besprochen, erste Prototypen angefertigt und getestet sowie Skizzen über die Form und Größe der Installation erstellt. Aus all diesen Entscheidungen und Einflüssen wurde die Installation dieser Arbeit erschaffen. Dieses Kapitel geht näher auf die technische Umsetzung ein und beschreibt praktische Tools, welche zur Umsetzung notwendig waren.

5.1 Überblick Umsetzung

Die interaktive Installation besteht aus Sensorik, Datenverarbeitung, Visualisierung, Aktorik und Sonifikation. Dabei übernimmt verschiedene Hard- und Software diese Aufgaben. Für die Sensorik wird ein Ultraschallsensor verwendet, der über die serielle Ausgabe eines Arduino-Boards in die Steuereinheit, den Laptop eingelesen wird. Dieser übernimmt die Datenverarbeitung und generiert für die gesamte Installation Steuerdaten, mit denen die Position, sowie der Content der Displays gesteuert werden kann. Auch die Sonifikation, wird vom Laptop aus, umgesetzt. Alle beteiligten Geräte sind über WLan mit dem Netzwerk verbunden. Die Steuerdaten gelangen übers Netzwerk zu den Clients, welche mit Raspberry Pis realisiert werden. Diese empfangen die Steuerdaten und wandeln diese in Bildsignale und Signale für die Motoren um. Die Module mit den Raspberry Pis übernehmen also die Aufgabe der Visualisierung und der Aktorik.

Die Installation besteht dabei aus 16 einzelnen Modulen. Wie in Abschnitt 4.12 besprochen, soll dabei der Content auf den Raspberry Pis errechnet werden und dieser von außen gesteuert werden. Dadurch verlagert sich die Rechenleistung auf die Clients. Die Latenz wird so nicht mehr spürbar. Die Möglichkeit einer Skalierung der Installation ist dadurch ohne Probleme gegeben.

Wie aus der Engagement Session in Abschnitt 4.12 hervorgegangen ist, werden zwei verschiedene Module entwickelt, da die Verwendung der Linear-Aktuatoren der Robustheit der gesamten Installation schadet und die Komplexität enorm erhöht. Deshalb werden insgesamt 4 Module entwickelt, welche mittels Linear-Aktuatoren nach vor und zurück bewegt werden können. Dieses Modul 1 wird in

Abschnitt 6.3 näher erklärt. Die anderen 12 Module werden ohne Linear-Aktuator umgesetzt. Dieses Modul 2 wird in Abschnitt 6.4 beschrieben.

5.1.1 Signalverarbeitung Installation

Wie im obigen Abschnitt angeschnitten, kann die gesamte Installation von einem Laptop aus gesteuert werden. Abschnitt 7.1 beschreibt diese Steuereinheit näher. Wie in Abbildung 58 ersichtlich, werden vom Laptop aus Steuersignale übers Netzwerk zu den Raspberry Pis gesendet. Dafür werden die Programme Touchdesigner[17] und vvvv[18] verwendet. Touchdesigner liest die seriellen Daten des Ultraschallsensors über das Arduino-Board ein und generiert daraus States, die den Abstand des Benutzers wiederspiegeln. Daraus werden Steuerdaten errechnet, welche den Content der Module gestalten. Wie unterhalb des Touchdesigner-Kästchens zu sehen werden OSC-Werte über die Adressen eyeXYZ gesendet, um die Position der Kamera innerhalb der virtuellen Szene festzulegen. Jede Adresse ist für einen Parameter innerhalb der Szene verantwortlich. Diese, für jeden Client individuellen, OSC-Daten werden von jedem Raspberry Pi in dem Programm Processing[19] empfangen und können so Bildänderungen verursachen. So kann also der Content der Installation gestaltet werden. Abschnitt 7.3 geht näher auf den Processing Sketch ein.

Die Berechnungen der Positions-Steuerdaten finden im Programm vvvv am Laptop statt. vvvv bekommt dabei die States der Sensorik über OSC, PC-intern von Touchdesigner. Dadurch kann vvvv Bewegungsdaten, die auf den Benutzer der Installation eingehen, berechnen. Diese werden über UDP ebenfalls zu allen Clients gesendet. Diese Daten bestehen aus einem String. Dieser wird aus zwei Zahlen zusammengesetzt, welche durch ein Komma getrennt sind. Die erste, im Bereich von 0 bis 90, gibt die gewünschte Rotation des Displays in Grad wieder, die zweite die gewünschte Position des Linear-Aktuators, im Bereich von 0 bis 2800. Auf der Clientseite empfängt ein Python-Skript diese Daten und gibt diese aufbereitet den GPIO Pins weiter, die damit die Motoren steuern.

[17] https://www.derivative.ca/

[18] https://vvvv.org/

[19] https://processing.org/

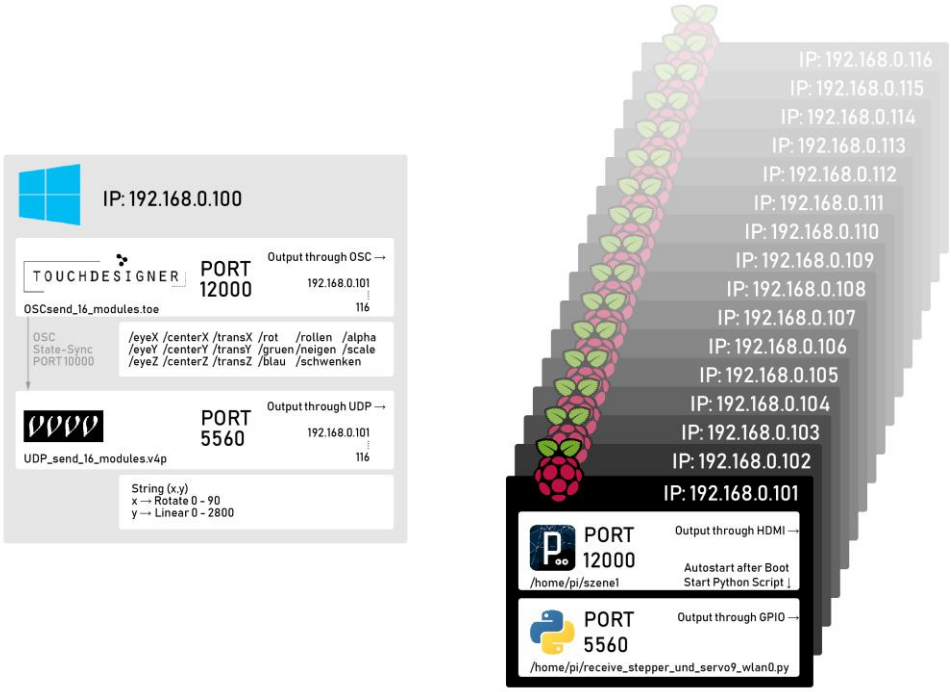


Abbildung 58 – Signalverarbeitung Installation

In dem Setup haben alle Geräte eine fixe IP-Adresse. Die Steuereinheit startet mit 100 am Ende. Die 16 Raspberrys haben die IPs von 101 bis 116. Die Clients 1 bis 10 werden mit Raspberry Pi Zero Ws umgesetzt, die Clients von 11 bis 16 mit Raspberry Pi 3 Model B. Dies hat mit der erforderlichen Rechenleistung zu tun. Die Abschnitte 6.3 und 6.4 gehen näher darauf ein.

Da durch die Verwendung von so vielen Geräten die Komplexität des Handlings steigt, werden einige Tools notwendig, die den Umgang mit diesen einfacher machen. Der nächste Abschnitt beschäftigt sich mit diesen Tools.

5.2 Tools

5.2.1 MobaXterm

Um den Umgang mit den Raspberrys zu erleichtern, wurde das Tool MobaXterm[20] verwendet. Mit diesem ist es möglich, eine Secure Shell (SSH) Verbindung zwischen dem Laptop und den Raspberrys aufzubauen (Kofler u. a., 2015, S. 137).

[20] https://mobaxterm.mobatek.net/

Dadurch erspart man sich das ständige Umstecken von Maus und Tastatur beziehungsweise die Notwendigkeit von 16 Mäusen und Tastaturen. So können Änderungen direkt vom Laptop aus mittels Kommandozeile gemacht werden.

Für den Aufbau einer Verbindung, muss bei dem verwendeten Raspberry in raspi-config SSH aktiviert sein. Danach kann man am Laptop in MobaXterm eine neue Session starten. Wie in Abbildung 59 zu sehen, muss oben links SSH ausgewählt werden und lediglich die IP-Adresse des Pis eingegeben werden. Nach der Eingabe des Standardbenutzers „pi“ und des Standardpasswortes „raspberry“ wird die Verbindung aufgebaut.

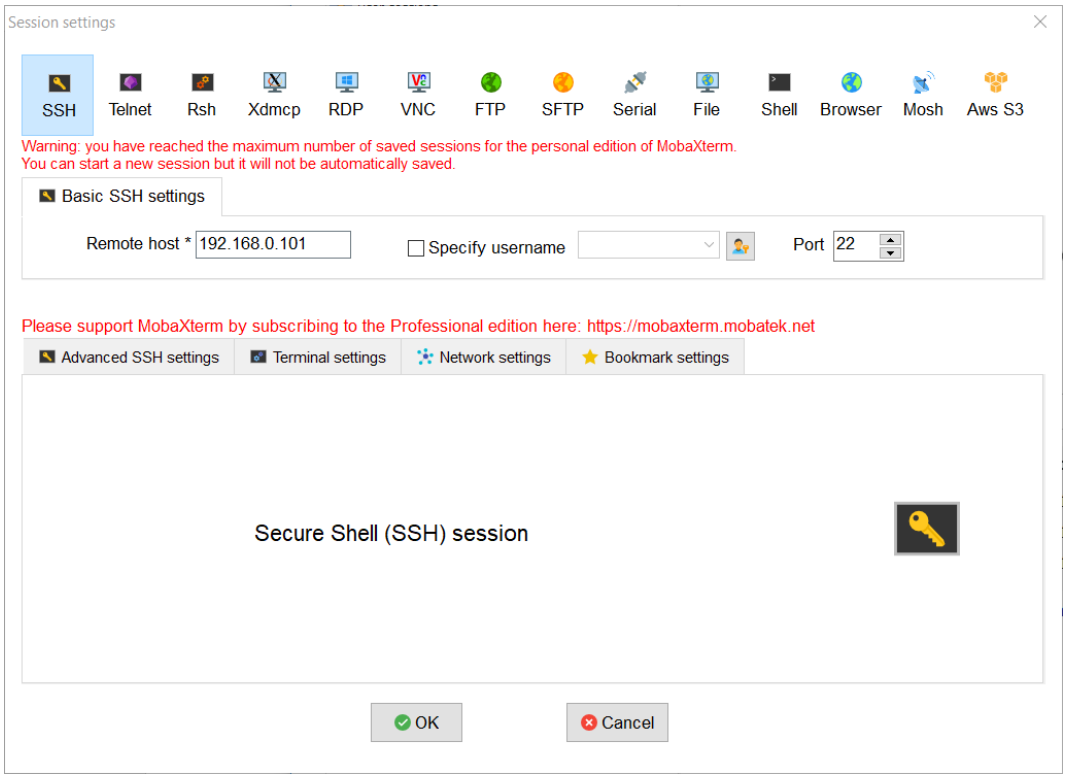


Abbildung 59 – MobaXterm Verbindungsaufbau

Praktisch dabei ist die automatisch aufgebaute SFTP-Verbindung. Über sie können direkt Dateien zwischen Laptop und Raspberry transferiert werden. Eine weitere wichtige Funktion ist MultiExec. Damit können, wie in Abbildung 60 zu sehen, Kommandozeilenbefehle auf allen Verbindungen gleichzeitig eingegeben werden. Sollen beispielsweise alle Raspberrys neu gestartet werden, muss der Befehl „sudo reboot“ nur einmal statt sechzehnmal eingegeben werden.

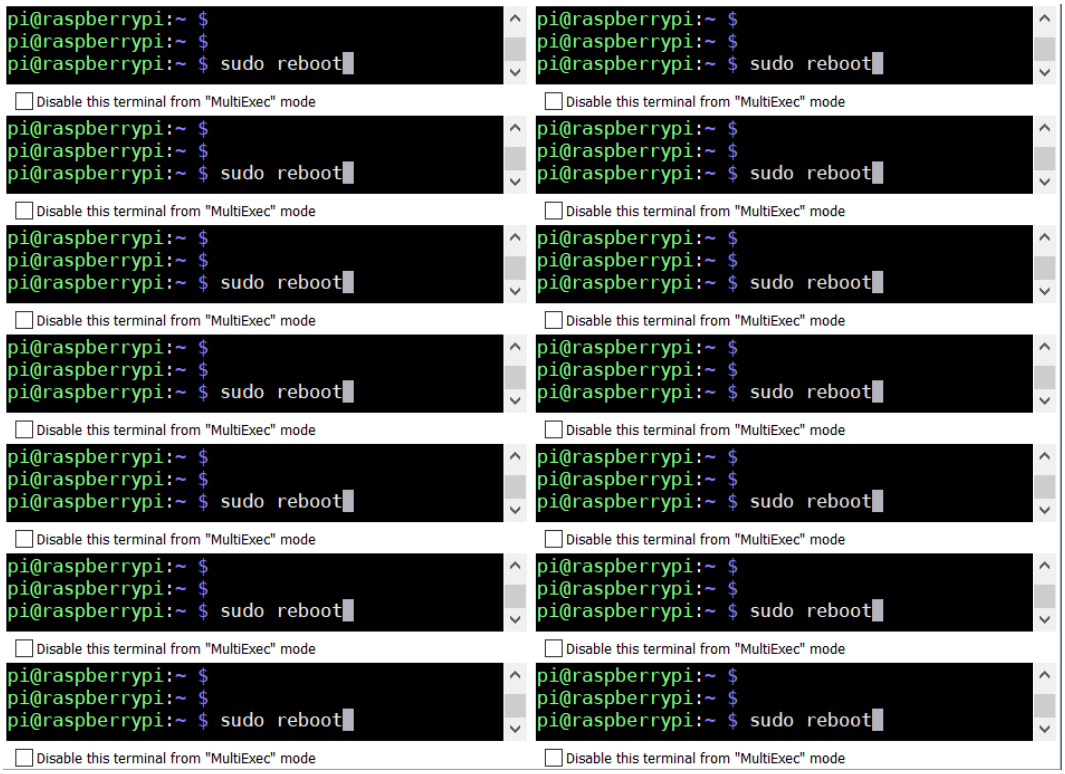


Abbildung 60 – MultiExec MobaXterm

5.2.2 UploadToPi

Upload to Pi[21] ist ein Tool für Processing. Während man am Laptop in Processing an einem Sketch arbeitet, kann man, wie in Abbildung 61 zu sehen, mit einem Klick über Tools → Upload to Pi den Sketch über Netzwerk auf den Raspberry laden. Dabei werden alle notwendigen Files hochgeladen und der Sketch gestartet. In den Preferences von Upload to Pi kann man die IP-Adresse auswählen, mit der die Verbindung hergestellt werden soll. Auf dem Raspberry wird der Sketch so eingerichtet, dass er nach dem Booten automatisch gestartet wird. Dies ist für diese Arbeit sehr praktisch. Nachdem alle Raspberrys mit Strom versorgt sind, fahren sie alle eigenständig hoch und starten den Sketch, der wiederum das Python-Skript startet. Dadurch ist keine Eingabe für das Starten der Installation notwendig.

[21] https://github.com/gohai/processing-uploadtopi

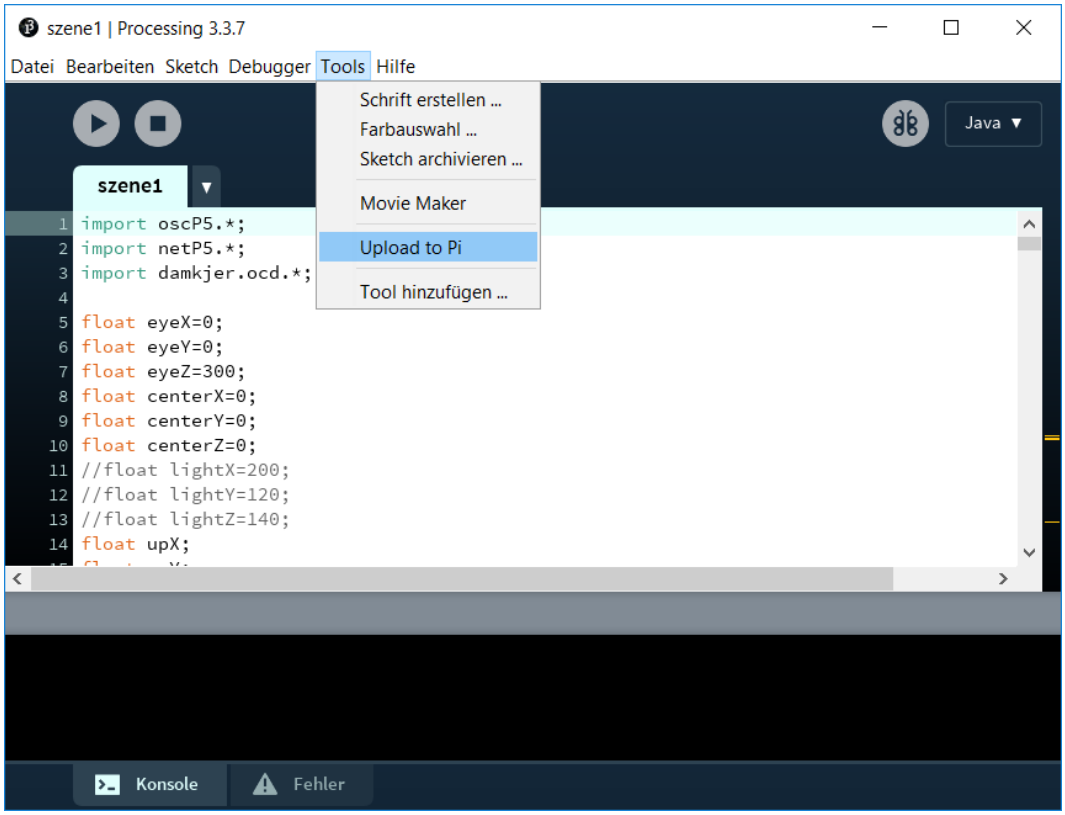


Abbildung 61 – Upload to Pi

Da Upload to Pi nicht mehrere IP-Adressen unterstützt, ist für die Verwendung von 16 Raspberrys eine andere Herangehensweise notwendig. Nachdem der Sketch an einem Raspberry angepasst worden war, wurde der Inhalt des erstellten Ordners „szene1“, in Abbildung 62 links zu sehen, mit MobaXterm über SFTP auf alle Raspberrys übertragen. Dieser Arbeitsschritt ist etwas mühsam, was dadurch den Workflow, für das Erstellen von Content, verkompliziert.

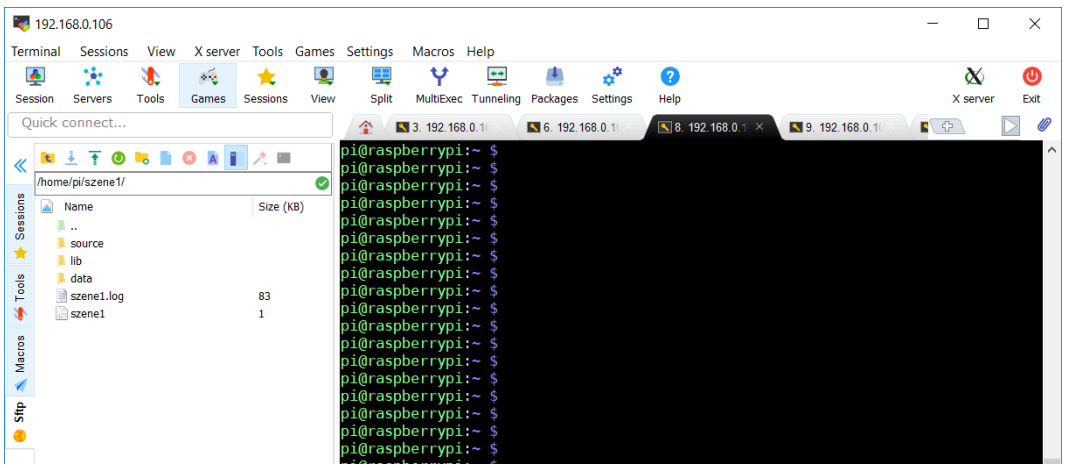


Abbildung 62 – SFTP – Vervielfältigung Processing Sketch

5.2.3 Imagesicherung

Um die mehrfache Ausführung von gleichen Arbeitsschritten beim Umgang mit den Raspberrys zu verhindern, wurden des Öfteren Images der Speicherkarten angefertigt. So musste beispielsweise die richtige Auflösung nur einmal anstatt sechzehnmal eingestellt werden. Dafür wurde das Programm Win32DiskImager[22] verwendet. Nach jedem großen Arbeitsschritt wurde ein Sicherheitsimage angefertigt.

Für die Einrichtung der gesamten Installation wurden die neuen microSD-Karten mit dem aktuellen Image bespielt. Danach wurde die Karte in einen Raspberry eingelegt und hochgefahren. Nach Änderung der IP-Adresse war die jeweilige Karte fertig für den Betrieb. Dadurch konnten unnötige, mehrfache Arbeitsschritte verhindert werden.

[22] https://sourceforge.net/projects/win32diskimager/

6 Hardware

6.1 Displayhalterungen

Für die Umsetzung der finalen Installation wurden Displayhalterungen und Verbindungselemente entwickelt und produziert. Aufgrund der vielfachen Stückzahlen und der Anforderung auf Genauigkeit wurden diese mit Hilfe eines 3D-Druckers gefertigt. Für die Anforderungen und die Entwicklung der einzelnen Elemente konnte auf die Erfahrungen der in Kapitel 4 angeführten Prototypen zurückgegriffen werden. Außerdem stand Christoph Braun bei allen möglichen Fragen rund um den 3D-Druck zur Verfügung. Vielen Dank an dieser Stelle. Die beiden Pfeile in Abbildung 63 veranschaulichen die Position und notwendige Funktion der Displayhalterung und des Verbindungselements.

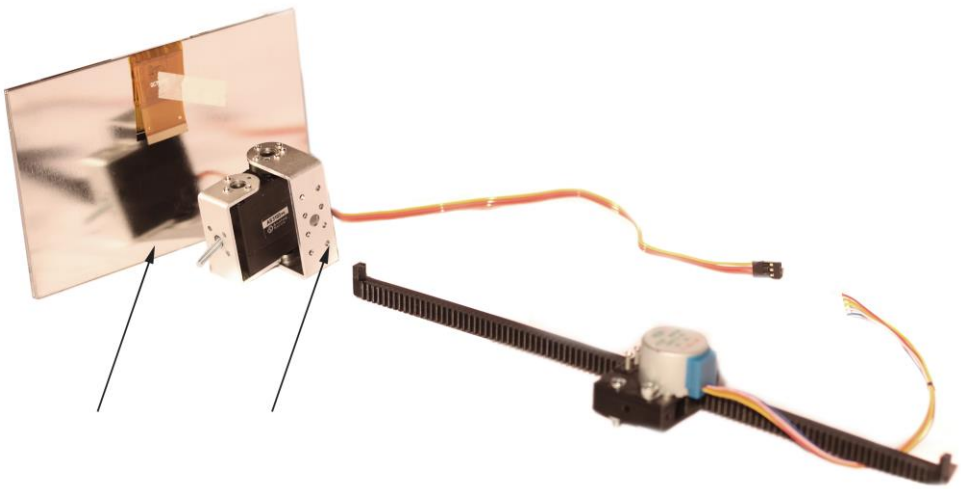


Abbildung 63 – Position Displayhalterung und Verbindungselement

Folgende Anforderungen bestehen an den zu entwickelnden Elementen:

Schnelle Druckzeit

Es sollen insgesamt 20 Displayhalterungen und Verbindungselemente produziert werden. Gerade deshalb ist es wichtig den Zeitaufwand jedes Elements so gering wie möglich zu halten, um eine vernünftige Produktionszeit zu gewährleisten.

Möglichst wenig Materialverbrauch

Ressourcen sind kostbar. Dadurch ist es notwendig nur so viel Material zu verwenden wie unbedingt notwendig ist. Außerdem ist diese Anforderung sehr stark mit der Druckzeit und dem Gewicht verbunden.

Leichtes Gewicht

Da die beweglichen Module von Servos und Schrittmotoren betrieben werden, spielt jedes Gramm eine Rolle. Ein leichteres Gehäuse resultiert in geringerem Drehmoment, dadurch besserer Performance und weniger Stromverbrauch der gesamten Installation.

Ausreichende Stabilität

Trotz der Notwendigkeit des leichten Gewichts, ist es wichtig ausreichende Stabilität zu gewährleisten. Die Displays sollen ausreichend geschützt werden, um eine lange Lebensdauer der gesamten Installation garantiert zu können.

Design passend zur gesamten Installation

Die Möglichkeit zur aktiven Gestaltung der Wirkung der Installation soll auch in diesem Entwicklungsprozess wahrgenommen werden.

6.1.1.1 CAD-Software

Alle Elemente wurden mit der freien Software OpenSCAD[23] entwickelt. Dieser Modeller ist eine Art 3D-Compiler der aus Skripten, in denen Parameter von Objekten definiert sind, in 3D Objekte rendert. Dadurch kann in den gesamten Modellierprozess eingegriffen werden (Kintel, 2018a). Folgendes Beispiel soll einen Einblick in diesen Prozess geben.

In diesem Code werden drei Cubes erzeugt. Mit dem Befehl „translate“ werden diese im Raum positioniert und mit „color“ eingefärbt.

```
color([1,0,0]) cube([2,3,4]);
translate([3,0,0])
color([0,1,0]) cube([2,3,4]);
translate([6,0,0])
color([0,0,1]) cube([2,3,4]);
```

Daraus errechnet OpenSCAD folgende Objekte.

[23] http://www.openscad.org/

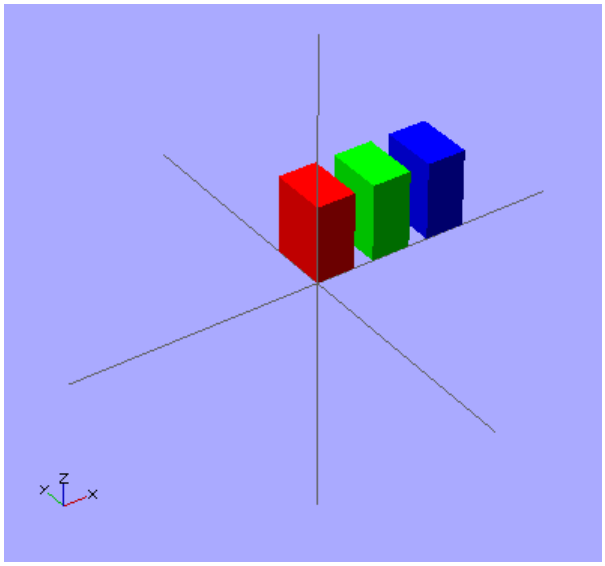


Abbildung 64 – Render-Beispiel OpenSCAD (Kintel, 2018b)

Die mit OpenSCAD entwickelten Objekte können nach Fertigstellung in den üblichen Austauschformaten, wie STL, OFF, AMF, DXF, SVG, CSG, exportiert werden. Die in dieser Arbeit entwickelten Objekte werden dem 3D-Drucker im STL-Format übergeben.

6.1.1.2 3D-Drucker

Die Elemente wurden alle auf dem, von der FH Sankt Pölten zur Verfügung gestellten, 3D-Drucker gedruckt. Dankeschön an dieser Stelle. Es handelt sich hierbei um das Model Witbox2[24], auf dem 1,75mm starkes Filament verarbeitet werden kann. Alle Teile dieser Arbeit wurden mit PLA-Filament gedruckt. Mit diesem Gerät können Teile mit maximal 297 x 210 x 200 mm produziert werden (Mundo Reader S.L, 2018). Abbildung 65 zeigt dieses Modell.

[24] https://www.bq.com/de/witbox-2

Abbildung 65 – 3D-Drucker Witbox 2 (Mundo Reader S.L, 2018)

Das aus OpenSCAD exportierte STL-Objekt, wird für die Verarbeitung im 3D-Drucker zuvor in die Ultimaker Cura software[25] geladen. Darin können die importierten Objekte auf der Grundfläche des Druckers positioniert, gedreht und skaliert werden. Des Weiteren können an dieser Stelle detailliertere Einstellungen für Material und Druckqualität getroffen werden. Für diese Arbeit wurde das für diesen Drucker optimierte Profil „fh_witbox2_profile“ mit 0,2mm Auflösung verwendet. Abbildung 66 zeigt das Interface dieser Software. Sie gibt dem User Auskunft über die Größe, die Druckdauer, die Länge des benötigten Filaments und über das Gewicht des fertigen Objekts. Diese Informationen sind für die Entwicklung sehr hilfreich, da noch vor dem Druck das Objekt optimiert werden kann. Zudem kann hier noch die Art der Füllung definiert werden und gegebenenfalls die Stützkonstruktion für überhängende Teile aktiviert werden. Zur Kontrolle können die einzelnen Schichten begutachtet werden, um danach das Objekt als g-code Druckdatei dem Drucker zu übermitteln (Ultimaker, 2018).

[25] https://ultimaker.com/en/products/ultimaker-cura-software

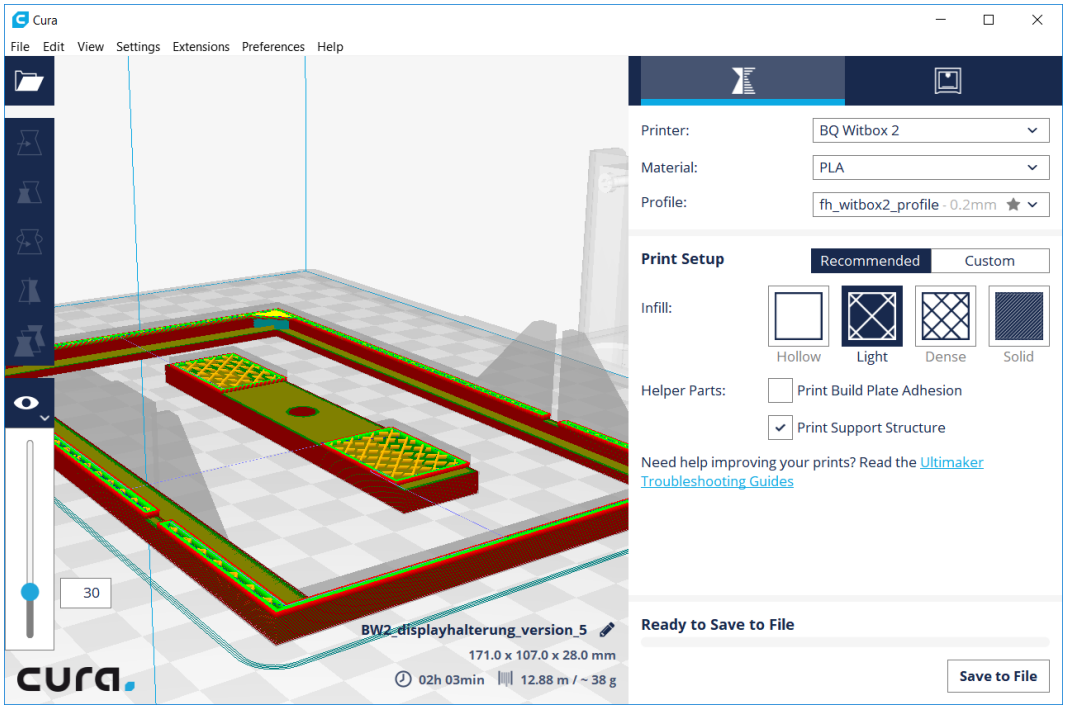


Abbildung 66 – Ultimaker Cura software (Ultimaker, 2018)

6.1.1.3 Displayhalterung

Als Basis für die Entwicklung der Displayhalterung dient der in Abschnitt 4.9 beschriebene Prototyp 2. Dieser wurde aus Metallteilen von Meccano[26] Spielzeugen zusammengeschraubt. Wie in Abbildung 67 und Abbildung 68 zu erkennen ist, liegt der Drehpunkt des Servomotors sehr nahe an der Displayfläche. Dies soll bei den 3D gedruckten Displayhalterungen beibehalten werden, da dies eine Drehung mit geringstem Drehmoment ermöglicht. Außerdem sollen bei den gedruckten Displayhalterungen die Drehachsen mittig angeordnet sein um eine symmetrische Drehung zu gewährleisten.

[26] http://www.meccano.com/products

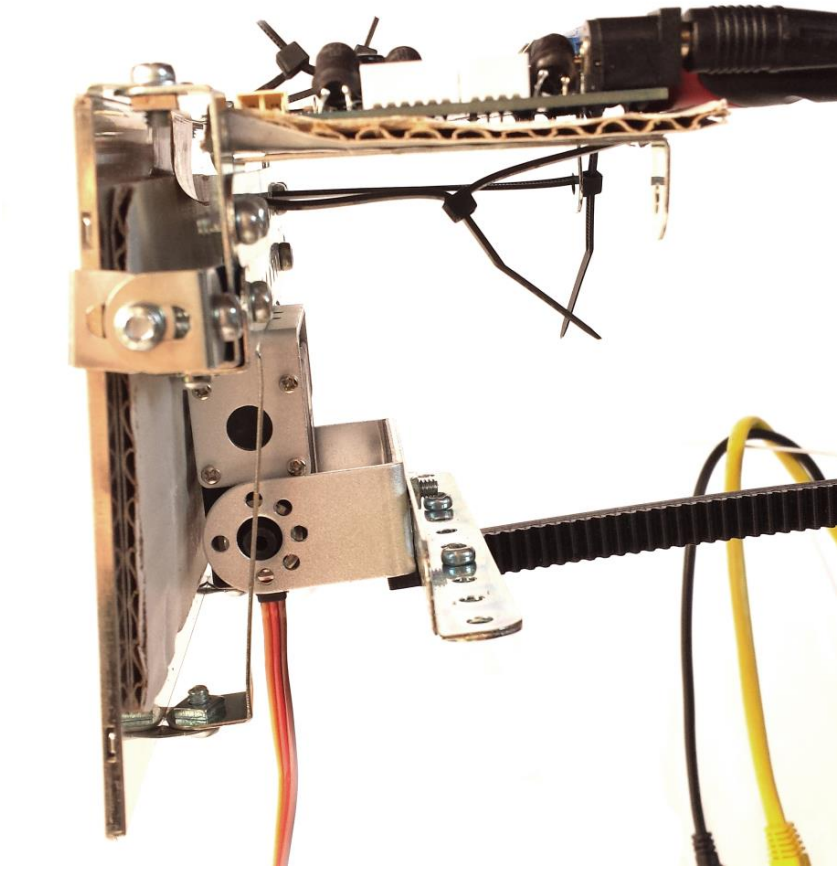


Abbildung 67 – Prototyp 2 Seitenansicht

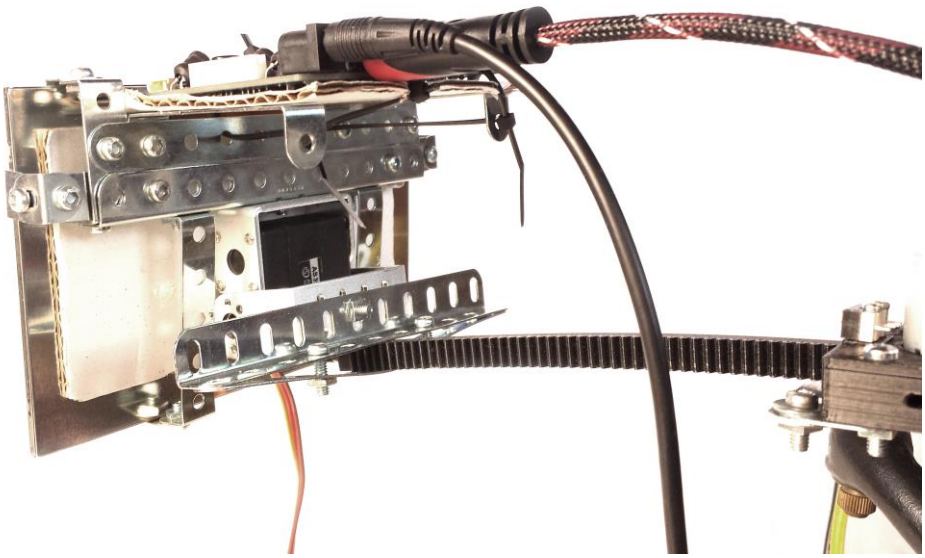


Abbildung 68 – Prototyp 2 Rückansicht

Beim Prototyp 2 wurde das Display geneigt. Es erfolgte also eine Drehung um die horizontale Achse des Displays. Da wie in 4.9 besprochen, bei den ersten Tests das Problem des Betrachtungswinkels aufgekommen ist, soll hier eine Änderung vorgenommen werden. Die Angaben des Herstellers bezüglich Betrachtungswinkel sind 70° links und rechts, 50° oben und 60° unten (52Pi, 2018). Da der horizontale Betrachtungswinkel größerer als der Vertikale ist, soll das Display geschwenkt statt geneigt werden, um bei der Bewegung der Displays einen homogeneren Farb- und Helligkeitseindruck zu bekommen. Das bedeutet, dass der Servo um 90° gedreht werden muss.

Beim Prototyp 2 ist im oberen Bereich die Displaytreiberplatine angebracht. Diese soll in der finalen Version des Moduls nach hinten verlegt werden, um die vom Servo zu drehende Masse geringer zu halten und so die Performance zu erhöhen. Mit einer Flachbandkabelverlängerung werden die Platine und das Display anschließend verbunden.

Die Displayhalterung wurde in drei Druckzyklen entwickelt. Alle scad-Codes sind im Anhang unter A. Hardware ersichtlich. Das Rendering und das fertige Objekt der ersten gedruckten Displayhalterung „displayhalterung_version_3.scad“ sind in Abbildung 69 und Abbildung 70 zu sehen.

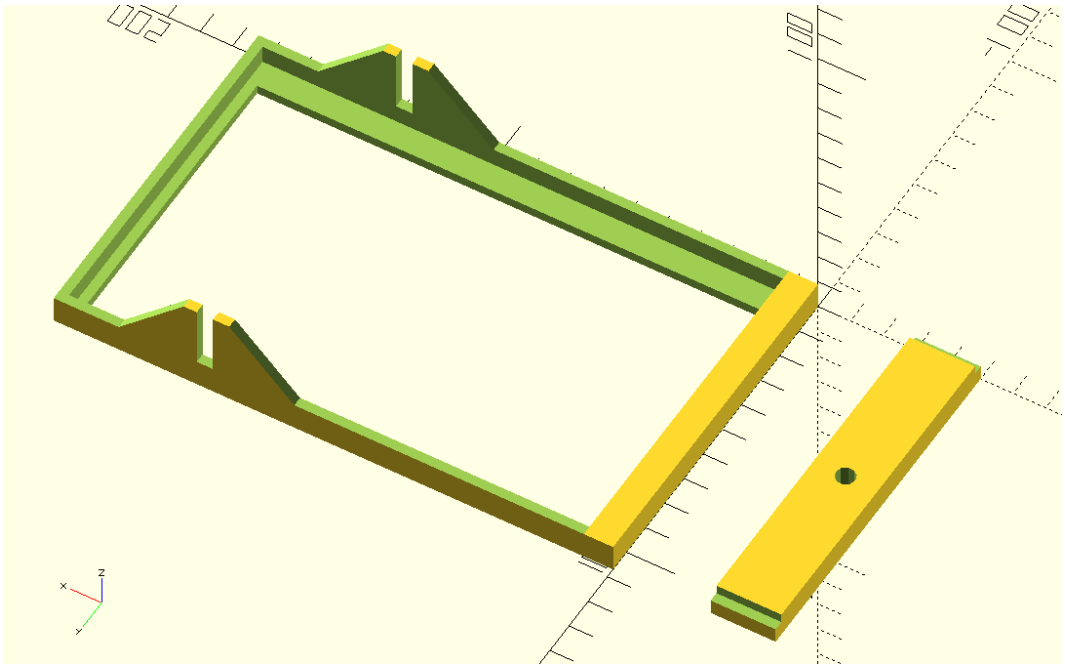


Abbildung 69 – Rendering displayhalterung_version_3

Abbildung 70 – 3D-Druck displayhalterung_version_3

Die Idee hinter dieser Version war es, dass das Display auf einer Seite von der Nut fixiert wird und an der anderen Seite von dem Quereinschub niedergehalten wird. Das Gehäuse wurde mit einer 3mm Wandstärke konstruiert. Das Objekt konnte ohne Schwierigkeiten gedruckt werden. Die auf der rechten Seite in den obigen Abbildungen zu sehende Nut erforderte eine Stützkonstruktion, da der Oberteil überhängend war. Diese musste danach entfernt werden. Das Wegschneiden mit Messer stellte sich als aufwendig heraus. Mit einem Lötkolben konnte die Stützkonstruktion schließlich entfernt und das Display eingesetzt werden. Für die Umsetzung aller Elemente war diese Version also nicht geeignet, da das Entfernen der Stützkonstruktion einen zu hohen zeitlichen Aufwand darstellen würde. Nach dem Zusammenbau wurde die fehlende Befestigungsmöglichkeit für das Verbindungsstück der Flachbandkabel bemerkt. Sonst war dieser Druck sehr zufriedenstellend.

Das Problem mit dem Entfernen der Stützkonstruktion und mit der Befestigungsmöglichkeit wurden in der folgenden Version berücksichtigt. Außerdem wurde die Wandstärke auf 2mm reduziert um Material einzusparen und um die Druckzeit zu minimieren. Des Weiteren wurde eine Vertiefung auf dem Quereinschub konstruiert, um einerseits die Befestigung des Servos zu verbessern und andererseits um wieder Material und Zeit zu sparen. In Abbildung 71 ist das Objekt von „displayhalterung_version_4.scad“ zu sehen. Der scad-Code ist wieder im Anhang zu finden.

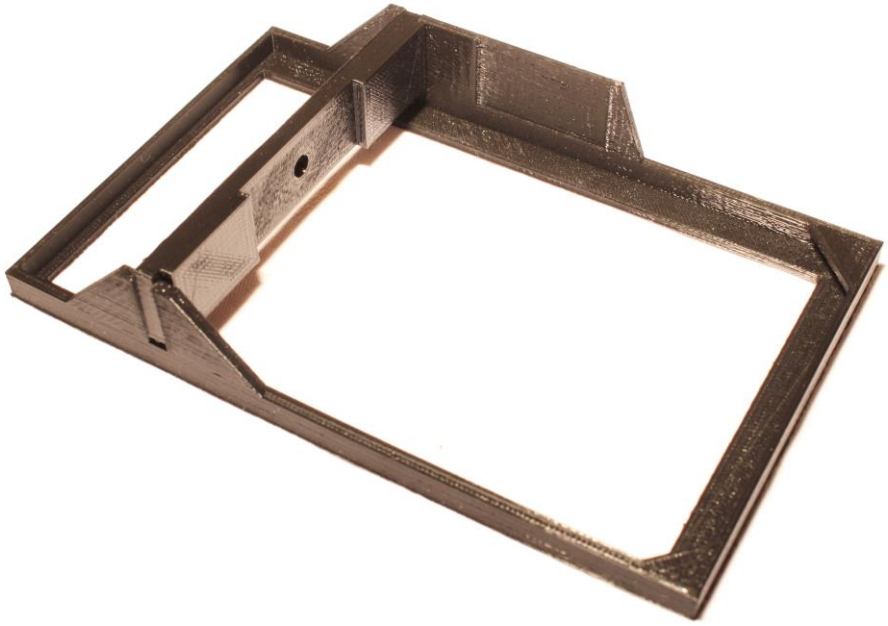


Abbildung 71 – 3D-Druck displayhalterung_version_4

Wie in der obigen Abbildung rechts zu sehen, wurde hier die Nut nicht über die gesamte Breite ausgeführt, sondern nur an den Ecken im 45° Winkel angebracht. Dadurch wurde der Zugang zum Entfernen der Stützkonstruktion enorm erleichtert und der Aufwand verringert. Für das Verbindungsstück des Flachbandkabels wurde an der Oberseite eine Fläche mit passender Vertiefung konstruiert, an der es leicht festgeklebt werden kann. Diese Version war schon sehr zufriedenstellend. Allerdings gab es leichte Passungenauigkeiten zwischen Display und Halterung.

Im letzten Entwicklungszyklus wurden die Maße für das Display gering angepasst, um die zuvor besprochenen Passungenauigkeiten zu verringern. Außerdem wurde der Schlitz für den Quereinschub ein wenig herabgesetzt um die Passgenauigkeit zu erhöhen. Abbildung 72 zeigt die finale Displayhalterung „displayhalterung_version_5.scad“ mit dem Verbindungsstück des Flachbandkabels. Diesmal wurde sie in weiß gedruckt um einen Eindruck über die Wirkung unterschiedlicher Farben zu bekommen. Der scad-Code ist wieder im Anhang zu finden. Dieser kann nun für die Produktion der restlichen Displayhalterungen verwendet werden.

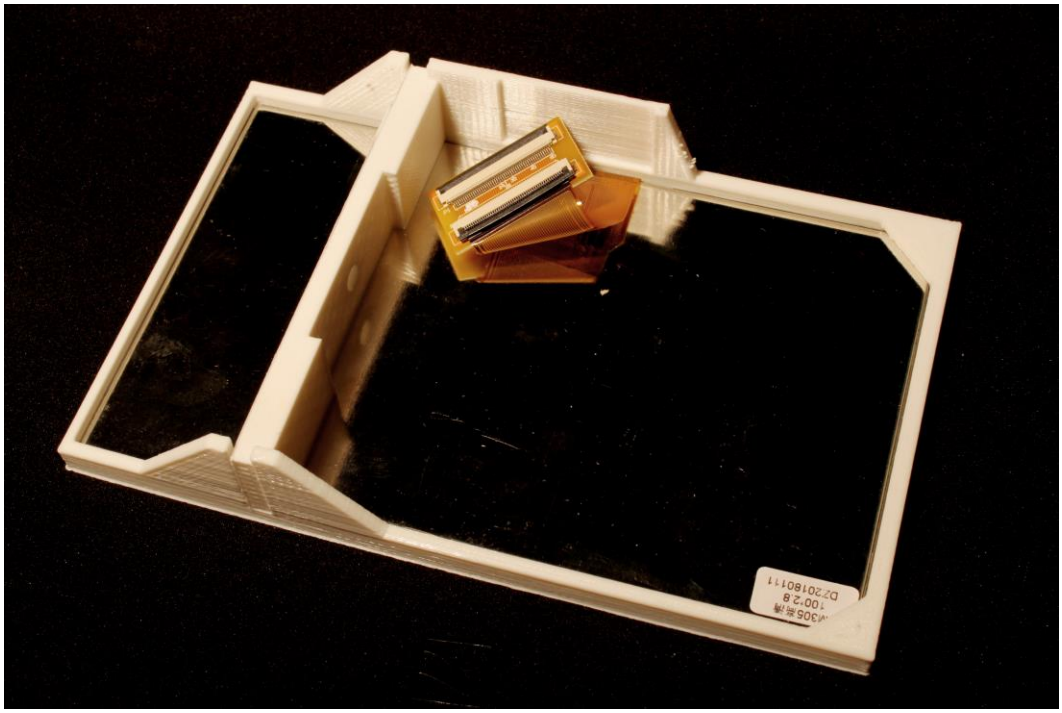


Abbildung 72 – 3D-Druck displayhalterung_version_5

6.1.1.4 Verbindungselement

Das Verbindungselement fixiert den flachen Metallbügel des Servos an einem Ende des Linear-Aktuators. Dieses Verbindungselement wurde in drei Entwicklungszyklen entworfen. Es besteht aus zwei Einzelteilen, die zusammengesetzt werden können. Abbildung 73 zeigt es auf der linken Seite aufgeklappt. Dies entspricht der Druckposition. Auf der rechten Seite sind die beiden Teile zusammengesteckt. Durch die quadratische Öffnung an der Oberseite kann die Zahnstange des Linear-Aktuators gesteckt werden. Mit einer Schraube kann das Verbindungselement an den Metallbügel des Servos fixiert werden. Durch die schräge Verbindung verklemmt sich die Zahnstange darin.

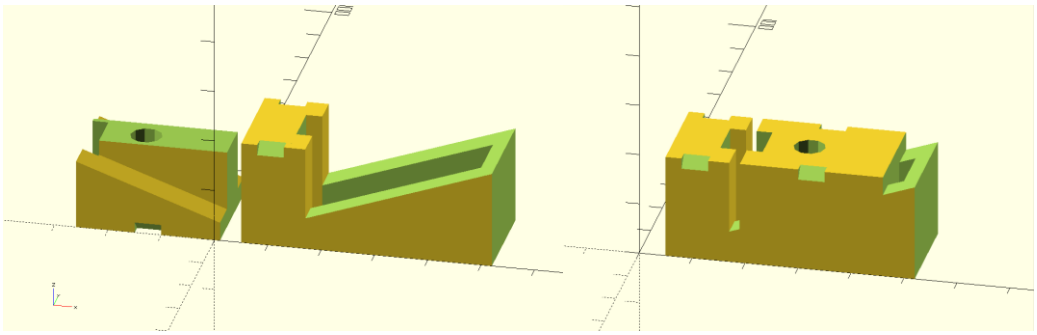


Abbildung 73 – Verbindungselement linear_servo_connector_version_3

Abbildung 74 zeigt das montierte Verbindungselement. Dadurch können die beiden Teile mit nur einer Schraube verbunden werden.

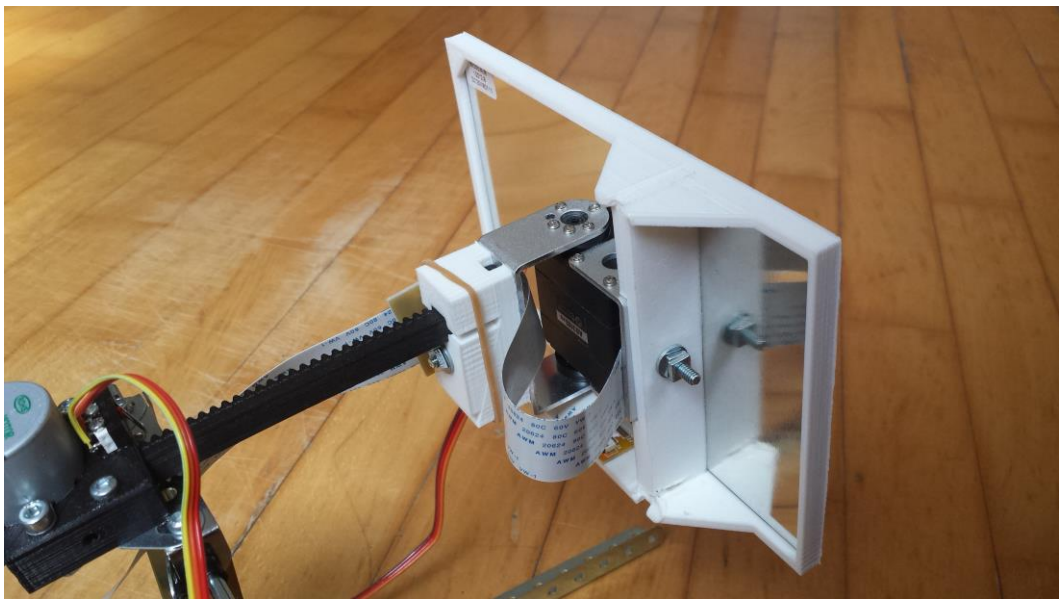


Abbildung 74 – Verbindungselement montiert

6.2 Der Zapfen

In der achten Engagement Session in Abschnitt 4.10 wurde intensiv über die Form und Anordnung der Installation nachgedacht und diskutiert. Darin wurde sich auf eine nicht planare Form geeinigt, da sich die im Raum versetzten Anordnungen der Module als imposanter herausstellten. Für den Bau der Installation dient die Form eines Zapfens als Vorlage. Abbildung 75 zeigt diesen von vorne. Dabei entsprechen die seitlich hervorstehenden Blätter den Displays. Diese Anordnung hat den Vorteil, dass die Displays in der Tiefe gestaffelt sind und dass alle ins Zentrum des Zapfens geneigt werden können. So können sich alle auf den Benutzer der Installation beziehen.

Abbildung 75 – Der Zapfen als Vorlage

Wie in der natürlichen Vorlage, besteht auch die Grundkonstruktion der Installation aus einem geraden Mittelstück. Dabei wurde, wie in Abbildung 76 ersichtlich, ein Leimholz aus Fichte mit etwa 12x12x100cm verwendet. Von diesem wurden die 4 länglichen Ecken durch Längsschnitt entfernt, sodass sich im Querschnitt die Form eines Achtecks ergab. Anschließend wurden auf allen acht Seiten Längsschnitte gemacht. Dadurch ergaben sich vier durchgängige Schlitze, in denen genau die Bretter der Module passten. In Abbildung 76 ist ein Brett zur Demonstration durchgeschoben.

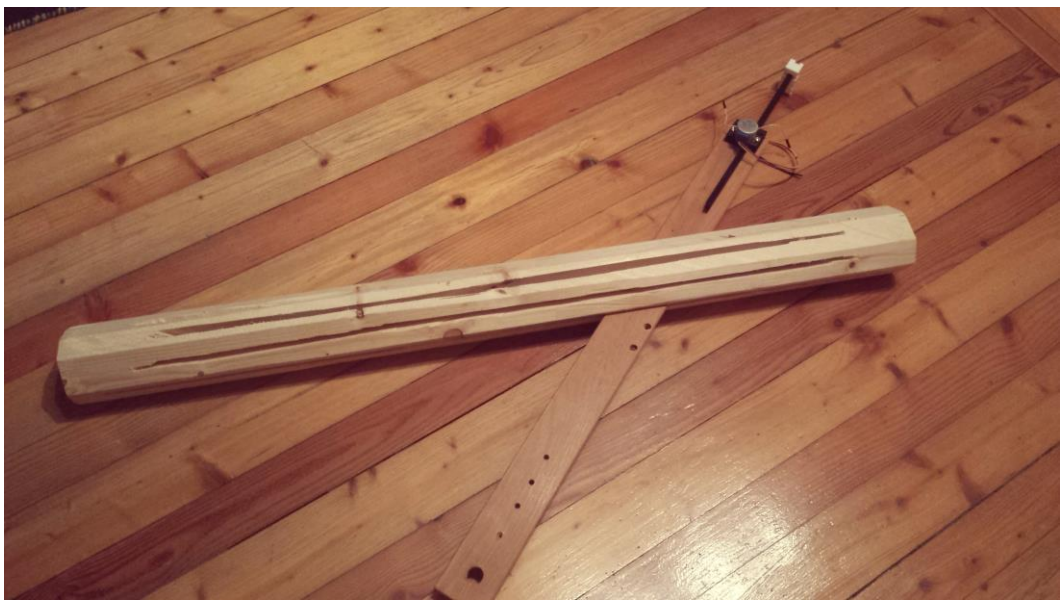


Abbildung 76 – Basis Konstruktion

Dieser Mittelteil des Zapfens wurde in ein Fahrradmontagestativ eingespannt. Dadurch können die Position und Rotation des Holzes variiert werden. Die einzelnen Module können mit dieser Konstruktion von acht verschiedenen Seiten in den Schlitzen montiert werden. So ist es möglich die Form des Zapfens variabel nachzuahmen. In Abbildung 77 sind zwei Blätter montiert.

Abbildung 77 – Zusammenbau der Installation

Die Bretter, auf denen die einzelnen Module befestigt sind, wurden am anderen Ende zweimal eingeschnitten. Dadurch konnten die Module nach dem Einführen ineinandergeschoben werden. So war es möglich die Module in nur geringer Tiefe anzuordnen. Abbildung 78 und Abbildung 79 zeigen die fertiggestellte Installation von der Seite und von vorne. Wenn man diese nochmal mit dem Vorbild, dem Zapfen vergleicht ergeben sich gewisse Ähnlichkeiten. Die Installation ist weniger tief angeordnet. Sie erinnert mehr an eine Blume, als an einen Zapfen. An der Mittelkonstruktion verlaufen die Stromkabel nach links zu den PC-Netzteilen, die die Raspberrys mit Strom versorgen.

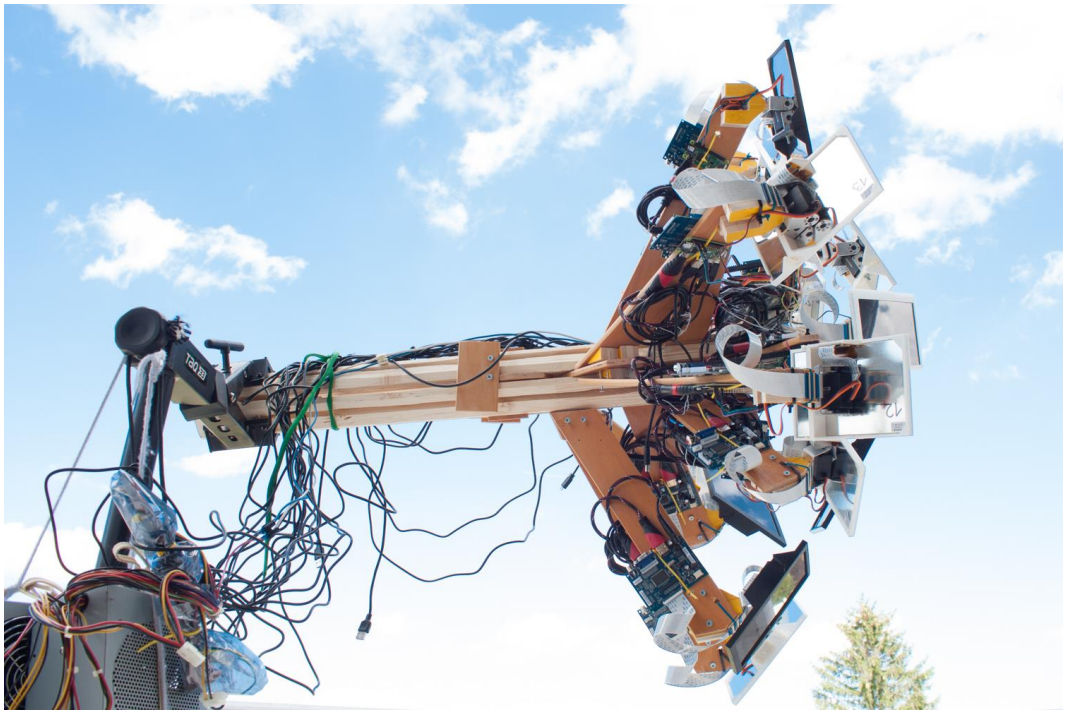


Abbildung 78 – Fertige Installation seitlich

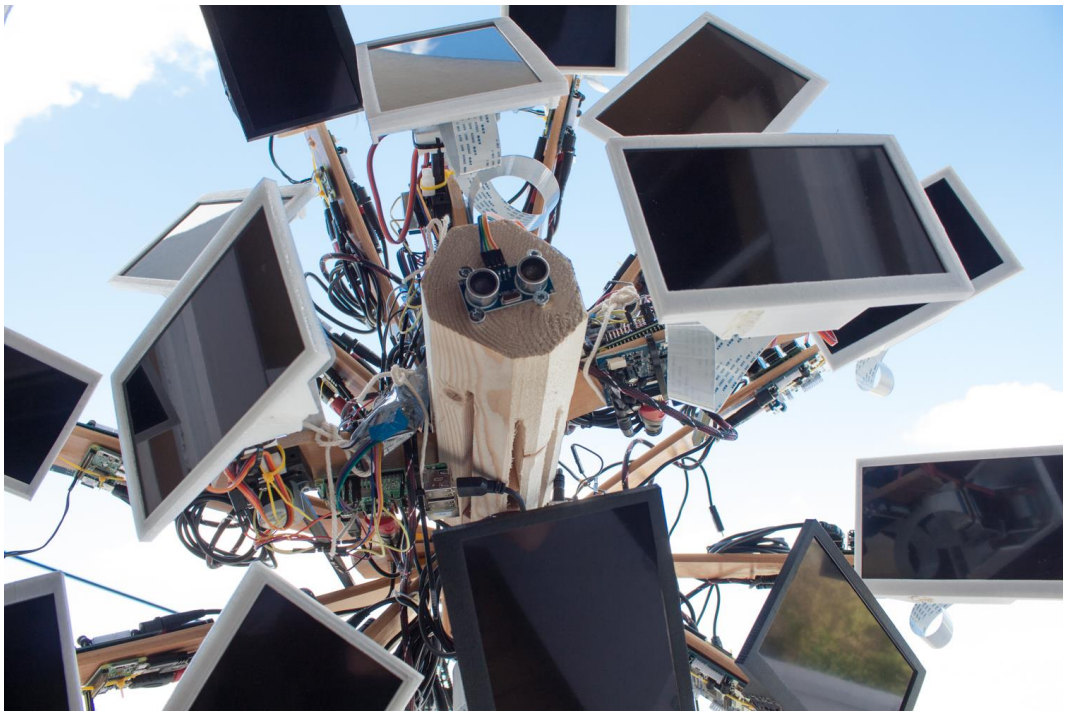


Abbildung 79 – Fertige Installation vorne

6.3 Modul 1

Das Modul 1 ermöglicht die lineare Bewegung mit Hilfe des Linear-Aktuators und die Rotation mittels Servo. Abbildung 80 zeigt die Verkabelung von Modul 1. Alle Elemente werden durch das PC-Netzteil mit Spannung versorgt. Abschnitt 6.5 geht näher darauf ein. Damit die Strombelastung am Raspberry nicht zu hoch ist, erfolgt die Spannungsversorgung für Servo und Schrittmotor gesondert. Über Pin 18 wird das PWM-Signal an den Servo geschickt. Über die Pins 4, 17, 23 und 24 werden die Steuersignale für den Schrittmotor an das ULN2003 Treiber-Board geleitet. Der Referenztaster des Schrittmotors gelangt über Pin 7 in das Raspberry. Die Displaytreiberplatine wird über den USB Ausgang des Raspberrys mit Strom versorgt und bekommt über HDMI die Bilddaten.

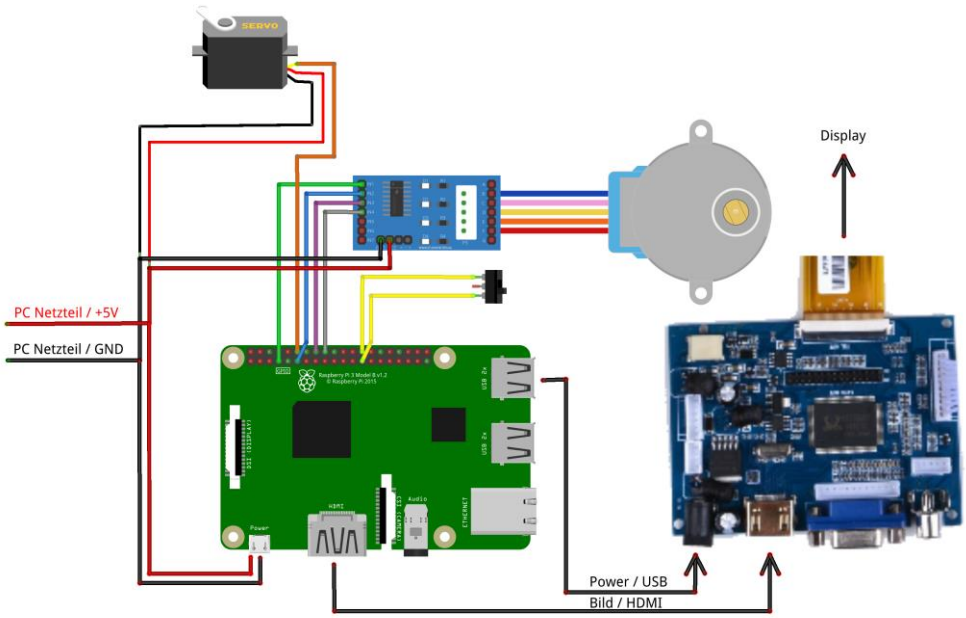


Abbildung 80 – Signalverarbeitung Modul 1

Für die Umsetzung von Modul 1 wird ein Raspberry Pi 3 Model B verwendet. Grund dafür ist die höhere Rechenleistung im Gegensatz zum Raspberry Pi Zero W. Der Raspberry Pi Zero W hat einen single-core Prozessor mit 1GHz, der Raspberry Pi 3 Model B hingegen einen quad-core Prozessor mit 1.2GHz. Der Raspberry Pi Zero W wurde ebenfalls für diese Aufgabe eingesetzt. Manchmal kam es aber aufgrund der begrenzten Rechenleistung zu verspäteten Reaktionen der Motoren oder zum Ausfallen einiger Frames.

Für die Fertigung von Modul 1 wurde die Basis des Linear-Aktuators etwa im rechten Winkel zum Brett auf diesen fixiert. Wichtig dabei ist der Freiraum, der für

die bewegliche Zahnstange notwendig ist. Abbildung 81 zeigt dieses Modul 1. Auf der Rückseite des Bretts ist die Displaytreiberplatine angebracht. Von dort aus gelangt das Videosignal über zwei Flachbandkabel und zwei Verbinder zum Display. Ein Verbinder ist an der Basis des Servos mit einem schwarzen Kabelbinder montiert. Von dort aus geht das zweite Flachbandkabel weiter zum zweiten Verbinder am Display. Das erste Flachbandkabel ist einmal eingerollt und dient zur Überbrückung zwischen Displaytreiberplatine und Servo. Dieser Abstand ist je nach Auszug variabel. Das zweite Flachbandkabel dient zur Überbrückung vom Servo zum Display. Es ist locker gebogen um ein Abknicken bei einer Rotation des Displays zu verhindern.

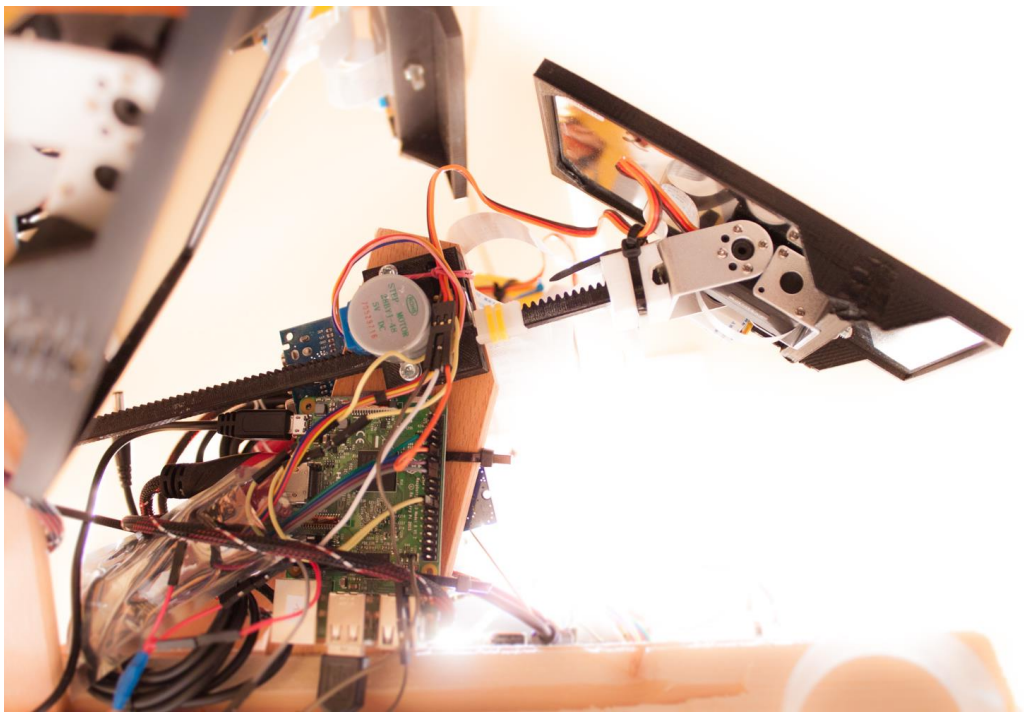


Abbildung 81 – Zusammenbau Modul 1

6.4 Modul 2

Im Gegensatz zum Modul 1 ist dieses Modul 2 ohne Linear-Aktuator ausgeführt. Dieses Modul ist in der Lage Content am Display anzuzeigen und dieses durch die Verwendung eines Servomotors zu neigen. Da dadurch die Stromaufnahme geringer als mit Schrittmotor ist, wird hier der Servomotor direkt am Raspberry angeschlossen und bezieht auch den Strom davon. Das PWM-Signal für die Steuerung des Servos gelangt wie beim vorherigen Modul über den Pin 18 zum

Servo. Der Positive Draht des Servos wird wie in Abbildung 82 ersichtlich auf einen +5V Pin gehängt und der Ground Draht des Servos auf Ground des Raspberrys. Der Rest ist analog zum Modul 1 ausgeführt. Der Raspberry wird über das PC-Netzteil mit Strom versorgt. Die Displaytreiberplatine wird über USB vom Raspberry mit Strom versorgt und die beiden sind mit HDMI miteinander verbunden.

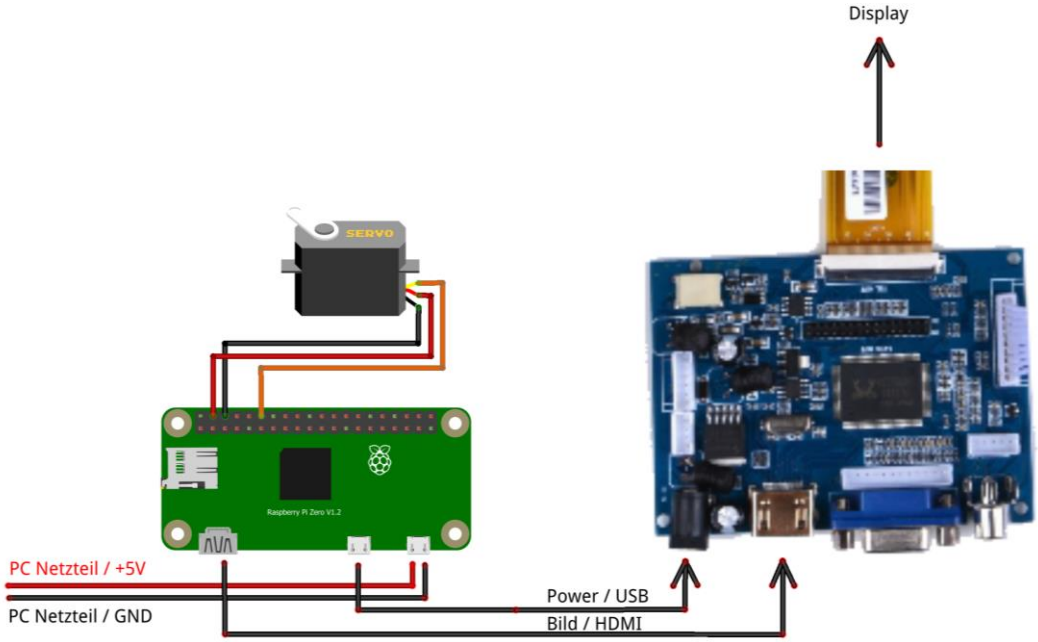


Abbildung 82 – Signalverarbeitung Modul 2

Das Modul 2 wird mit einem Raspberry Pi Zero W umgesetzt. Im Gegensatz zum Modul 1 ist hier der notwendige Rechenaufwand geringer. Dieses Modell ist wesentlich kleiner und kostet mit etwa 15€ pro Stück zirka die Hälfte des größeren eingesetzten Modells.

Abbildung 83 zeigt das Modul während dem Zusammenbau. Die Basis für das Modul bildet das Brett eines alten Lattenrostes. Dieses wird wie in Abschnitt 6.2 beschrieben, in den Schlitz des Hauptträgers eingeführt. Durch Ausschneiden einer Hakenform verkeilen sich die einzelnen Module ineinander und verhindern so deren Rausfallen. Mit einer zusätzlichen Holzkonstruktion wird das Display und der Servo an dem Brett fixiert. Auf der einen Seite ist mit Kabelbinder der Raspberry Pi Zero W fixiert. Über einen mini HDMI auf HDMI Adapter wird das Videosignal über das HMDI Kabel auf die andere Seite in die Displaytreiberplatine geleitet. Von dort aus gelangt das Videosignal über mehrere Flachbandkabel zum Display. Dabei ist, an der Stelle des hellblauen Kabelbinders, ein Verbindungsstück angebracht, welches zwei Flachbandkabel miteinander

verbindet und für die Zugentlastung eingesetzt wird. Danach führt ein locker gelegtes Flachbandkabel das Videosignal zum Display, sodass durch Neigung keine Knicke entstehen können. In der Abbildung sind der Servo und das Stromkabel fürs Display noch nicht angeschlossen.

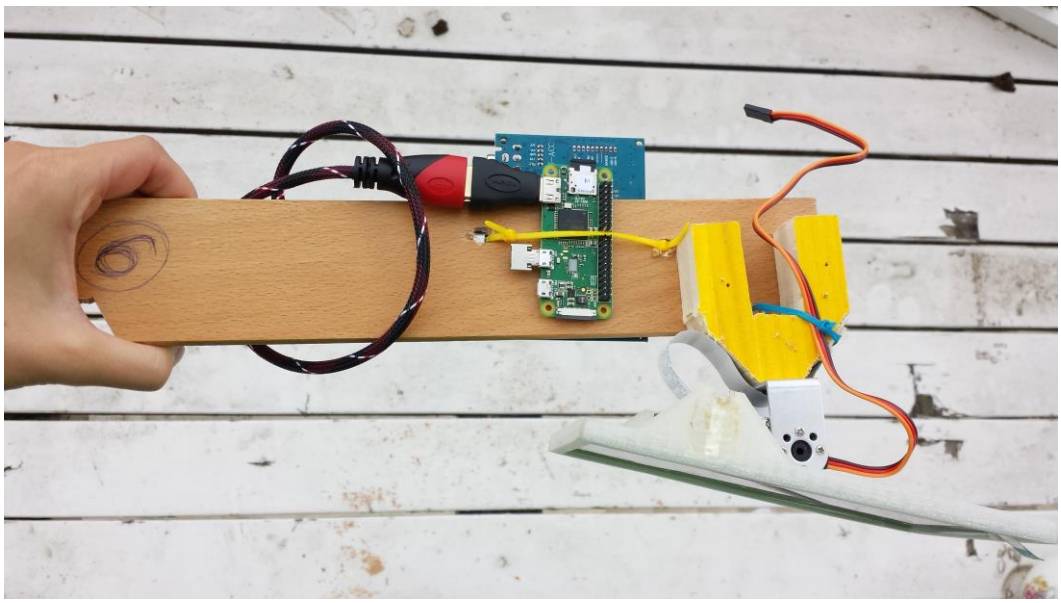


Abbildung 83 – Zusammenbau Modul 2

6.5 Stromversorgung

Die gesamte Stromversorgung der Installation, also der Raspberrys, der Displays und der Motoren läuft über zwei alte PC-Netzteile. Jedes PC-Netzteil liefert eine saubere Spannung von +5V, sie eignen sich daher perfekt für die Versorgung der Raspberrys. Abbildung 84 zeigt die beiden Netzteile inklusive der angelöteten microUSB-Kabeln. Ein Netzteil liefert laut Datenblatt 32A, das andere 25A. Die Stromversorgungen für die Raspberrys wurden gleichmäßig aufgeteilt. Jedes Netzteil versorgt also 8 Module. Beim schwächeren Netzteil stehen jedem Modul rechnerisch 3,125A zur Verfügung. Die Installation läuft mit den beiden Netzteilen größtenteils stabil. Bei schnellster Aussteuerung der Servos reicht die Spannung nicht mehr aus. In diesem Fall schaltet das Raspberry sicherheitshalber die USB Ausgänge ab. Dies ist dadurch bemerkbar, dass sich bei schnellster Bewegung ein bis zwei Displays wieder aufdrehen. Wird die Installation normal betrieben, bleiben die Displays stabil. Für eine hundertprozentige Stabilität wäre die Verwendung eines dritten PC-Netzteiles von Vorteil. Die Spannungsversorgungen der Displays könnten für noch mehr Stabilität direkt am PC-Netzteil angeschlossen werden, da dadurch die Raspberrys weiter entlastet werden würden.

Abbildung 84 – PC-Netzteile für Stromversorgung

Für die Stromversorgung der Raspberrys wurden 1,8m lange microUSB-Kabeln verwendet. Diese hatten am Ende bereits zwei abisolierte Litzen. Diese wurden mit Hilfe einer Lochrasterplatte, wie in Abbildung 85 zu sehen, mit der Stromversorgung des PC-Netzteiles zusammengelötet.

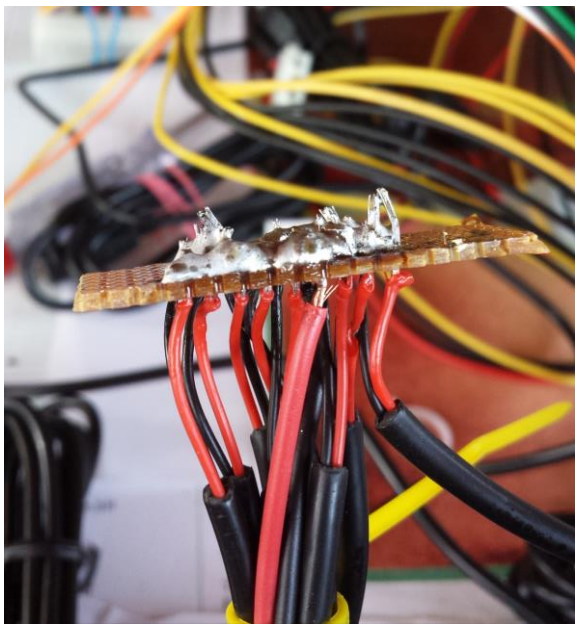


Abbildung 85 – Lötstelle Stromversorgung

6.6 Netzwerk

Das Herzstück der Kommunikation dieser Installation ist der Wireless Router. Für diese Arbeit wurde der, in Abbildung 86 gezeigte, „3HuiTube ZTE 4G Wireless Router“ verwendet. Laut Herstellerangaben sind mit diesem Gerät 32 Teilnehmer möglich (Henning, 2018). Alle 17 Geräte dieser Installation sind über WLan mit dem Router verbunden.

Abbildung 86 – Wireless Router

Dabei wurde jedem Gerät eine statische IP-Adresse zugewiesen. Über die IP-Adresse „192.168.0.1“ kann auf das Webinterface des Wireless Routers zugegriffen werden. Dabei kann man alle verbundenen Geräte kontrollieren. Abbildung 87 zeigt einen Screenshot dieses Interfaces. Alle Clients 16 sind hier verbunden, wobei die Steuereinheit die Adresse „192.168.0.100“ hat.

WLAN Geräte

Nummer	Host Name		MAC Adresse	IP Adresse	Adress-Typ	Anwenden
1	--		B8:27:EB:CD:D9:B5	--	IP zuweisen	Blockieren
2	--		B8:27:EB:F6:E0:61	--	IP zuweisen	Blockieren
3	raspberrypi		B8:27:EB:22:E8:DF	--	IP zuweisen	Blockieren
4	--		B8:27:EB:53:4B:B4	--	IP zuweisen	Blockieren
5	--		B8:27:EB:94:15:03	--	IP zuweisen	Blockieren
6	--		B8:27:EB:3A:63:49	--	IP zuweisen	Blockieren
7	--		B8:27:EB:23:85:66	--	IP zuweisen	Blockieren
8	--		B8:27:EB:7B:5B:7A	--	IP zuweisen	Blockieren
9	--		B8:27:EB:9F:3B:D2	--	IP zuweisen	Blockieren
10	--		B8:27:EB:4C:98:8B	--	IP zuweisen	Blockieren
11	--		B8:27:EB:F4:15:B9	--	IP zuweisen	Blockieren
12	--		B8:27:EB:E7:69:BF	--	IP zuweisen	Blockieren
13	--		B8:27:EB:3F:07:54	--	IP zuweisen	Blockieren
14	--		B8:27:EB:50:98:B3	--	IP zuweisen	Blockieren
15	--		B8:27:EB:30:A5:4B	--	IP zuweisen	Blockieren
16	--		B8:27:EB:17:20:A2	--	IP zuweisen	Blockieren
17	--		5A:00:74:DE:5F:7E	192.168.0.100	**DHCP**	Blockieren

Abbildung 87 – Webinterface Wireless Router

6.7 Sensorik

Für die Sensorik dieser Installation wurde ein Ultraschallsensor vom Modell „HC-SR04“ verwendet. Dieses ist mit einem Preis von nur ein paar Euros ein sehr kostengünstiger Sensor. Damit ist es möglich kontaktlos den Abstand von Objekten zu messen, indem der Sensor Tonimpulse mit einer Frequenz 40kHz sendet. Werden diese Impulse von einem Objekt reflektiert, gelangen diese wieder zum Sensor. Dadurch kann dieser die Zeitdifferenz zwischen senden und empfangen ermitteln. Mit folgender Formel kann von der Zeit auf die Distanz geschlossen werden (Elek, 2018).

Distanz [m] = (Zeitdifferenz [s] x Schallgeschwindigkeit [343m/s]) / 2

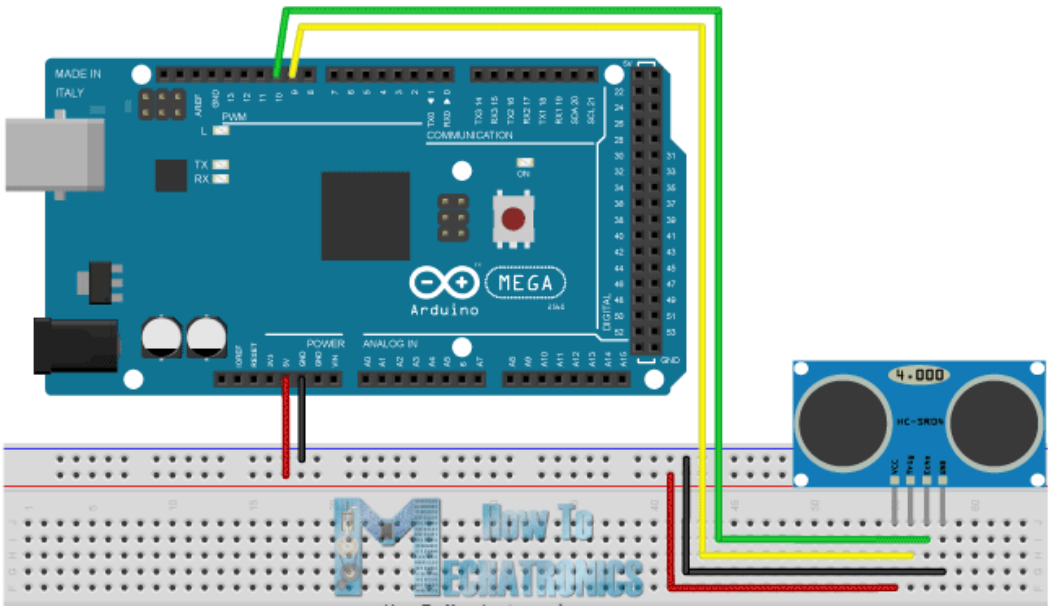


Abbildung 88 – Verkabelung Ultraschallsensor (Dejan, 2015)

Die Verkabelung und Programmierung wurde von einem Online-Tutorial von „How to Mechatronics[27]“ übernommen. Wie in Abbildung 88 ersichtlich, wird der Sensor mit vier Drähten mit dem Arduino Uno Board verbunden. Die Stromversorgung erfolgt über +5V und Ground. Die Datenleitungen verbinden die Trigger und Echo Pins mit den Pins 9 und 10 des Arduino. Abschnitt 7.4 geht näher auf die Programmierung des Arduino Sketches ein. Die kalkulierte Distanz wird dann über die serielle Schnittstelle dem Laptop übergeben. Mit diesem Setup ist es theoretisch möglich den Distanzbereich von 2cm bis 4m mit einer Genauigkeit von 3mm abzudecken (Elek, 2018).

Der Messbereich des Sensors wird mit 15° angegeben (Elek, 2018). Um das bestmögliche Ergebnis zu bekommen, wurde der Sensor direkt in der Mitte des Zapfens, wie in Abbildung 89 zu sehen, angebracht. So wird die Näherung des Benutzers zum Zentrum der Installation gemessen und weitergegeben. Dadurch kann mit der Installation interagiert werden.

[27] https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

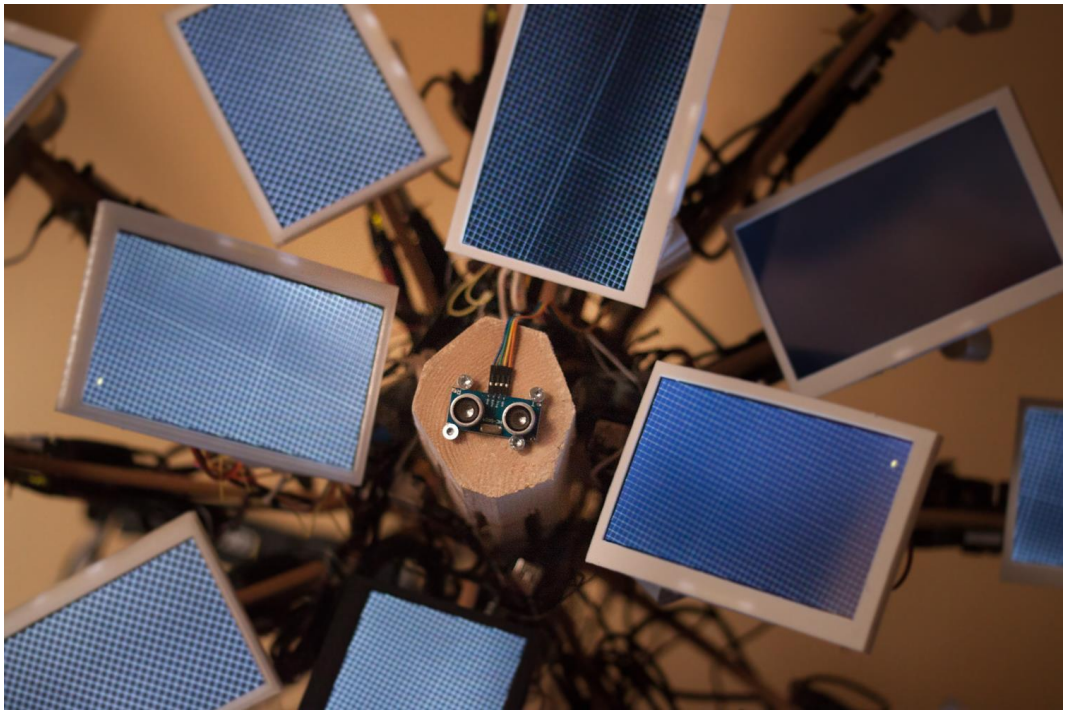


Abbildung 89 – Sensormontage an der Installation

7 Software

7.1 Steuereinheit

7.1.1 Touchdesigner

Das Programm Touchdesigner wurde für die Umsetzung der Datenverarbeitung verwendet. Die vom Sensor empfangenen Daten werden durch Filterung weiter aufbereitet und bilden Grundlage für die Steuerung der Installation. Wie in Abschnitt 5.1.1 besprochen, werden von Touchdesigner aus, die Daten für die Steuerung des Contents gesendet. Außerdem wird die Sonifikation, sowie die Videodokumentation hier umgesetzt.

Das Herzstück des Touchdesigner-Projektes ist die Tabelle „/project1/table1“. Die Tabelle ist in Abbildung 90 und Abbildung 91 in zwei Teilen dargestellt. Darin sind alle wichtigen Parameter zur Steuerung des Contents enthalten. In der Spalte 0 sind alle IP-Adressen aufgelistet. Die folgenden Spalten enthalten alle Werte für die, in Abbildung 58 aufgelisteten, OSC-Adressen. Dadurch kann durch einfaches Eintragen von Zahlenwerten ein Grundsetup erstellt werden.

	0	1	2	3	4	5	6	7	8	9
0	IP	eyeX	eyeY	eyeZ	centerX	centerY	centerZ	lightX	lightY	lightZ
1	192.168.0.101	-40	40	300	-40	35	0	0	0	0
2	192.168.0.102	40	40	300	40	40	0	0	0	0
3	192.168.0.103	40	-40	300	40	-40	0	0	0	0
4	192.168.0.104	-40	-40	300	-40	-40	0	0	0	0
5	192.168.0.105	-30	70	300	-30	70	0	0	0	0
6	192.168.0.106	30	70	300	30	70	0	0	0	0
7	192.168.0.107	70	30	300	70	30	0	0	0	0
8	192.168.0.108	67	-30	300	67	-30	0	0	0	0
9	192.168.0.109	30	-70	300	30	-70	0	0	0	0
10	192.168.0.110	-30	-70	300	-30	-70	0	0	0	0
11	192.168.0.111	0	30	300	0	30	0	0	0	0
12	192.168.0.112	30	0	300	30	0	0	0	0	0
13	192.168.0.113	0	-30	300	0	-30	0	0	0	0
14	192.168.0.114	-30	0	300	-30	0	0	0	0	0
15	192.168.0.115	-70	-30	300	-70	-30	0	0	0	0
16	192.168.0.116	-70	30	300	-70	30	0	0	0	0

Abbildung 90 – /project1/table1 Teil 1

13	14	15	16	17	18	19	20	21	22	23	24	25	26
rollen	neigen	schwenken	scale	transX	transY	transZ	rot	gruen	blau	alpha	rotierenX	rotierenY	rotierenZ
130	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
40	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
315	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
215	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
103	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
72	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
13	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
-13	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
-65	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
-105	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
80	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
-10	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
260	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
170	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
192	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0
168	0	0	1.0	0	0	-500.0	255	255.0	255.0	120	42.0	42.0	0

Abbildung 91 - /project1/table1 Teil 2

Die beiden Operatoren in „/project1/Container1“ dienen als Bauplan für alle OSC Sender, wie in Abbildung 92 zu sehen. Der Operator „constant1“ dient dazu die Werte aus der richtigen Spalte der Tabelle zu laden. Der Wert für eyeX wird beispielsweise mit der Python-Expression „op('../table1')[me.parent().digits, 1]“ geladen. Es wird dabei auf die oben angeführte Tabelle zugegriffen. Mit „me.parent().digits“ wird jeweils die richtige Zeile für den jeweiligen Client aufgrund der Nummerierung ausgewählt. Mit der Nummer 1 wird die Spalte 1 für eyeX ausgewählt. Diese aus der Tabelle abgegriffenen Daten gelangen in den „oscout1“ Operator. Dessen Netzwerkadresse wird ebenfalls mit der Expression „op('../table1')[me.parent().digits, 0]“ ausgewählt. Mit der Ziffer 0 wird auf die Spalte 0 der IP-Adressen zugegriffen.

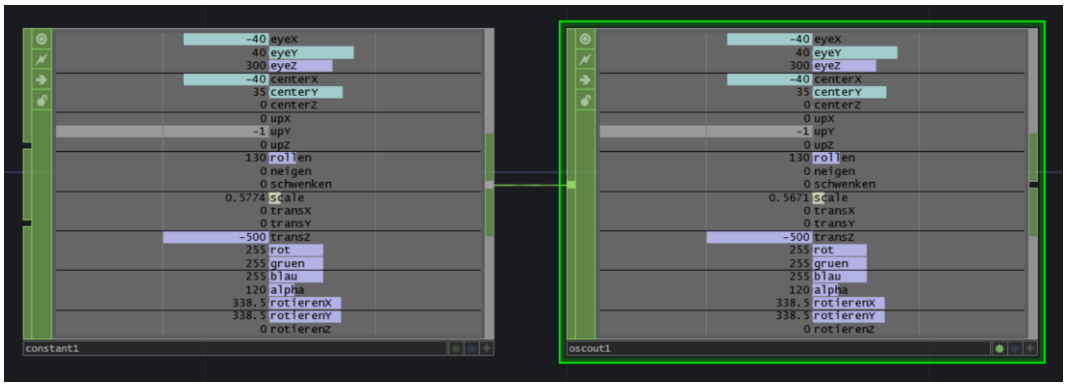


Abbildung 92 – Bauplan OSC Send

Mit dem „/project1/replicator1“ wird dieser obig beschriebene Bauplan für alle IP-Adressen automatisch vervielfältigt. Dadurch werden an jede eingetragene IP-Adresse die jeweiligen Werte aus dessen Zeile gesendet.

Abbildung 93 zeigt die angesteuerte Installation. Dabei ist eine Szene mit einem Kopf und einem weißen Gitter im Hintergrund geladen. Hier sind in der Tabelle

noch willkürliche Werte eingetragen. Die Displays sind hier dadurch noch nicht gemappt.

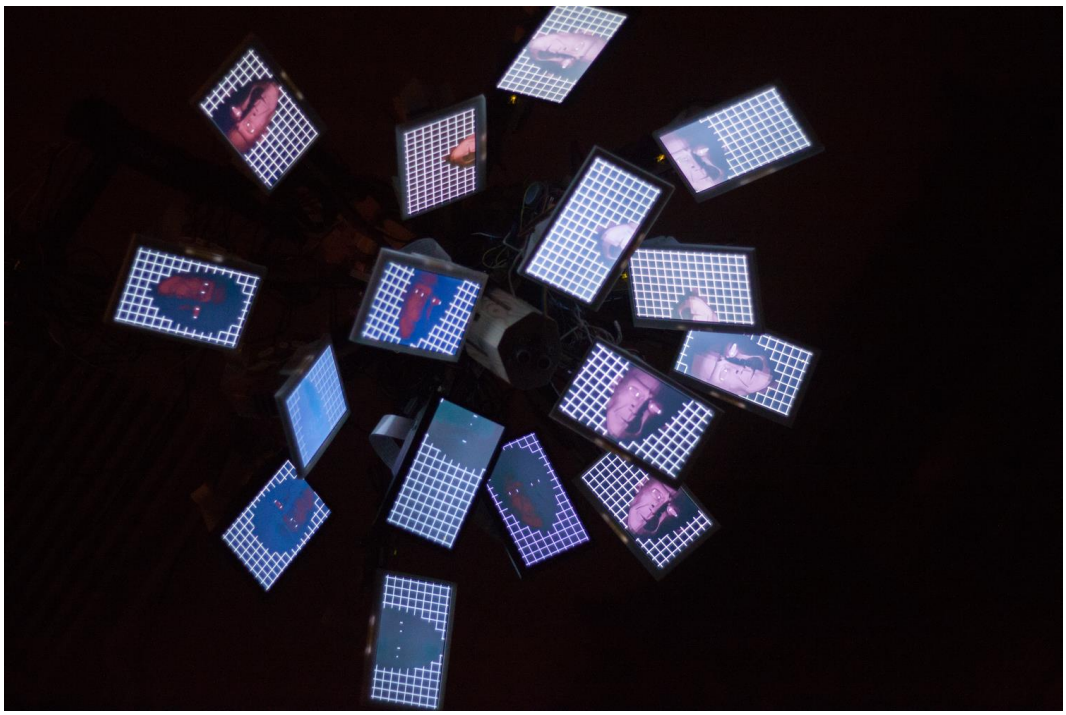


Abbildung 93 – Installation ohne Mapping

Die einzelnen Werte für eyeXYZ, centerXYZ und rollen wurden für alle Clients sukzessive angepasst. Dadurch konnten die Position und Rotation der virtuellen Kamera im Raum für jedes Display angepasst werden. Diese wurden im selben Verhältnis wie die Displays im physikalischen Raum verschoben und gedreht, sodass sich ein gesamtes Bild über die Installation wahrnehmen lässt. Abbildung 94 zeigt die Installation mit eingemappten Content.

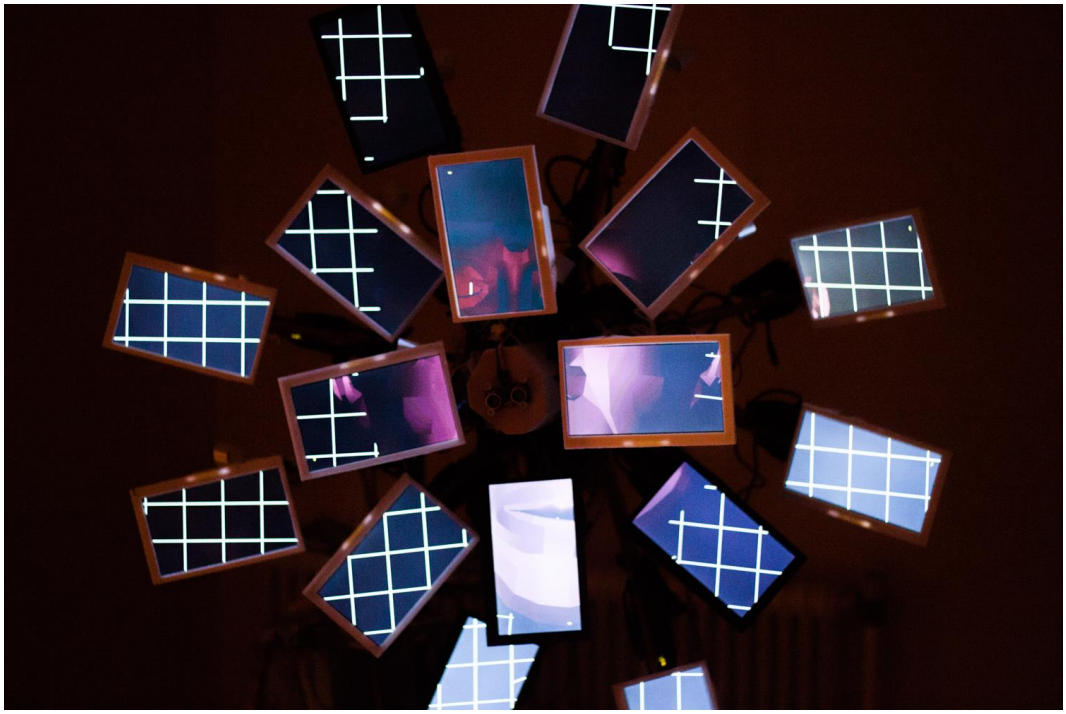


Abbildung 94 – Installation mit Mapping

Mit diesem Setup wäre ständig ein statisches Bild auf den Displays zu sehen. Um im Content Bewegungen und Veränderungen vollziehen zu können werden einige Zellen, der oben besprochenen Tabelle, unter Zuhilfenahme eines Python-Skripts manipuliert. Zuständig dafür sind die beiden Operatoren innerhalb von „/project1“ „chopto1“ und „datexec1“. Abbildung 95 zeigt diese. Im linken Operator werden 5 verschiedene, dynamische Werte, die durch andere Operatoren erzeugt werden, gesammelt. Sobald sich einer dieser ändert, wird das Skript innerhalb von „datexec1“ ausgeführt.

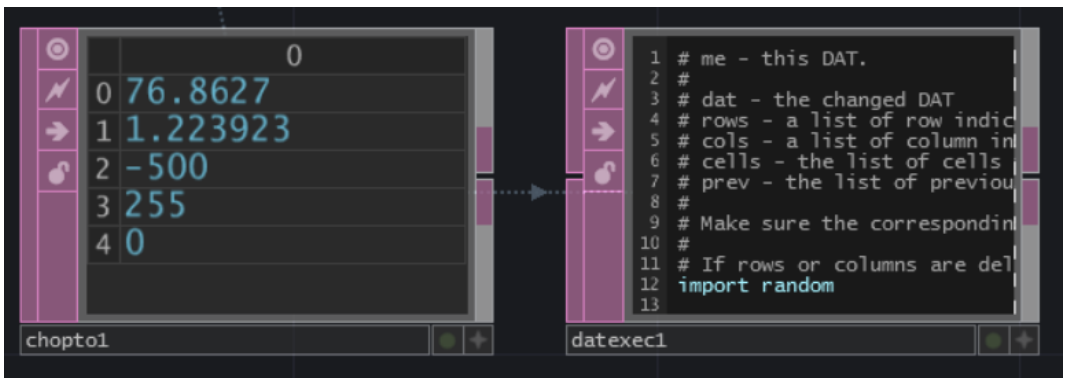


Abbildung 95 – Wertemanipulation

Dieses Skript ist in Code 1 zu sehen. Sobald sich ein Wert der Tabelle ändert, wird die Funktion „onTableChange“ aufgerufen. Mit Hilfe der for-Schleifen werden für alle Clients dieselben Änderungen gemacht. Mit „op(‘table1‘)[i,*]“ wird die zu verändernde Spalte ausgewählt. Die Zellen werden auf bestimmte Werte des „null1“ Operators gesetzt. Mit dem Befehl „random.uniform(0,255)“ werden individuelle, zufällige Werte für die Displays in die einzelnen Farbadressen geschrieben. So wird das blitzartige Event auf der Installation dargestellt.

```
import random

def onTableChange(dat):
   for i in range(1,17):
      op('table1')[i,24] = op('null1')[0].eval()
      op('table1')[i,25] = op('null1')[0].eval()
   for i in range(1,17):
      op('table1')[i,16] = op('null1')[1].eval()
   for i in range(1,17):
      op('table1')[i,19] = op('null1')[2].eval()
   for i in range(1,17):
      op('table1')[i,21] = op('null1')[3].eval()
      op('table1')[i,22] = op('null1')[3].eval()
   if op('null1')[4].eval() == 1:
      for i in range(1,17):
         op('table1')[i,20] = random.uniform(0,255)
         op('table1')[i,23] = random.uniform(0,255)
   else:
      for i in range(1,17):
         op('table1')[i,20] = 255
         op('table1')[i,23] = 120
   return
```

Code 1 – Python-Skript für Wertemanipulation

Für die Sensorik der Installation wird innerhalb der Base „/project1/sensor“ mit Hilfe eines Serial Operators die errechnete Distanz des Arduinos empfangen. Da dieser Wert oft sehr sprunghaft ist, ist eine Aufbereitung des Signals notwendig. Nach Tiefpassfilterung, für eine Glättung des Signals, wird dieses in 4 verschiedene States eingestuft. Der State 1 entspricht dabei der maximalen Distanz und State 4 der nächsten Distanz. Diese States werden einerseits innerhalb von Touchdesigner für Sound und Bild verwendet und andererseits über OSC Port 10000 PC intern an vvvv geschickt um dort ebenfalls die Informationen über die Eingaben des Nutzers zur Verfügung zu haben.

Die Sonifikation wurde innerhalb von „/project1/sound“ umgesetzt. Das grundsätzliche Thema der Geräuschkulisse sind Geräusche aus dem

menschlichen Körper. Als Material für diese Geräusche wurde dieses YouTube-Video[28] herangezogen. Innerhalb von 35min werden mit einem Stethoskop unzählige Körperstellen eines 28 Jahre alten gesunden Mannes abgehört. Diese Sounds geben einen tiefen Einblick in die Geräuschwelt des Körpers. Dabei sind Atmung, Blutfluss und viele andere Ereignisse zu hören.

Das akustische Grundelement dieser Installation ist der Herzschlag. Dabei wurden zwei Loops in der Länge von 7 und 8 Sekunden angelegt. Ein Loop enthält einen langsamen Herzschlag, der andere einen schnelleren. Wie in Abbildung 96 zu sehen, werden beide Files ständig abgespielt. Mit Hilfe des „cross1“ Operators kann zwischen den beiden Files hin und her gefadet werden. Der Fader wird dabei von den States gesteuert. Überschreitet der Benutzer eine bestimmte Grenze, erhöht sich dadurch der Puls der Installation. Diese beiden Files gestalten den tieffrequenteren Bereich der Sonifikation.

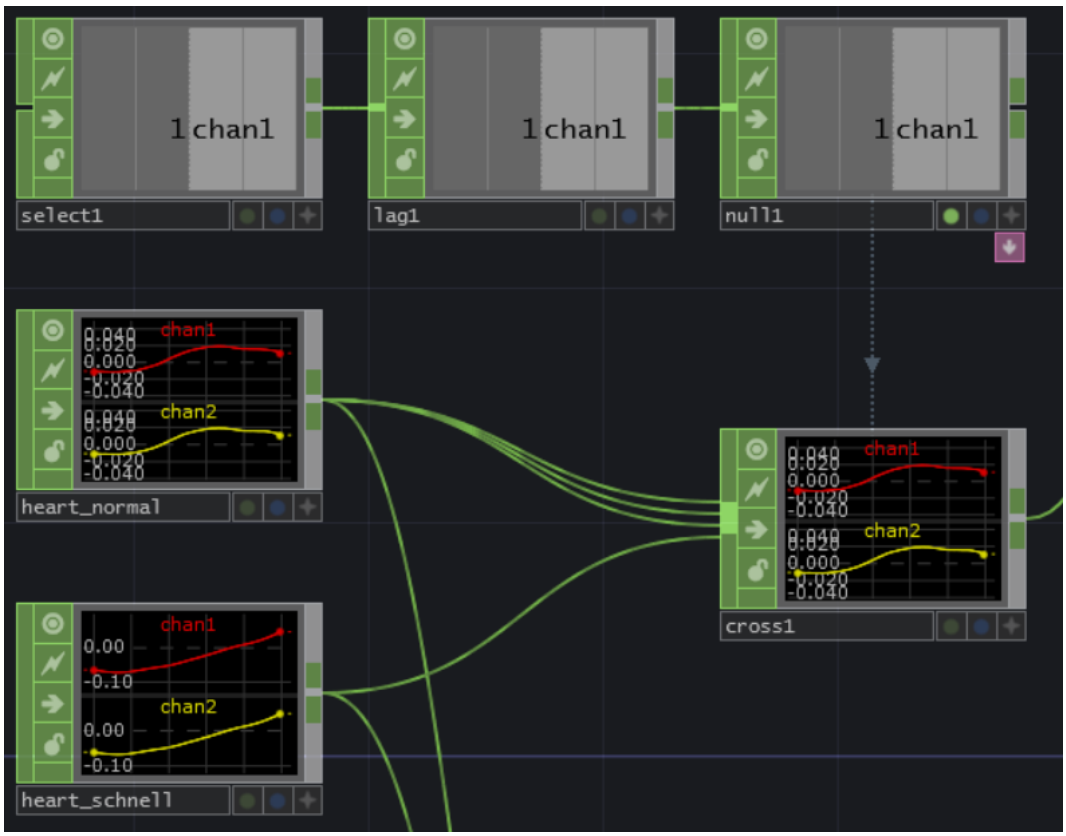


Abbildung 96 - Teile der Sonifikationserzeugung

[28] https://youtu.be/oiMZzVR7f0w

Für den höherfrequenten Bereich läuft kontinuierlich ein Loop, bestehend aus Atemgeräuschen, im Hintergrund. Sobald der Benutzer nah an der Installation steht und der State 4 erreicht ist, wird ein blitzartiges Geräusch abgespielt. Dieses wird durch die Division der beiden Herzgeräusche erzeugt.

7.1.2 vvvv

Das Programm vvvv wurde für die Steuerung der Display Bewegungen eingesetzt. Dabei bekommt der Patch über OSC von Touchdesigner die States, damit die Position des Benutzers bekannt ist. Einerseits entscheiden die States über die Frequenz der Bewegungen und andererseits über die Phasenverschiebung der Bewegungen zueinander. Die errechneten Positions- und Rotationsdaten werden, wie in Abbildung 97 ersichtlich, in Strings gewandelt. Danach werden die beiden Strings mit einem Komma getrennt zusammengeführt. An den UDP Node werden alle 16 Wertpaare sowie alle 16 IP-Adressen angeschlossen. Dadurch werden an jeden Client die jeweiligen Werte geschickt. Fürs Senden wird der Port 5560 verwendet.

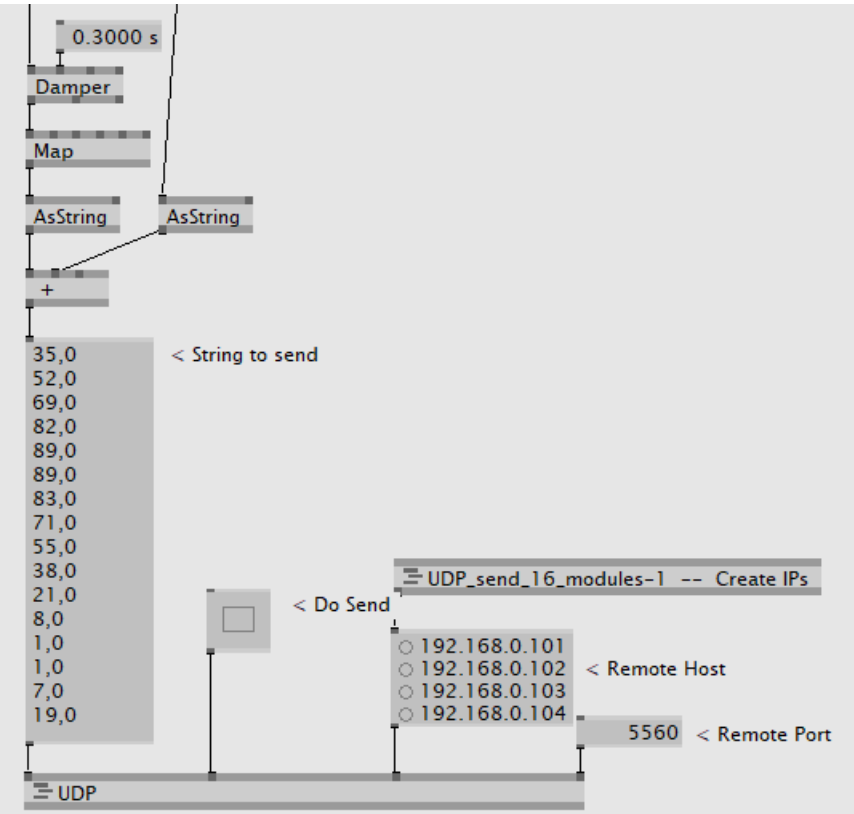


Abbildung 97 – Senden der Rotation und Position in vvvv

7.2 Python Skript

In dieser Arbeit wurde ein Python-Skript für die Ansteuerung der Servo- und Schrittmotoren verwendet. Das Skript läuft unter der Python Version 2.7.13. Wie im nächsten Abschnitt 7.3 beschrieben, wird dieses Skript von Processing nach dem Hochfahren gestartet. Auf jedem Raspberry Pi läuft dasselbe Skript. Durch das Empfangen von UDP Nachrichten, können von außen die Motoren gesteuert werden. Das Skript dient neben der richtigen Weiterleitung der Steuerdaten auch zur Absicherung der Motoren, falls ungültige Steuerdaten empfangen werden. In den nächsten Abschnitten wird das verwendete Skript „receive_stepper_und_servo9_wlan0.py“ zerlegt und erklärt.

Dieses Skript verwendet fünf verschiedene Librarys. Mit dem Netzwerk Interface socket können UDP Nachrichten empfangen werden. GPIO ist für die Ansteuerung der Pins zuständig, mit denen die Schrittmotoren verbunden werden. Die Library wiringpi wird zur Ansteuerung der Servos mittels Hardware PWM genutzt. Abschnitt 7.2.1 geht näher darauf ein. Netifaces wird zur Ermittlung der eigenen IP-Adresse verwendet, dadurch kann dieses Skript auf allen Raspberry Pis ohne Modifikation verwendet werden.

```
import socket
import RPi.GPIO as GPIO
import wiringpi
import netifaces as ni
import time
```

Code 2 – Librarys Python-Skript

7.2.1 Servo

Wie in Kapitel 6 besprochen, ist jedes Display an einem Servomotor montiert, sodass dieses in einem Bereich von 90° rotiert werden kann. Dieser Servo ist neben der Spannungsversorgung mit einer Steuerleitung verbunden. Darüber werden PWM-Signale geschickt, die den Ausschlag des Servos steuern. Bei der Pulsweitenmodulation (PWM) wird die Einschaltdauer eines Rechtecksignals mit konstanter Frequenz verändert. Die Einschaltdauer regelt dabei den Ausschlag des Servos. Diese ist von Servo zu Servo verschieden. Der für diese Arbeit verwendete Servo „AS2103PG“ wird mit einer Frequenz von 50Hz betrieben (Autobotic, 2018).

Die ersten Ansteuerungen wurden mit der GPIO Library durchgeführt. Diese unterstützt aber nur Software-PWM. Dabei wird das PWM-Signal vom Prozessor berechnet. Das heißt, die Genauigkeit des Reckteckssignals kann durch andere Prozesse beeinflusst werden (Kofler u. a., 2015, 412). Bei der Ansteuerung einer

LED mag dies kein Problem darstellen. Bei der Ansteuerung der Servos macht dies jedoch einen signifikanten Unterschied. Das langsam nach links und rechts rotierende Display, machte bei den Tests schnelle, willkürliche Sprünge.

Da diese Ansteuerung über Software-PWM keine zufriedenstellenden Ergebnisse lieferte, wurde die wiringpi Library herangezogen, welche Hardware-PWM auf einem Pin des Raspberry Pis ermöglicht (wiringpi, 2018b).

Für das richtige Setup der PWM-Ansteuerung wird folgender Code verwendet. Mit dem Befehl wiringpi.wiringPiSetupGpio() wird die Bezeichnung der Pins auf die klassische Broadcom GPIO Nummerierung festgelegt (wiringpi, 2018a). Für die Berechnung der PWM Frequenz ergibt sich laut (StackExchange, 2014) folgender Zusammenhang:

pwmFrequenz in Hz = 19.2 MHz / pwmClock / pwmRange

Da die gewünschte pwmFrequenz 50Hz beträgt wurden folgende Werte gewählt:

50 Hz = 19.2 MHz / 375 / 1024

```
servoPIN1 = 18 # gelb

wiringpi.wiringPiSetupGpio()
wiringpi.pinMode(servoPIN1, 2)
wiringpi.pwmSetMode(wiringpi.PWM_MODE_MS)
wiringpi.pwmSetRange(1024)
wiringpi.pwmSetClock(375)
```

Code 3 – wiringpi Setup

In der Hauptschleife des Scripts werden die UDP Nachrichten empfangen. Danach wird der Wert für den Servo vom Wert für den Schrittmotor getrennt und in eigene Variablen gespeichert. Für den Servo werden Werte zwischen 0 und 90 erwartet. Dieser entspricht der Rotation in Grad. Diese Variable data1 wird dann in den neuen Wertebereich zwischen 25 und 66 gesetzt. Diese beiden Werte wurden durch praktisches Probieren herausgefunden. Durch diese map Funktion wird auch eine falsche Ansteuerung der Servos verhindert, da die Steuerdaten immer im richtigen Wertebereich liegen.

```
try:
    while True:
        data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
        MESSAGE1,MESSAGE2 = data.split(",")

        data1 = float(MESSAGE1)
        data2 = float(MESSAGE2)
```

```
        inrangedata1 = maprange((0,90),(25,66), data1)
        wiringpi.pwmWrite(servoPIN1,int(inrangedata1))
        #print inrangedata1

except KeyboardInterrupt:
        GPIO.cleanup()
```

Code 4 - Servoansteuerung

7.2.2 Schrittmotor

Die Ansteuerung des Schrittmotors erfolgt über das ULN2003 Treiber-Board. Vom Raspberry Pi werden über 4 Pins Steuerdaten für den Motor gesendet. Die Stromversorgung erfolgt direkt vom PC-Netzteil über das Treiber-Board. Über diese 4 Pins kann der Schrittmotor gedreht werden, der eine Zahnstange nach vor und zurück bewegt. Rückmeldung über die Position der Zahnstange erfolgt lediglich über einen Taster, der bei einer Extremstellung betätigt wird. Basis für den Code bildet dieses Online-Tutorial[29].

Zur Ansteuerung werden die Pins 24, 23, 17, 4 verwendet, für das Einlesen des Tasters der Pin 7. In der Variable pos wird die aktuelle Anzahl der Umdrehungen gespeichert, wodurch man wiederum auf die Position der Zahnstange schließen kann. In der Variable data2 wird, die von außen empfangene, gewünschte Position der Zahnstange, gespeichert. Der Wert in der Variable Delay hat Einfluss wie schnell die Spulen des Schrittmotors angesteuert werden. Sie legt somit das Verhältnis zwischen Kraft und Geschwindigkeit fest. Die Variablen reftime und reftimecount werden für die Referenzierung des Linear-Aktuators verwendet. Der Schrittmotor vollführt maximal 3200 Umdrehungen bis er den Referenztaster betätigt. Theoretisch findet die Betätigung schon früher statt. Falls ein Fehler auftritt, soll eben nach 3200 Umdrehungen gestoppt werden.

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
coil_A_1_pin = 24 # grau
coil_A_2_pin = 23 # violett
coil_B_1_pin = 17 # blau
coil_B_2_pin = 4 # gruen
taster_pin = 7 # gelb
pos=0
data2=0.0
delay=1
```

[29] https://tutorials-raspberrypi.com/how-to-control-a-stepper-motor-with-raspberry-pi-and-l293d-uln2003a/

```
reftime=3200
reftimecount=0
```

Code 5 – Variablen Schrittmotor

Für die richtige Ansteuerung der Schrittmotor Spulen, wird eine Liste mit acht verschiedenen High- und Low-Pegeln angelegt.

```
StepCount = 8
Seq = range(0, StepCount)
Seq[0] = [1,0,0,0]
Seq[1] = [1,1,0,0]
Seq[2] = [0,1,0,0]
Seq[3] = [0,1,1,0]
Seq[4] = [0,0,1,0]
Seq[5] = [0,0,1,1]
Seq[6] = [0,0,0,1]
Seq[7] = [1,0,0,1]
```

Code 6 – Ansteuerungssequenz Schrittmotor

Mit der Funktion setStep werden die Sequenzen über die Pins an das Treiberboard geschickt. Die Funktion resetStep verhindert unnötige Wärmeentwicklung im Schrittmotor. Während der Linear-Aktuator ruht, werden damit alle Spulen deaktiviert.

```
def setStep(w1, w2, w3, w4):
    GPIO.output(coil_A_1_pin, w1)
    GPIO.output(coil_A_2_pin, w2)
    GPIO.output(coil_B_1_pin, w3)
    GPIO.output(coil_B_2_pin, w4)

def resetStep():
    GPIO.output(coil_A_1_pin, 0)
    GPIO.output(coil_A_2_pin, 0)
    GPIO.output(coil_B_1_pin, 0)
    GPIO.output(coil_B_2_pin, 0)
```

Code 7 – Schrittmotor ansteuern

Die folgende Funktion ruft die Schrittmotorsequenz so auf, damit der Linear-Aktuator vorwärtsbewegt wird. In jedem durchlauf wird pos um eins erhöht, sodass die Position mitverfolgt werden kann. Die Rückwärtsbewegung erfolgt analog dazu in der backwards Funktion.

```
def forward(delay, steps):
    for i in range(steps):
        global pos
        pos+=1
```

```
        for j in range(StepCount):
            setStep(Seq[j][0], Seq[j][1], Seq[j][2], Seq[j][3])
            time.sleep(delay)
```

Code 8 – Linear-Aktuator Bewegung vorwärts

Die Funktion my_callback fängt die Betätigung des Schalters ab. Aus Sicherheitsgründen öffnet der Schalter bei Betätigung den Stromkreis. Sobald der Linear-Aktuator an der Extremposition angekommen ist und den Schalter geöffnet hat, wird pos auf 0 gesetzt und die Variable istref auf True. Die Funktion resetStep soll wie oben erwähnt, die Wärmeentwicklung verhindern.

```
def my_callback(channel): ##### Taster fuer Referenzierung
    if channel == taster_pin:
      if GPIO.input(taster_pin):
        global pos
        pos=0
        global istref
        istref = True
        resetStep()
```

Code 9 – Tasterbetätigung Linear-Aktuator

Die Funktion referenzieren wird nach dem Hochfahren des Scripts aufgerufen. Dabei fährt die Zahnstange, unter Beachtung der reftime von 3200, solange zurück, bis der Schalter betätigt wird. Beim Empfangen einer 0 von außen, wird ebenfalls diese Funktion aufgerufen. Dieses wiederholende Referenzieren soll ein Hinausfahren der Zahnstange verhindern.

```
def referenzieren():
    global reftimecount
    global reftime
    global istref
    global varfalse
    global taster_pin
    global data2
    while reftimecount < reftime and istref == varfalse and
          GPIO.input(taster_pin) == 0 and data2 == 0:
        backwards(int(delay) / 1000.0, int(1))
        reftimecount = reftimecount + 1
    else:
        pos = 0
        istref = True
        resetStep()
```

Code 10 – Linear-Aktuator referenzieren

Die Variable data2 wird kontinuierlich über die UDP Nachrichten von außen aktualisiert. In der Hauptschleife des Scripts findet ein Vergleich der errechneten Position des Linear-Aktuators und der Sollposition statt. Entweder werden die Funktionen forward oder backward aufgerufen. Falls eine 0 gesendet wird, findet eine Referenzierung statt, falls die beiden Werte gleich sind, wird die Ansteuerung ausgeschalten.

```
if data2 > pos and data2 < 2801:
   forward(int(delay) / 1000.0, int(1))
elif data2 < pos and data2 > 0:
   backwards(int(delay) / 1000.0, int(1))
elif data2 == 0 and pos != 0:
   reftimecount = 0
   istref = False
   referenzieren()
elif data2 == pos:
   resetStep()
```

Code 11 – Positionssteuerung Linear-Aktuator

Dadurch kann durch das Senden von zwei Werten, nämlich einen für die Rotation des Servos und einen für die Position des Linear-Aktuators, das Modul im Raum positioniert werden. Das Raspberry Pi übernimmt die Wandlung von UDP Nachrichten in Steuerdaten. Im nächsten Abschnitt wird näher auf die Erzeugung der Bilder eingegangen.

7.3 Processing Sketch

Die Software Processing ist ein flexibles Skizzenbuch und eine Sprache mit der Projekte im Umfeld der visuellen Kunst realisiert werden können (Fry & Reas, 2018b). In dieser Arbeit wurde Processing zur Generierung der Bilder für jedes Modul verwendet. Auf jedem Raspberry Pi läuft derselbe Sketch. Durch die, über Netzwerk empfangenen OSC-Daten, werden bestimmte Parameter verändert, sodass jedes Modul ein eigenes Bild wiedergibt.

Der zentrale Bestandteil des verwendeten Sketches „szene1“ ist ein Ikosaeder, ein geometrischer Körper mit 20 identen gleichseitigen Dreiecken als Seitenflächen. Im Hintergrund befindet sich ein weißes geradliniges Gitter, dessen Rasterung kleiner als die Auflösung der Bildschirme ist. Durch unterschiedliche Skalierung des Gitters ergibt sich aufgrund der Überlagerung mit den Displaypixeln ein Moiré-Effekt. Abbildung 98 zeigt einen Ausschnitt aus dem gerenderten Processing Sketch „szene1“.

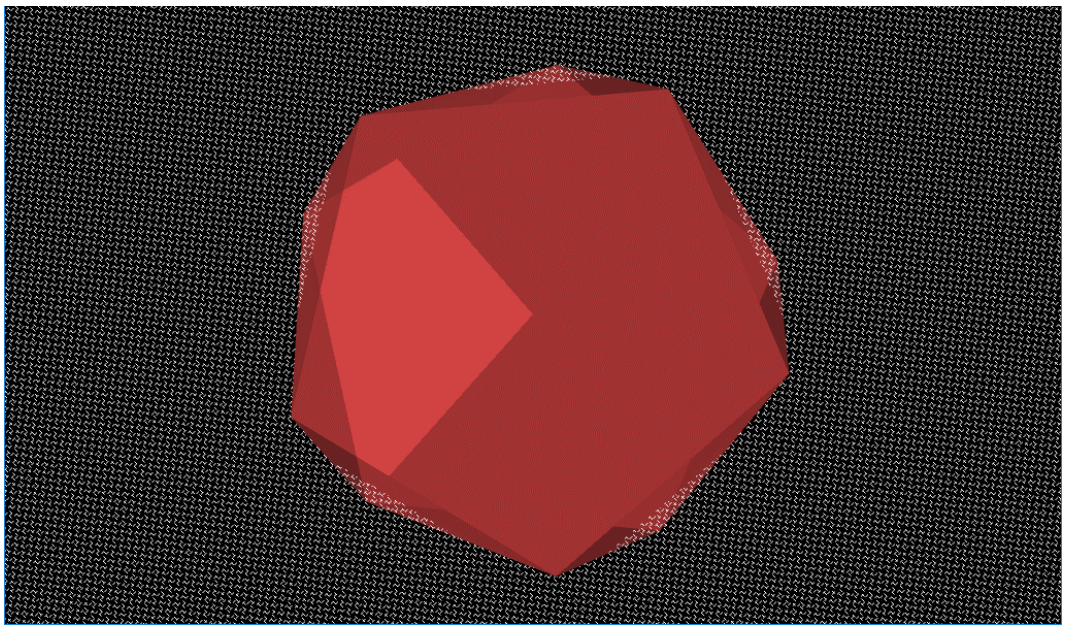


Abbildung 98 – Processing Render

In den folgenden Absätzen wird näher auf die einzelnen Abschnitte des Processing Sketches „szene1“ eingegangen. Das Programm wird einzeln aus unterschiedlichen Perspektiven betrachtet und ist deshalb getrennt dargestellt. Der Sketch „szene1“ besteht aber aus einem langen Textfile, es befindet sich im Anhang.

Dem laufenden Sketch werden kontinuierlich 23 Werte über OSC gesendet, die das Aussehen des gerenderten Bildes verändern. Der folgende Abschnitt 7.3.1 geht näher auf die Kommunikation ein. Diese 23 Werte sind in Code 12 zu sehen.

```
float eyeX=0;
float eyeY=0;
float eyeZ=300;
float centerX=0;
float centerY=0;
float centerZ=0;
float upX;
float upY;
float upZ;
float rollen;
float neigen;
float schwenken;
float scale;
float transX;
float transY;
float transZ;
float rot;
```

```
float gruen;
float blau;
float alpha;
float rotierenX;
float rotierenY;
float rotierenZ;
```

Code 12 – Variablen Sketch „szene1“

Die eyeXYZ Variablen legen die Position der Kamera fest, die centerXYZ Variablen die Position wohin sich die Kamera richtet. Mit upXYZ können die Richtungen des Koordinatensystems geändert werden. Die Variablen rollen, neigen, schwenken legen die 3 Rotationen der Kamera im Raum fest. Mit scale werden die Objekte in der Szene skaliert und mit transXYZ verschoben. Die Variablen rot, gruen, blau und alpha legen die Farbe des Ikosaeders fest. Mit rotierenXYZ kann das Objekt um sich selbst rotiert werden.

Der Ikosaeder wird zu Beginn instanziiert. Die „Icosahedra-Class“ wurde von Ira Greenberg zur Verfügung gestellt und von der offiziellen Processing Website verwendet (Fry & Reas, 2018a). Mit dem Befehl exec() kann direkt eine Eingabe in der Kommandozeile des Raspberry Pis erfolgen. Dieser startet das Python-Skript, welches für die Steuerung der Motoren zuständig ist. Mit noCursor() wird der Mauszeiger ausgeblendet und mit fullScreen(P3D) wird der 3D Renderer im Vollbildmodus gestartet.

```
Icosahedron ico1;

void setup() {
   exec("sudo", "python",
                "/home/pi/receive_stepper_und_servo9_wlan0.py");
   noCursor();
   fullScreen(P3D);
   ico1 = new Icosahedron(75);
}
```

Code 13 – Ikosaeder Instanziierung

Innerhalb der Draw-Funktion wird ein schwarzer Hintergrund dargestellt und die Standard Lichteinstellungen festgelegt. Mit Hilfe von push- und popMatrix werden die aktuellen Transformationen abgelegt und wiederhergestellt. Dazwischen wird jedes Mal der Ikosaeder mit der aktuellen Translation, Skalierung, Rotation und Farbe gezeichnet. Innerhalb der for-Schleifen wird das skalierte Gitter mit Hilfe von Linien dargestellt.

```
void draw() {

  background(0);
  lights();

  pushMatrix();
  translate(transX, transY, transZ);
  scale(scale);
  rotateX(rotierenX);
  rotateY(rotierenY);
  rotateZ(rotierenZ);
  noStroke();
  fill(rot,gruen,blau,alpha);
  ico1.create();
  popMatrix();

  stroke(255);
  noFill();

  scale(scale*0.5);
  for(int i=-1000; i<1000; i+=2){
   line(i,-1000,i,1000);
  }
  for(int w=-1000; w<1000; w+=2){
   line(-1000,w,1000,w);
  }
}
```

Code 14 – Content Erzeugung

7.3.1 oscP5

oscP5 ist eine Library für Processing, die die Verwendung von Open Sound Control (OSC) ermöglicht. Sie wurde von Andreas Schlegel zur Verfügung gestellt und 2012 das letzte Mal upgedated. Mit OSC können Softwareanwendungen auf Computern, Synthesizer und andere Multimediageräte über Netzwerk aber auch intern miteinander kommunizieren (Schlegel, 2010).

Mit folgenden Befehlen wird die Library zu Beginn des Sketches importiert und instanziiert.

```
import oscP5.*;
import netP5.*;

OscP5 oscP5;
NetAddress myRemoteLocation;
```

Code 15 – oscP5 Import

Innerhalb der Setup-Funktion wird oscP5 gestartet. Es wird auf dem Port 12000 auf eingehende Nachrichten gehört.

```
void setup() {
   oscP5 = new OscP5(this,12000);
}
```

Code 16 – oscP5 Port

Sobald eine OSC-Nachricht eintrifft und sie die Adresse „/eyeX“ aufweist, wird dieser enthaltene Wert in die Variable „eyeX“ gespeichert.

```
void oscEvent(OscMessage theOscMessage) {
  if(theOscMessage.checkAddrPattern("/eyeX")==true)
  {
    eyeX = theOscMessage.get(0).floatValue();
  }
}
```

Code 17 – oscP5 Adressenüberprüfung

Dadurch ist es möglich in jeden Processing Sketch von außen über OSC eingreifen zu können, um so die Werte von verschiedenen Variablen ändern zu können.

7.3.2 Obsessive Camera Direction (OCD)

Diese Processing Library wurde von Kristian Linn Damkjer zur Verfügung gestellt. Sie ermöglicht es innerhalb von Processing einfach virtuelle Kameras zu erzeugen und mit diesen, intuitive Bewegungen zu vollführen. Es können hier bekannte Parameter wie zum Beispiel, Zoom, Truck, Boom, Dolly, Tilt, Pan, Roll und so weiter gesetzt werden. Diese Parameter sind für diese Arbeit deshalb sinnvoll, weil dadurch auf jedem Modul der gleiche Sketch laufen kann und von außen die virtuellen Kameras positioniert werden können, sodass sich über die gesamte Installation das Bild wahrnehmen lässt.

Mit den folgenden Befehlen wird diese Library importiert und eine neue Kamera instanziiert:

```
import damkjer.ocd.*;

Camera camera1;
float cameraZ;
```

Code 18 – OCD Import

Innerhalb der Setup-Funktion werden die Kameraeinstellungen festgelegt. Die Variable cameraZ dient zur Errechnung von nearClip und farClip, von wo bis wohin

die Kamera quasi sieht. Die eye-Variablen legen die Position der Kamera fest, die center-Variablen den Punkt, auf den die Kamera schaut. Der Field of View (FOV) wurde folgendermaßen festgelegt. Wenn der Betrachter der Installation 75cm davon entfernt ist, entspricht sein Blickwinkel durch die Displays, dem FOV der virtuellen Kamera.

```
void setup() {
   cameraZ = (height/2.0) / tan(PI*60.0/360.0);
   camera1 = new Camera(this, eyeX, eyeY, eyeZ,
                              centerX, centerY, centerZ,
                              0, -1, 0,
                              0.112851804, cameraZ/10, cameraZ*10);
}
```

Code 19 – OCD Kamerasettings

Um die Kamera auch während dem Laufen des Sketches bewegen zu können, legen in der Draw-Funktion die eye-Variablen mittels camera1.jump() und die center-Variablen mittels camera1.aim() die Position und Richtung der Kamera fest. Um auch die Rotationen der Kamera verändern zu können werden die Funktionen tilt(), pan() und roll() aufgerufen.

```
void draw() {

  camera1.jump(eyeX, eyeY, eyeZ);
  camera1.aim(centerX, centerY, centerZ);
  tilt();
  pan();
  roll();
  camera1.feed();

}
```

Code 20 – OCD Kamerabewegungen

Die von OCD zur Verfügung gestellten Methoden können nur relativ rotieren, das heißt die Kamera bei jedem Durchlauf der Draw-Funktion nur um X rotieren lassen und nicht die Kamera bis zur Rotation X hin rotieren. Da aber von außen ein bestimmter Wert gesendet werden soll, auf die sich die Kamera drehen soll, war es notwendig für roll, tilt und pan eine eigene Funktion zu schreiben. Folgendes Codebeispiel zeigt die Funktion für roll. Dabei wird der aktuelle Wert der Rotation abgefragt, mit dem von außen gewünschtem Wert verglichen und um deren Differenz rotiert. Dadurch ist es von außen möglich, absolute Werte für die Rotationen zu senden.

```
void roll(){
```

```
  float[] attitude = camera1.attitude();
  if(floor(degrees(attitude[2])-rollen) > 0){
    for(int i = 0; i < abs(floor(degrees(attitude[2])-rollen));i ++){
      camera1.roll(radians(-1));
    }
  }
  if(floor(degrees(attitude[2])-rollen) < 0){
    for(int i = 0; i < abs(rollen-floor(degrees(attitude[2])));i ++){
      camera1.roll(radians(1));
    }
  }
}
```

Code 21 – OCD roll, tilt, pan Funktionen

7.4 Arduino Sketch

Für den Arduino Sketch wurde der Code dieses Online-Tutorials[30] verwendet. Dieser Code konnte ohne Probleme über die Arduino IDE[31] auf den Arduino Uno gespielt werden.

Wie in Code 22 zu sehen, wird in der Hauptschleife des Sketches am Trigger-Pin ein Impuls von 10µs abgegeben. Dieser startet am Ultraschallmodul den Sendevorgang der akustischen Impulse.

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

Code 22 – Sensor – Senden

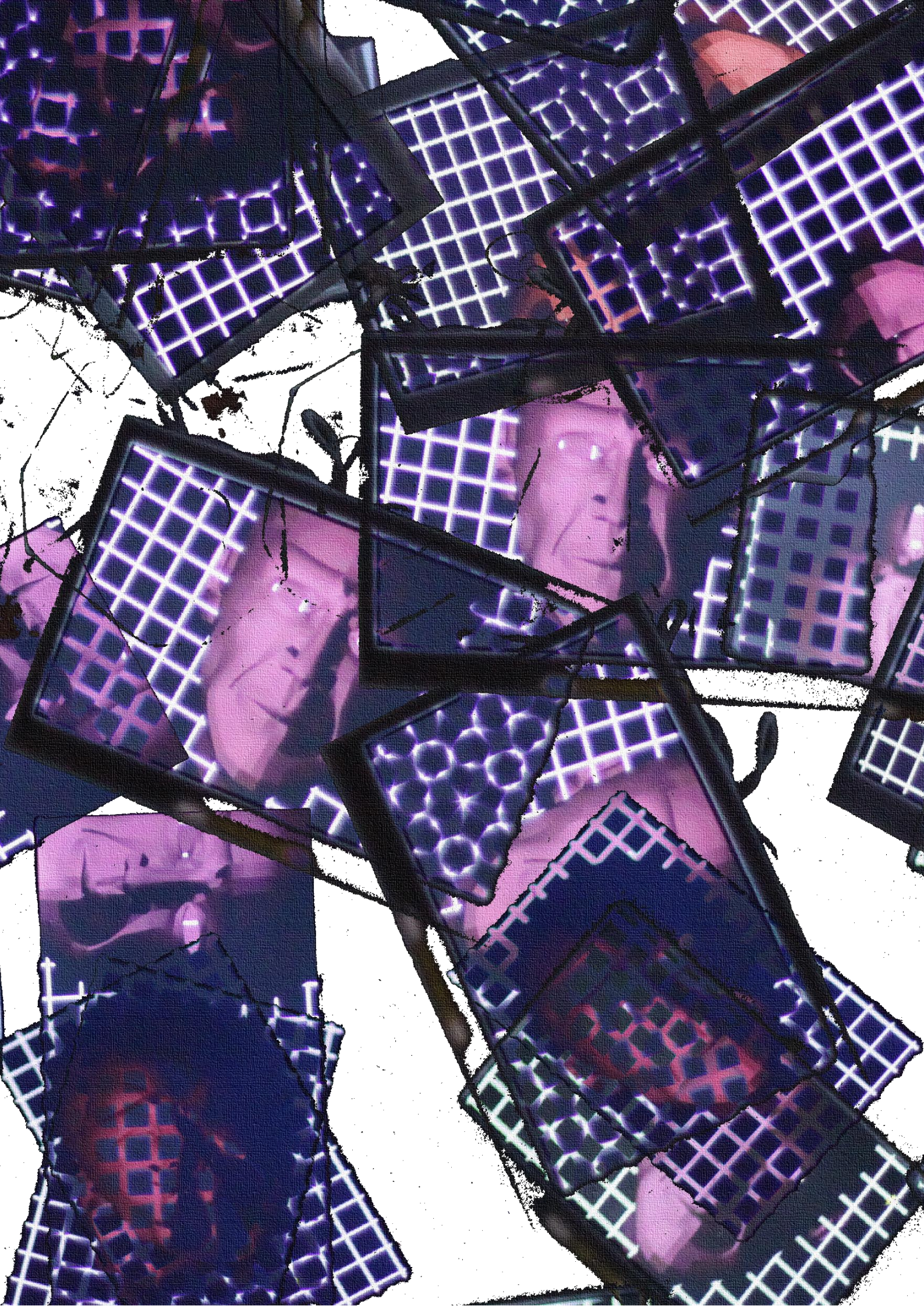
Danach wird die Dauer gemessen, bis am Echo-Pin eine reflektierte Welle ankommt. Mit der in Abschnitt 6.7 vorgestellten Formel wird auf die Distanz rückgeschlossen und diese seriell ausgegeben.

```
duration = pulseIn(echoPin, HIGH);
distance= duration*0.034/2;
Serial.println(distance);
```

Code 23 – Sensor – Empfangen

[30] https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

[31] https://www.arduino.cc/en/Main/Software



8 Der Bauplan

8.1 Software

Für die Einrichtung der Raspberrys kann entweder direkt das fertige Image geladen werden oder Schritt für Schritt neu installiert werden.

8.1.1 Fertiges Image

Das fertige Image kann direkt von hier[32] geladen werden. Mit dem Programm Win32DiskImager[33] kann das Image auf alle Speicherkarten der Raspberrys geschrieben werden.

Nach Fertigstellung muss lediglich an der Stelle der Sternchen (***) die IP-Adresse geändert werden und eine Verbindung mit dem Netzwerk hergestellt werden:

```
sudo nano /etc/dhcpcd.conf
```

```
interface wlan0
static ip_address=192.168.0.***/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1
```

```
sudo reboot
```

Danach können die Raspberrys über vvvv und Touchdesigner angesteuert werden. Beide Files sind auf GitHub[34] verfügbar.

8.1.2 Neue Installation

Von der offiziellen Raspberry-Website[35] kann das aktuelle Raspbian-Image downgeloadet werden. Für diese Arbeit wurde das Raspbian-Image der Version 9 mit der Kernel-Version 4.14.52+ verwendet. Das Image kann wieder mit Win32DiskImager auf die Karte des Raspberrys geschrieben werden.

---

32 https://fhstp-my.sharepoint.com/:u:/g/personal/dm161560_fhstp_ac_at/EW6FDfZxjm5OjANzsmvuRTkBsXHmRq2qElmcvoSbSqHDeA?e=zZFICC

33 https://sourceforge.net/projects/win32diskimager/

34 https://github.com/fabwald-vienna/Rosette-4.0

35 https://www.raspberrypi.org/downloads/raspbian/

Sicherheitshalber kann das System auf den neuesten Stand gebracht werden:

```
sudo apt-get update

sudo apt-get upgrade
```

In den Einstellungen müssen SSH und Remote-GPIO aktiviert werden:

```
sudo raspi-config

    enable: ssh, remote gpio
```

Für die Einstellung der richtigen Auflösung kann dieses Textfile folgend geändert werden[36]:

```
sudo nano /boot/config.txt
```

```
hdmi_group=2
hdmi_mode=87
hdmi_cvt 1024 600 60 3 0 0 0
hdmi_force_hotplug=1
```

Anstatt der Sternchen (***) wird in diesem File die gewünschte IP eingetragen.

```
sudo nano /etc/dhcpcd.conf
```

```
interface wlan0
static ip_address=192.168.0.***/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1
```

Um über die USB Ports der Raspberrys maximal verfügbaren Strom zu bekommen wird folgendes in diesem File angehängt[37]:

```
sudo nano /boot/config.txt
```

```
max_usb_current=1
```

[36]https://www.wiki.52pi.com/index.php?title=7-Inch-1024x600_Display_Kit_(without_Touch_Screen)_SKU:Z-0051

[37] https://projects.drogon.net/testing-setting-the-usb-current-limiter-on-the-raspberry-pi-b/

Mit diesem Befehl wird Processing downgeloadet und installiert:

```
curl  https://processing.org/download/install-arm.sh  |
sudo sh
```

Danach muss der Raspberry neu gestartet werden:

```
sudo reboot
```

Damit im Python-Skript die Netzwerkadresse automatisch ermittelt werden kann, muss Netifaces installiert werden[38]:

```
sudo pip install netifaces
```

Damit die Hardware PWM verwendet werden kann, muss wiringpi installiert werden[39]:

```
sudo apt-get install python-dev python-pip

sudo pip-3.2 install wiringpi2

sudo pip install wiringpi2

sudo python

    import wiringpi
```

Strg+D für Exit

Nach der Verbindungsherstellung mit dem gewünschten Netzwerk, ist hier der Raspberry fertig aufgesetzt. Nun können die notwendigen Files raufgespielt werden.

[38] https://pypi.org/project/netifaces/

[39] https://pypi.org/project/wiringpi/

Das Python-Skript sowie der Processing Sketch sind auf GitHub[40] zum Download verfügbar. Das Python-Skript kann mit MobaXterm über SFTP in folgenden Ordner kopiert werden:

```
/home/pi/
```

Das Python-Skript kann auch anderswo gespeichert werden. Wichtig ist nur das der Processing Sketch es beim Durchlaufen der Setup-Funktion startet.

Der Processing Sketch wird am Laptop, der ebenfalls mit dem gleichen Netzwerk verbunden ist, geöffnet. Mit dem „Upload to Pi[41]“ Tool kann der Sketch auf den Raspberry geladen werden. Dadurch startet der Sketch jedes Mal nach dem Booten.

Ist dies alles erledigt, ist der Raspberry bereit zum Einsatz. Mit Win32DiskImager kann dieses Image auf die anderen Raspberrys übertragen werden.

Danach kann die Installation mit Touchdesigner und vvvv gesteuert werden. Die beiden Files sind ebenfalls auf GitHub[42] verfügbar.

8.2 Hardware

Die Hardware wurde aufgrund der hohen Stückzahlen Großteils von AliExpress bezogen, Raspberrys von Reichelt. Folgende Tabelle listet die verwendeten Komponenten mit Links dazu auf:

ULN2003 Treiber-Board	https://de.aliexpress.com/item/10Pcs-ULN2003-Stepper-Motor-Driver-Board-AVR-ULN2003A-Module-For-Arduino-28BYJ-48-XH-5P-Electronic/32868226874.html
Flachband-kabel 25cm inkl. Adapter	https://de.aliexpress.com/item/25CM-Type-A-0-5-Pitch-50P-FFC-Extension-cord-Connector-50-Pin-TTL-Flexible-Flat/32808774705.html

[40] https://github.com/fabwald-vienna/Rosette-4.0

[41] https://github.com/gohai/processing-uploadtopi

[42] https://github.com/fabwald-vienna/Rosette-4.0

Flachband-kabel 50cm	https://de.aliexpress.com/item/WZSM-Wholesale-New-FFC-FPC-Flexible-Flat-Cable-0-5mm-Pitch-50-Pin-Length-400mm-500mm/32818560410.html
Servos	https://de.aliexpress.com/item/4-2kg-torque-analog-servo-easy-FPV-Pan-Tilt-Camera-Platform-Camera-Mount-For-Aircraft-FPV/1942923446.html
Displays	https://de.aliexpress.com/item/7-inch-LCD-Panel-Digital-LCD-Screen-and-Drive-Board-HDMI-VGA-2AV-for-Raspberry-PI/32314647847.html
HDMI Kabel 50cm	https://de.aliexpress.com/item/High-Speed-HDMI-Cable-Gold-Plated-Connection-with-Red-black-and-white-mesh-1080P-0-5m/32469824473.html
Raspberry Pi 3 Model B	https://www.reichelt.at/raspberry-pi-3-b-4x-1-2-ghz-1-gb-ram-wlan-bt-raspberry-pi-3-p164977.html?r=1
Raspberry Pi Zero WH	https://www.reichelt.at/raspberry-pi-zero-wh-v-1-1-1-ghz-512-mb-ram-wlan-bt-rasp-pi-zero-wh-p222531.html?r=1
OTG-Adapter	https://www.reichelt.at/raspberry-pi-zero-hi-speed-otg-adapter-rpiz-usb-otg-p223604.html?&trstct=pos_5
microUSB Stromkabel	https://www.reichelt.at/micro-usb-b-freie-enden-1-8-m-sw-musb-10080100-p198956.html?&trstct=pos_1
mini HDMI auf HDMI Adapter	https://www.reichelt.at/adapter-hdmi-buchse-auf-mini-hdmi-stecker-ad-hdmi-mini-p103082.html?&trstct=pos_0
Ultraschall-sensor	https://www.amazon.de/Elegoo-HC-SR04-Ultraschallmodul-Distanzsensor-MEGA2560/dp/B01M9CMJ9O/ref=sr_1_1_sspa?ie=UTF8&qid=1536438205&sr=8-1-spons&keywords=hcsr04&psc=1
Linear-Aktuator	https://www.ebay.com/itm/Linear-Actuator-Stepper-Motor-4-Arduino-Driver-Raspberry-PI-micro-OTAMAT-UK-/282363055991
PC-Netzteile	Alter Bestand

Tabelle 11 – Linkliste Komponenten

Die Displayhalterungen und Verbindungselemente wurden 3D gedruckt. Abschnitt 6.1 geht näher darauf ein. Die Displayhalterung[43] und das Verbindungselement[44] sind auf Thingiverse zum Download verfügbar.

Die Holzkonstruktion sowie die Bretter der Module sind eigene Anfertigungen. Es existieren leider keine Pläne dafür.

Falls die Links von der gedruckten Form aus aufgerufen werden wollen, stehen hier die notwendigen QR-Codes zur Verfügung.

Link zum fertigen Image

Link zu GitHub

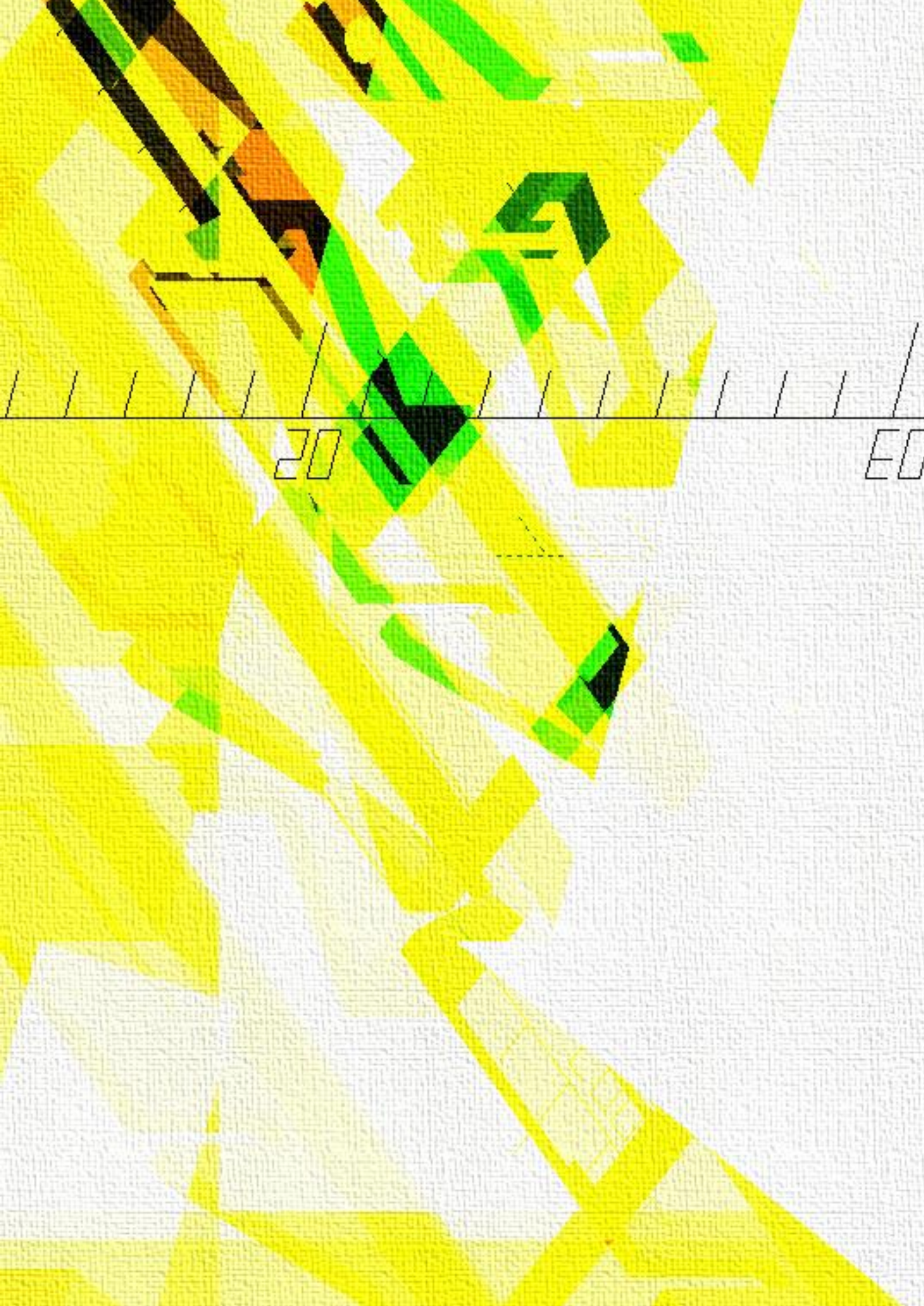
Python Skript
Processing Sketch
Arduino Sketch
vvvv Patch
Touchdesigner Project

Link zu Thingiverse

Displayhalterung
Verbindungselement

[43] https://www.thingiverse.com/thing:3086755

[44] https://www.thingiverse.com/thing:3087666


20

9 Qualitative Befragung

9.1 Handlungsort

Zur Beantwortung der restlichen Forschungsfragen und zum Einholen von Feedback, wurde die funktionierende Installation den externen Stakeholdern präsentiert. Dabei wurde die Installation in einem Abstellraum eines Einfamilienhauses inszeniert. Der Raum befindet sich im Keller des Hauses. Dieser ist über Betonstiegen von außen zugänglich. Abbildung 99 zeigt den Abgang und den Raum mit geöffneter Tür. Während der Inszenierung war die Türe geschlossen. Beim Hinabsteigen der Treppen war lediglich das tiefe Wummern des pulsierenden Herzschlages zu fühlen.

Abbildung 99 – Zugang zum Abstellraum

Während des gesamten Abends war der Raum, bis auf das Licht der Displays, dunkel. Beim Öffnen der Türe waren die Körpergeräusche deutlicher zu hören. Man erblickte die bewegte Installation in etwa 4 Metern Entfernung. Das Zentrum der Installation war etwa auf Brusthöhe eines Erwachsenen Menschen. Abbildung 100 zeigt die Installation aus der Perspektive des Betrachters.

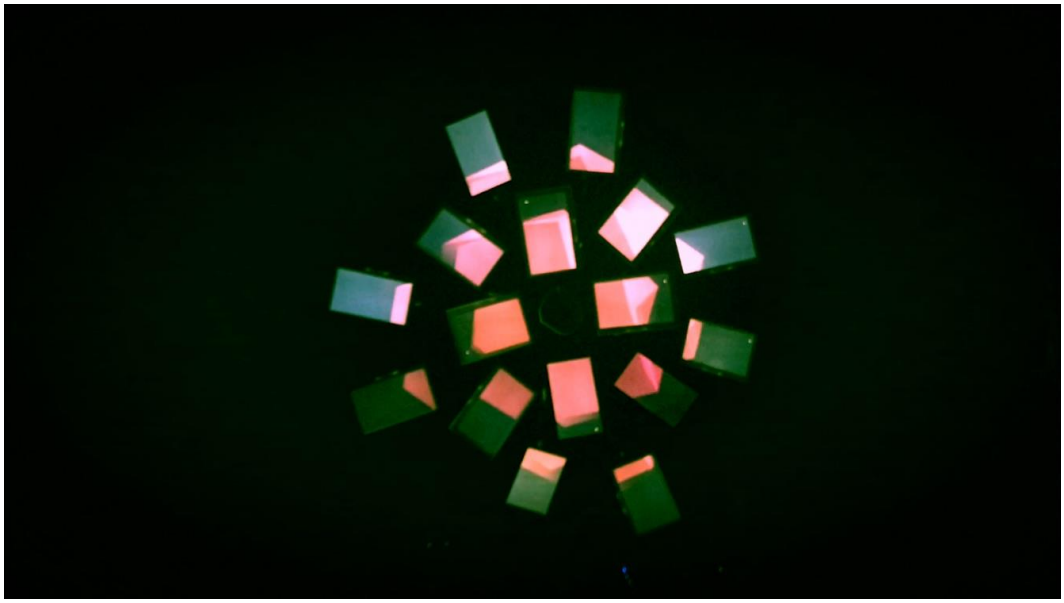


Abbildung 100 – Installation beim Betreten des Abstellraums

Innerhalb des Raumes stand ein etwa 1,5m x 4m langer Gang zur Verfügung, um sich der Installation zu nähern. Abbildung 101 zeigt diesen Raum mit eingeschaltetem Licht. Wie in Abbildung 101 ersichtlich, wurde dieser Gang links durch Objekte am Boden und rechts durch ein Regal begrenzt. Dadurch wurde der Betrachter der Installation gezwungen, sich vor der Installation aufzuhalten. Der Gang war aber groß genug, um sich in ihm frei zu bewegen und die Installation aus unterschiedlichen Perspektiven zu betrachten. Durch den Ultraschallsensor an der Spitze der Installation, konnte diese den Betrachter wahrnehmen und darauf reagieren. Je weiter sich der Betrachter näherte, desto schneller pulsierten der Herzschlag und der Ikosaeder auf den Displays. Sobald die Distanz unter 40cm betrug, war ein blitzartiges Geräusch zu hören und die Displays flackerten. Dadurch konnte der Betrachter mit der Installation interagieren.

Abbildung 101 – Abstellraum

9.2 Das narrative Interview

Die Ausstellung beziehungsweise die qualitative Befragung wurde am 01.09.2018 an einem Abend abgehalten. Dabei stand der Ausstellungsraum den ganzen Abend zur freien Verfügung. Die externen Stakeholder konnten jederzeit ein- und austreten und sich solange wie sie wollten darin aufhalten. Als wissenschaftliche Methode wurde die qualitative Befragung gewählt, genauer die Form des narrativen Interviews.

> *„Im narrativen Interview wird von den Befragten eine Erzählung erwartet, in welcher einerseits die Orientierungsmuster ihres Handelns deutlich werden und zugleich rückblickend Interpretationen dieses Handelns erzeugt werden (Salat, 2011).“*

Dabei wird das Interview in vertrauter, freundschaftlicher Atmosphäre abgehalten. Die Interviewten sollen dabei zwanglos erzählen und es soll ihnen genug Raum für ihre Beschreibungen und Begründungen gegeben werden (Salat, 2011). Als Einleitung in das Thema beziehungsweise als Startschuss für das Interview, diente die Betrachtung und Anwesenheit der Installation. Dabei starteten die Interviewten oft selbst das Gespräch, eine Einführung in das Thema war aus Sicht des

Interviewers nicht notwendig. Es ergaben sich Interviews im Ausstellungsraum sowie auch außerhalb des Raumes und abseits der Installation.

Durch die Interviews konnten auf folgende Forschungsfragen Antworten gefunden werden:

- Welchen Mehrwert bietet die Bewegung für die Installation?
- Welche Anordnungen sind mit solchen Modulen möglich bzw. wirken besonders immersiv?
- Welche Interaktionsmuster eignen sich für Installationen dieser Art?
- Wie soll die Umgebung der Installation beschaffen sein?

An der qualitativen Befragung nahmen insgesamt 13 externe Stakeholder teil. Innerhalb von vier Stunden wurden 13 Interviews abgehalten. Die Dauer der Interviews reichte von 3min bis 20min. Die durchschnittliche Dauer der Interviews betrug 12min. Dabei kamen von den Teilnehmern Fragen zur Installation, neue Ideen, Verbesserungsvorschläge sowie Feedback, über das durch die Installation Erlebte. Die Kernaussagen dieser Erzählungen wurden vom Interviewer in Stichworten festgehalten. Im Abschnitt 9.3 bilden die Nacherzählungen und Interpretationen daraus die Antworten auf die Forschungsfragen.

Im Ausstellungsraum wurde den ganzen Abend über ein Video aufgezeichnet, welches die Interaktionen zwischen der Installation und der Benutzer enthält. Abbildung 102 und Abbildung 103 zeigen chronologisch geordnete Ausschnitte des Abends. Diese Erhebung dient dem Interviewer als zusätzliche Perspektive. Damit soll die Möglichkeit gegeben sein, Aussagen aus den Interviews mit den Handlungen abzugleichen, um so eine vernünftigere Interpretation zu ermöglichen. Dem Leser dieser Arbeit sollen diese Ausschnitte eine bessere Vorstellung über das Setting der Ausstellung und der narrativen Interviews geben.

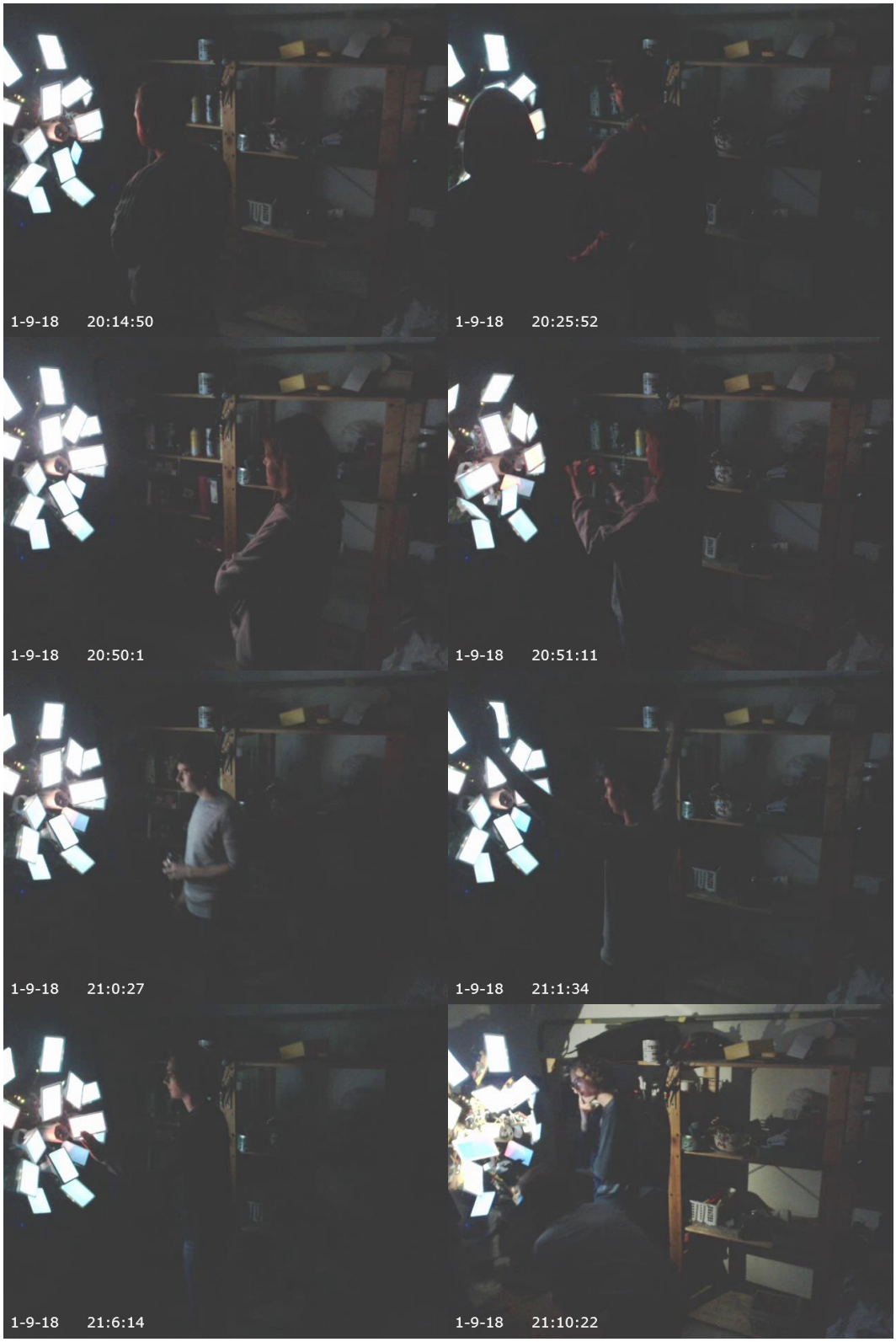


Abbildung 102 – Ausschnitte der Ausstellung 1

Abbildung 103 – Ausschnitte der Ausstellung 2

9.3 Ergebnisse

Diese Ergebnisse bestehen aus der Zusammenfassung der in den Interviews besprochenen Punkte, sowie Interpretationen daraus. Die Interviewten äußerten zur Installation Fragen, Assoziationen, Bemerkungen sowie Verbesserungsvorschläge. All diese Äußerungen dienen zur Beantwortung folgender Forschungsfragen:

Welchen Mehrwert bietet die Bewegung für die Installation?

Neben der Bewegung der einzelnen Module, lässt sich eine Gesamtbewegung, die sich über die Installation erstreckt, wahrnehmen. Einige der Interviewten beschreiben dies als „lebendig“ bzw. „es lebt“. Durch die Bewegung lässt sich also eine gewisse Lebendigkeit herbeiführen. Diese Lebendigkeit wurde vor allem auch mit den organischen Körpergeräuschen unterstützt.

Einer der Interviewten äußerte die Bemerkung, dass sich die Installation wie ein Hund verhalte. Je mehr man sich ihr nähert desto „aufgeregter“ ist sie. Diese wahrgenommene Aufregung, hängt mit der Erhöhung der Frequenz zusammen, mit der sich die Displays hin und her bewegen und mit der die Installation pulsiert.

Des Weiteren wurde der Vorschlag gemacht Sinusschwingungen über die Ausrichtungen der Module darzustellen. Dabei könnte die Rotation jedes Moduls die Werte eines Sinus abfahren. Wenn die Module zueinander phasenverschoben sind, könnte man über die gesamte Installation gleichmäßige Wellenbewegungen wahrnehmen.

Welche Anordnungen sind mit solchen Modulen möglich bzw. wirken besonders immersiv?

Mit einem Durchmesser von etwa 80cm bildet diese Installation ein gutes Gegenstück zur Größe eines Menschen. Mehrere Interviewte erwähnten die imposante Form und Größe. Der Vorteil der bei dieser Installation gewählten Form des Zapfens, ist die sternförmige Anordnung der Displays zueinander. Dadurch können alle Displays zum Zapfenzentrum gedreht werden. Wenn sich der Betrachter nähert, kann sich die Installation also um ihn wölben, wodurch ein immersiveres Erlebnis ermöglicht wird. Jedoch ist diese Anordnung nur eine Möglichkeit von vielen.

Es wurde der Vorschlag hervorgebracht und diskutiert, diese Installation an der Wand innerhalb eines Containers oder einer Halle größer umzusetzen. Aufgrund der jetzigen Funktionsweise der Module, würde eine Skalierung ohne Probleme funktionieren, da in jedem Modul der notwendige Rechenaufwand stattfindet und

über das Netzwerk nur Steuersignale mit geringem Datenaufkommen geschickt werden müssen. Je größer diese Installation umgesetzt werden würde, desto imposanter und einschüchternder würde diese wirken.

Ein weiterer Vorschlag war mit den Modulen die Form einer Kugel nachzubauen. Dadurch könnte der Betrachter um diese herumgehen, um diese von allen Seiten zu betrachten. So könnten beispielsweise Abbildungen des Erdballs dargestellt werden.

Eine andere Idee wäre die Umsetzung in einer Kuppel, sodass sich der Betrachter innerhalb der Kugel befinden würde. Diese Displayoberfläche würde den Betrachter umhüllen. Dadurch könnten Betrachter innerhalb einer lebendigen, beweglichen Oberfläche 360° Videos erleben.

Welche Interaktionsmuster eignen sich für Installationen dieser Art?

Bei der Ausstellung reagierte die Installation auf die Nähe des Betrachters. Je näher die Person kam, desto schneller pulsierte sie. Bei starker Annäherung gab die Installation Stromschlaggeräusche von sich und blitzte visuell.

Die externen Stakeholder waren ständig auf der Suche nach Interaktionsmustern. Oft wurden Entdeckungen bezüglich Zusammenhänge weitererzählt und anderen erklärt. Zwei Interviewte meinten, dass die Interaktionszusammenhänge stärker und schneller ausgeprägt sein könnten. Ihnen fiel es schwer Zusammenhänge zu finden.

Zwei andere Interviewte waren auf der Suche nach der Verbindung zum Puls. Sie stellten Überlegungen an, ob die Installation ihren Herzschlag messen könne und diesen auf sich selbst überträgt. Sie erzählten auch, dass ihr Herzschlag höher wurde, sobald die Installation ihre Frequenz erhöhte. Dies spricht für eine enge Verbindung und Interaktion zwischen Benutzer und Installation. Dies hängt vor allem auch mit den körpernahen Sounds zusammen.

Zwei weitere Interviewte erzählten davon, das Stromschlaggeräusch als Fehler beziehungsweise Störung der Installation wahrgenommen zu haben und dass sie dadurch erschrocken wären. Zwei andere Personen ließen sich intensiver mit der Interaktion des Stromschlages ein. Durch Annäherung mit der Hand an den Ultraschallsensor, lösten sie dieses Event aus. Sie beschrieben, dass sie ein wenig den Stromschlag fühlen konnten, obwohl dieses Event nur durch Sound und Bild stattfand. Dies spricht für eine Verbindung aller Sinne und die Möglichkeit eines immersiven Erlebnisses.

Während eines Interviews kam die Idee auf, die Displays auf den Kopf des Benutzers auszurichten. Für solch eine Umsetzung wäre ein Tracking des Kopfes notwendig. Dies könnte beispielsweise mit einer Kinect[45] erfolgen. Dadurch wäre es mit Hilfe der Position des Kopfes möglich, die Rotationen der Displays zu steuern. So könnten sich die Displays immer in Richtung Betrachter drehen.

Wie soll die Umgebung der Installation beschaffen sein?

Zu Beginn dieser Arbeit wurde die Installation innerhalb einer Black Box gesehen. Dadurch würde die Illusion von schwebenden Displays entstehen, da sonst keine Anhaltspunkte zu sehen wären. Der Abstellraum in der die Installation ausgestellt wurde, hat weiße Wände und eine weiße Decke. Ein Interviewter erzählte von den beeindruckenden Lichtspielen, die durch die bewegten Displays an der Decke zu sehen waren. Er verglich diese mit den Lichtmustern, die durch die Reflexionen an einer Wasseroberfläche entstehen.

Die Umgebung kann deshalb sehr in die Inszenierung der Installation miteinbezogen werden, da das abgestrahlte Licht den Raum erweitert und die Installation so vergrößert. Die verwendeten Displays weisen auf deren Rückseite eine silbern reflektierende Oberfläche auf. Mit Hilfe eines steuerbaren DMX-RGB Scheinwerfers könnten diese Oberflächen auf der Rückseite angeleuchtet werden. Dadurch würden sich auf der Rückwand des Raumes die bewegten Reflexionen abbilden. So könnte die Umgebung als weiteres Gestaltungselement miteinbezogen werden und die gesamte Wirkung der Installation maximieren.

Technische Verbesserungsvorschläge

Ein Interviewter hatte einen technischen Verbesserungsvorschlag, um die Steuerdaten der Installation zu reduzieren. Im Moment werden kontinuierlich Werte gesendet, die die Position der Displays und das Aussehen des Contents verändern. Wenn ein Display beispielsweise in Form eines Sinus hin und her schwingen soll, wird am Laptop ein Sinus erzeugt und die Werte kontinuierlich dem Raspberry Pi gesendet. Um die zu sendenden Daten zu reduzieren, könnte der Sinus am Raspberry Pi erzeugt werden. Vom Laptop aus würden dann nur Parameter gesendet werden, die den Sinus beschreiben, wie Frequenz, Amplitude und Phasenverschiebung. Dadurch würden sich die Daten, die vom Laptop erzeugt und übers Netzwerk zu den Raspberry Pis gesendet werden, erheblich reduzieren. Im Falle einer Skalierung der Modulanzahl, wäre dies eine schlaue Überlegung.

[45] https://developer.microsoft.com/de-de/windows/kinect


transX transZ rot grün blau alpha
0 -500.0 255 255.0 255.0 120 315.350
audiofilter2
null2

10 Fazit

In dieser Arbeit wurden, in enger Zusammenarbeit mit den externen Stakeholdern, Module entwickelt, welche zum Bau von kinetischen, bildgebenden Körpern verwendet werden können. Dafür wurden mit der Methode Presumptive Design insgesamt 10 Engagement Sessions abgehalten, an denen insgesamt 26 externe Stakeholder teilnahmen.

All die Erfahrungen, der darin entwickelten Prototypen und angefertigten Skizzen, flossen in den Bau der Installation mit ein. Darin wurden vier Module 1 sowie zehn Module 2 in einer künstlerischen Arbeit zu einer gemeinsamen Installation zusammengesetzt. Dafür wurde mit unterschiedlichster Software ein System erschaffen, mit dem es möglich ist von einem Punkt aus, alle Module zu steuern.

All diese erarbeiteten Punkte wurden in einen Bauplan überführt, sodass Module, Teile der Installation oder die ganze Installation nachgebaut werden können. Die erstellten Files sind auf GitHub[46] zur Verfügung gestellt worden, die Modelle für den 3D Druck auf Thingiverse[47].

Diese Arbeit wurde im Zuge einer Ausstellung den externen Stakeholdern präsentiert. Dabei wurden 13 qualitative Befragungen in Form von narrativen Interviews durchgeführt. Mit diesen gewonnenen Informationen konnten Antworten auf die ausstehenden Forschungsfragen gefunden werden.

Welche Umsetzung bzw. Komponenten stellen sich als kostengünstig aber trotzdem zweckerfüllend heraus?

Diese Frage findet Schritt für Schritt im Laufe der Arbeit Antwort. Die verwendeten Komponenten werden in Tabelle 11 aufgelistet. In allen Kapiteln der Arbeit wird auf die Zeck Erfüllung der Komponenten Bezug genommen.

Dabei erfüllten fast alle Komponenten ihren Zweck. Die Servomotoren stellten sich als sehr zuverlässig und kostengünstig heraus. Die 3D gedruckten Displayhalterungen sind auch robust genug und erfüllen ihren Zweck. Die Displays sind mit einer Auflösung von 1024 mal 600 auch zweckerfüllend.

[46] https://github.com/fabwald-vienna/Rosette-4.0

[47] https://www.thingiverse.com/fabwald/designs

Lediglich die Linear-Aktuatoren bringen bei deren Verwendung mehr Nachteile als Vorteile mit sich. Mit einer Geschwindigkeit von 7mm pro Sekunde ist dieses Modell sehr langsam. Dadurch kann dieser nur für leichte Formänderungen eingesetzt werden und nicht für impulsartige Bewegungen. Dafür wären professionellere Modelle erforderlich, die wiederum nicht kostengünstig sind.

Außerdem entsteht bei der Verwendung der Linear-Aktuatoren ein enormer Mehraufwand. Das Servokabel, sowie das Flachbandkabel fürs Display müssen so verlegt werden, dass sie mit der linearen Bewegung mitgeführt werden und durch sie nicht beschädigt werden. Des Weiteren wird durch die Verwendung des Linear-Motors eine externe Stromversorgung notwendig, was bei einer Installation mit 16 Modulen einen enormen Aufwand darstellen würde. Für die Steuerung wird ein leistungsfähigeres und dadurch teureres Raspberry notwendig.

Eine empfehlenswerte Umsetzungsform stellt das Modul 2 dar. Es kann mit einem Raspberry Pi Zero W und ohne externe Stromversorgung gebaut werden und fällt dadurch günstiger aus. Durch die schnelle Rotation des Servos können mit diesem Modul sehr immersive Installationen gestaltet werden. Aufgrund des kleineren Raspberrys und der wenigen Komponenten kann dieses Modul platzsparender montiert werden.

Welchen Mehrwert bietet die Bewegung für die Installation?

Diese Frage wird ausführlicher in Abschnitt 9.3 beantwortet. Allgemein lässt sich sagen, dass durch die Bewegung der Installation die Lebendigkeit dieser steigt. Der ganze Aufbau wirkt dadurch organischer und erweckt den Eindruck von Leben.

Welche Anordnungen sind mit solchen Modulen möglich bzw. wirken besonders immersiv?

Diese Frage wird ebenfalls in Abschnitt 9.3 ausführlicher beantwortet. Für die Umsetzung der Installation wurde die Form einer Blume beziehungsweise die eines Zapfens gewählt. Dadurch können alle Blätter ins Zentrum, also in Richtung der Blüte, gedreht werden. Diese Anordnung stellte sich als sehr immersiv und wandelbar heraus. In Abschnitt 9.3 werden auch andere Möglichkeiten der Anordnung aufgelistet, prinzipiell ist aber durch die wandelbare Bauform der Module jede fast erdenkliche Anordnung möglich.

Welche Interaktionsmuster eignen sich für Installationen dieser Art?

Aus den Auswertungen der qualitativen Befragungen stellte sich heraus, dass die Benutzer ständig auf der Suche nach Zusammenhängen zwischen ihren Handlungen und den Aktionen der Installation waren. Abschnitt 9.3 geht näher

darauf ein. Es lässt sich auch sagen, dass diese Interaktionsmuster die Charakteristiken und den Charme der Installation ausmachen. Die Interaktionsmuster können also zur Gestaltung der Charakteristiken der Installation verwendet werden. Für eine qualitativere Form der Interaktion ist die Notwenigkeit eines genaueren Trackingsystems gegeben.

Wie soll die Umgebung der Installation beschaffen sein?

Diese Frage wird auch im Abschnitt 9.3 näher behandelt. Grundsätzlich lässt sich sagen, dass die Umgebung, anders als anfangs vermutet, enorm zur Inszenierung der Installation eingesetzt werden kann. Das Leuchten der Displays verursacht bewegte Lichtflecken an den Wänden sowie der Decke. Durch die reflektierende Rückseite der Displays, können diese ebenso als kinetische Reflektoren eingesetzt werden und somit die Rückwand gestaltet.

Presumptive Design

Dadurch, dass ständig neue Artefakte den externen Stakeholdern vorgelegt wurden, konnten diese, mögliche Probleme aus unterschiedlichsten Perspektiven betrachten und Ideen dazu einbringen. Ganz nach dem Zitat der Autoren:

> *„Without this “surveying of the land” we won’t know what other hills there are that could have been climbed“ (Frishberg & Lambdin, 2015, S. 73).*

Oft wurden die Artefakte schon zu detailliert ausgeführt. Einerseits könnte der Aufwand eingespart werden, andererseits bleibt dem externen Stakeholder mehr Platz um dieses Produkt mitzugestalten. Denn genau ausgearbeitete Artefakte haben die Anmutung eines fertigen Produkts. Ungenaue lassen noch viel Platz zur weiteren Entwicklung und sind billiger in der Herstellung (Frishberg & Lambdin, 2015, S. 73).

Während dem Entwicklungsprozess sind außerdem einige weitere Ideen angeführt worden, die nicht umgesetzt wurden. Diese können als Inspirationsquelle für neue ähnliche Projekte dienen.

Die in dieser Arbeit entwickelten Pläne, Programme sowie Ausführungen stellen ein stabiles Grundgerüst für den Bau von kinetischen, bildgebenden Körpern da. Darin wurde ein System entwickelt, mit dem auf den Clients der Content für die Displays errechnet werden kann. Somit kann diese Installation leicht skaliert werden.

Weiterführende Forschungen in diesem Gebiet könnten sich mit dem erleichterten Handling der Processing Sketches beschäftigen. Beispielsweise könnte an der Steuereinheit ein Ordner eingerichtet werden, auf den alle Clients zugreifen. Dadurch könnten 3D-Objekte oder Processing Sketches geladen werden. So würde das mühsame bespielen aller Clients wegfallen.

Ein weiteres Forschungsgebiet eröffnet sich mit der Steuerung der Interaktion. Beispielsweise könnten die Bewegungen der virtuellen Kameras mit den physikalischen Bewegungen der Module abgeglichen werden. Dadurch würden verschiedene Illusionen entstehen. Im Falle eines Kopf-Trackings könnte die Perspektive des Benutzers ebenfalls in diese Perspektiven miteinbezogen werden.

Ein praktisches Forschungsgebiet stellt die Weiterentwicklung der Module sowie der Konstruktion der Installation dar. Beispielsweise könnten Module in fixen Formen verbaut werden, mit denen diese aneinandergesteckt werden könnten und dadurch schnell kinetische, bildgebende Körper erzeugt werden. Würden diese Positionen der Module noch automatisch ermittelt werden können, wäre ein Einmappen nicht mehr notwendig. Dadurch könnte der Körper beliebig ohne weiteren Aufwand gestaltet werden können.

Zum Schluss, ein herzliches Dankeschön für die Unterstützung von allen Seiten!

Literaturverzeichnis

2ndviolinist. (2013). *Alexeieff's Night on Bald Mountain (great sound)*. Abgerufen von https://www.youtube.com/watch?v=wYbjW7XrWDo

52Pi. (2018). 7-Inch-1024x600 Display Kit (without Touch Screen) SKU:Z-0051 - 52Pi Wiki. Abgerufen 7. Mai 2018, von https://wiki.52pi.com/index.php/7-Inch-1024x600_Display_Kit_(without_Touch_Screen)_SKU:Z-0051

Ali. (2018). Servo AS3103 - AliExpress. Abgerufen 8. Mai 2018, von //de.aliexpress.com/item/4-2kg-torque-analog-servo-easy-FPV-Pan-Tilt-Camera-Platform-Camera-Mount-For-Aircraft-FPV/1942923446.html?src=ibdm_d03p0558e02r02

Autobotic. (2018). AS3103 Continuous Servo - 360deg. Abgerufen 4. September 2018, von https://www.autobotic.com.my/as3103-continuous-servo-360deg

Baustoff. (2018). MEA Lichtschacht Gitterrost begehbar b=100 cm - t=40 cm | mein-baustoffshop24.de. Abgerufen 19. April 2018, von https://www.mein-baustoffshop24.de/mea-lichtschacht-gitterrost-begehbar-100x40cm

Bundulis, R., & Arnicans, G. (2013). Architectural and technological issues in the field of multiple monitor display technologies. *Frontiers in Artificial Intelligence and Applications*, *249*, 317–329. https://doi.org/10.3233/978-1-61499-161-8-317

Bundulis, R., & Arnicans, G. (2014). Concept of virtual machine based high resolution display wall (S. 1–6). IEEE. https://doi.org/10.1109/AIEEE.2014.7020317

Claire, P. (1937). *US2100148A*. United States. Abgerufen von https://patents.google.com/patent/US2100148/en

Club-3D. (2018). Club 3D | Multi Stream Transport (MST) Hub DisplayPort™ 1.2 Quad Monitor. Abgerufen 6. Mai 2018, von http://www.club-3d.com/de/detail/2411/multi_stream_transport_(mst)_hub_displayportt_1.2_quad_monitor/

Daniel Rozin Interactive Art. (2018). Abgerufen 28. Februar 2018, von http://www.smoothware.com/danny/

Dejan. (2015, Juli 26). Ultrasonic Sensor HC-SR04 and Arduino Tutorial. Abgerufen 8. September 2018, von https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

Elek, F. (2018). HC-SR04 Datasheet.

Facettenauge. (2018). In *Wikipedia*. Abgerufen von https://de.wikipedia.org/w/index.php?title=Facettenauge&oldid=176226826

Frishberg, L., & Lambdin, C. (2015). *Presumptive Design: Design Provocations for Innovation*. Morgan Kaufmann.

Fry, B., & Reas, C. (2018a). Icosahedra \ Examples \ Processing.org. Abgerufen 4. September 2018, von https://processing.org/examples/icosahedra.html

Fry, B., & Reas, C. (2018b). Processing.org. Abgerufen 4. September 2018, von https://processing.org/

Hannover, B. S. in. (2014). *Facettenauge - Andreas Rimkus*. Abgerufen von https://commons.wikimedia.org/wiki/File:2014-05-17_Fahrradtour_Freundeskreis_Hannover_Maschsee_Edelhof_Ricklingen_Park_der_Sinne,_(251)_Insektenauge_(Andreas_Rimkus).jpg

Heldmaier, G., & Neuweiler, G. (2013). *Vergleichende Tierphysiologie: Neuro- und Sinnesphysiologie*. Springer-Verlag.

Henning. (2018). Router Test 3HuiTube | LTE + WLAN Router von DREI Austria im Test. Abgerufen 7. September 2018, von https://www.fts-hennig.at/netztechnik/blog/test-3hui-tube-lte-router/

Hickman, C. P. (2008). *Zoologie*. Pearson Deutschland GmbH.

iart. (2014). The Kinetic Facade of the MegaFaces Pavilion - Sochi 2014 Winter Olympics - Projects - iart.ch. Abgerufen 5. März 2018, von https://iart.ch/en/-/die-kinetische-fassade-des-megafaces-pavillons-olympische-winterspiele-2014-in-sotschi

Kinect. (2018). Image Based Motion Analysis with Kinect V2 and OpenCV – Microsoft Faculty Connection. Abgerufen 20. April 2018, von https://blogs.msdn.microsoft.com/uk_faculty_connection/2017/05/15/image-based-motion-analysis-with-kinect-v2-and-opencv/

Kintel, M. (2018a). OpenSCAD - About. Abgerufen 26. März 2018, von http://www.openscad.org/about.html

Kintel, M. (2018b, Jänner 14). OpenSCAD User Manual/First Steps/Changing the colour of an object - Wikibooks, open books for an open world. Abgerufen 26. März 2018, von https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/First_Steps/Changing_the_colour_of_an_object

Kofler, Kuhnast, & Scherbeck. (2015). *Raspberry Pi Das umfassende Handbuch*. Rheinwerk Verlag GmbH. Abgerufen von http://gen.lib.rus.ec/book/index.php?md5=6a2bc07ad94372f0ca388b7a57dc5b4b

Lytro. (2018). Lytro Support. Abgerufen 1. Mai 2018, von https://support.lytro.com/hc/de

Mundo Reader S.L. (2018). BQ Witbox 2 3D-Drucker | BQ. Abgerufen 26. März 2018, von https://www.bq.com/de/witbox-2

Nachtigall, W., & Wisser, A. (2013). *Bionik in Beispielen: 250 illustrierte Ansätze*. Springer-Verlag.

Nagelbrett. (2018). Nagelbrett - Amazon. Abgerufen 19. April 2018, von https://www.amazon.de/Relaxdays-Nagelbrett-Pinpression-Abbildungen-Metallstiften/dp/B00DFW8W78

Neupert, R. (2011). *French Animation History*. John Wiley & Sons.

Otamat. (2017). eBay. Abgerufen 8. Mai 2018, von http://vi.vipr.ebaydesc.com/ws/eBayISAPI.dll?ViewItemDescV4&item=282363055991&t=1514840506000&tid=310&category=9723&seller=otas32&excSoj=1&excTrk=1&lsite=3&ittenable=false&domain=ebay.co.uk&descgauge=1&cspheader=1&oneClk=1&secureDesc=0

Otamat. (2018). Linear Actuator Stepper Motor 4 Arduino, Driver, Raspberry PI micro OTAMAT UK. Abgerufen 8. Mai 2018, von https://www.ebay.co.uk/itm/Linear-Actuator-Stepper-Motor-4-Arduino-Driver-Raspberry-PI-micro-OTAMAT-UK-/282363055991

Park der Sinne. (2018). 17 Insektenauge. Abgerufen 21. Jänner 2018, von http://www.verein-park-der-sinne.de/17-insektenauge.html

Plaugic, L. (2018, Jänner 9). Daniel Rozin's interactive sculpture is made of 832 tiles that respond to movement. Abgerufen 28. Februar 2018, von https://www.theverge.com/2018/1/9/16850280/daniel-rozin-interactive-art-interview-video-nespresso-last-chance-to-shine

Raspberry, P. (2018). Raspberry Pi 2 Model B. Abgerufen 8. Mai 2018, von https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

Salat, E. H. und J. (2011). Das narrative Interview. Abgerufen 6. September 2018, von https://www.univie.ac.at/ksa/elearning/cp/qualitative/qualitative-46.html

Schlegel, A. (2010). oscP5. Abgerufen 4. September 2018, von http://www.sojamo.de/libraries/oscp5/

Schulz, K. (2010). *Being Wrong: Adventures in the Margin of Error*. HarperCollins. Abgerufen von http://gen.lib.rus.ec/book/index.php?md5=DA5F4BE1ECE21B0F09975AE1E9290E99

Selvedge. (2018). Pin Up. Abgerufen 12. April 2018, von https://www.selvedge.org/blogs/selvedge/pins-and-needles

Spinger, C. (2016, Oktober 13). Oculus Touch Controller bei Amazon und Co. vorbestellen. Abgerufen 15. Mai 2018, von https://www.vrnerds.de/oculus-touch-controller-bei-amazon-und-co-vorbestellen/

StackExchange. (2014). gpio - Control Hardware PWM frequency. Abgerufen 16. Mai 2018, von https://raspberrypi.stackexchange.com/questions/4906/control-hardware-pwm-frequency/9725

Ultimaker. (2018). Ultimaker Cura 3D Printing Software | Ultimaker. Abgerufen 26. März 2018, von https://ultimaker.com/en/products/ultimaker-cura-software

wiringpi. (2018a). Setup | Wiring Pi. Abgerufen 4. September 2018, von http://wiringpi.com/reference/setup/

wiringpi. (2018b). WiringPi. Abgerufen 4. September 2018, von http://wiringpi.com/

Abbildungsverzeichnis

Tabellenverzeichnis

Codeverzeichnis

Anhang

A. Hardware

displayhalterung_version_3.scad

```
difference(){

    translate([-50,0,0]) //172 / 2 + 37 = 123
    cube([20,107,8]);    //QuerHalterung


    translate([-50,0,5])
    cube([20,3,3]);    //Seitenausschnitt1

    translate([-50,104,5])
    cube([20,3,3]);    //Seitenausschnitt1

    translate([-40,53.5,5])
    rotate([0,0,0])
    cylinder(r=3,h=10,center=true);

}

translate([0,0,6])
cube([9,107,3]);    //EinschubHalterungAusschnitt


difference(){
    cube([172,107,29]); //Rohling

    translate([3,3,3])
    cube([166,101,40]); //HauptAusschnitt

    translate([9,11,0])
    cube([157,89,20]); // DisplayAusschnitt

    translate([0,0,9])
    cube([98,107,20]); // SeitenAusschnitt1

    translate([152,0,9])
    cube([103,107,20]); // SeitenAusschnitt2

    translate([98,0,9])
    rotate([0,-45,0])
    cube([30,107,30]);    //45GradAusschnitt1

    translate([152,0,9])
    rotate([0,-45,0])
```

```
    cube([30,107,30]);    //45GradAusschnitt2

    translate([123,0,9]) //172 / 2 + 37 = 123
    cube([5,107,20]);    //QuerHalterungAusschnitt
}
```

displayhalterung_version_4.scad

```
translate([140,100,0])
rotate([0,0,90])
difference(){

    translate([-50,0,0]) //172 / 2 + 37 = 123
    cube([20,107,8]);    //QuerHalterung


    translate([-50,0,5])
    cube([20,3,3]);    //Seitenausschnitt1

    translate([-50,104,5])
    cube([20,3,3]);    //Seitenausschnitt1

    translate([-40,53.5,5])
    rotate([0,0,0])
    cylinder(r=3,h=10,center=true);

    translate([-50,31,5])
    cube([20,45,3]);    //ServoAusschnitt
}


translate([0,0,-1]) //von 3mm auf 2mm
difference(){
    translate([0,0,6])
    cube([20,107,3]);    //EinschubHalterungAusschnitt

    translate([20,0,6])
    rotate([0,0,45])
    cube([15,107,3]);

    translate([0,91,6])
    rotate([0,0,-45])
    cube([15,107,3]);

    translate([0,20,6])
    rotate([0,0,0])
    cube([50,71,3]);
}

translate([0,0,-1]) //von 3mm auf 2mm
difference(){
    translate([0,0,1])
    cube([171,107,28]); //Rohling
```

```
    translate([3.5,3.5,3])
    cube([165.5,100.5,40]); //HauptAusschnitt

    translate([10,12,0])
    cube([156,88,20]); // DisplayAusschnitt

    translate([0,5,9])
    cube([98,107,20]); // SeitenAusschnitt1.1

    translate([0,0,9])
    cube([60,107,20]); // SeitenAusschnitt1.2

    translate([152,0,9])
    cube([103,107,20]); // SeitenAusschnitt2

    translate([98,5,9])
    rotate([0,-45,0])
    cube([30,107,30]);    //45GradAusschnitt1.1

    translate([60,0,9])
    rotate([0,-70,0])
    cube([30,107,30]);    //45GradAusschnitt1.2

    translate([152,0,9])
    rotate([0,-45,0])
    cube([30,107,30]);    //45GradAusschnitt2

    translate([123,0,9]) //172 / 2 + 37 = 123
    cube([5,107,20]);    //QuerHalterungAusschnitt

    translate([70,2,9])
    rotate([0,0,0])
    cube([36,2,20]);    //ConnectorAusschnitt
}
```

displayhalterung_version_5.scad

```
translate([140,100,0])
rotate([0,0,90])
difference(){

    translate([-50,0,0]) //172 / 2 + 37 = 123
    cube([20,107,8]);    //QuerHalterung


    translate([-50,0,5])
    cube([20,3,3]);    //Seitenausschnitt1

    translate([-50,104,5])
    cube([20,3,3]);    //Seitenausschnitt1

    translate([-40,53.5,5])
```

```
    rotate([0,0,0])
    cylinder(r=3,h=10,center=true);

    translate([-50,31,5])
    cube([20,45,3]);    //ServoAusschnitt
}


translate([0,0,-1]) //von 3mm auf 2mm
difference(){
    translate([0,0,6])
    cube([20,107,3]);    //EinschubHalterungAusschnitt

    translate([20,0,6])
    rotate([0,0,45])
    cube([15,107,3]);

    translate([0,91,6])
    rotate([0,0,-45])
    cube([15,107,3]);

    translate([0,20,6])
    rotate([0,0,0])
    cube([50,71,3]);
}

translate([0,0,-1]) //von 3mm auf 2mm
difference(){
    translate([0,0,1])
    cube([171,107,28]); //Rohling

    translate([3.5,3.5,3])
    cube([165.5,100.5,40]); //HauptAusschnitt

    translate([10,12,0])
    cube([156,88,20]); // DisplayAusschnitt

    translate([0,5,9])
    cube([98,107,20]); // SeitenAusschnitt1.1

    translate([0,0,9])
    cube([60,107,20]); // SeitenAusschnitt1.2

    translate([152,0,9])
    cube([103,107,20]); // SeitenAusschnitt2

    translate([98,5,9])
    rotate([0,-45,0])
    cube([30,107,30]);    //45GradAusschnitt1.1

    translate([60,0,9])
    rotate([0,-70,0])
```

```
    cube([30,107,30]);    //45GradAusschnitt1.2

    translate([152,0,9])
    rotate([0,-45,0])
    cube([30,107,30]);    //45GradAusschnitt2

    translate([123,0,5.5]) //172 / 2 + 37 = 123
    cube([5,107,23.5]);    //QuerHalterungAusschnitt

    translate([70,2,9])
    rotate([0,0,0])
    cube([36,2,20]);    //ConnectorAusschnitt
}
```

B. Software

receive_stepper_und_servo9_wlan0.py

```
import socket
import RPi.GPIO as GPIO
import wiringpi
import netifaces as ni
import time

GPIO.setmode(GPIO.BCM) ##### Setup fuer Steppermotor
GPIO.setwarnings(False)
coil_A_1_pin = 24 # grau
coil_A_2_pin = 23 # violett
coil_B_1_pin = 17 # blau
coil_B_2_pin = 4 # gruen
taster_pin = 7 # gelb
servoPIN1 = 18 # gelb
pos=0
data2=0.0
delay=1
reftime=3200
reftimecount=0

data1 = 45                          # wert fuer servo
data2 = 7                           # wert fuer linear aktuator

wiringpi.wiringPiSetupGpio() ######## Setup fuer Servo (Hardware PWM)
wiringpi.pinMode(servoPIN1, 2)
wiringpi.pwmSetMode(wiringpi.PWM_MODE_MS)
wiringpi.pwmSetRange(1024)
wiringpi.pwmSetClock(375) ####### Einstellung der Richtigen PWM Puls
Laenge

varfalse = False
istref = False
```

```
# adjust if different
StepCount = 8
Seq = range(0, StepCount)
Seq[0] = [1,0,0,0]
Seq[1] = [1,1,0,0]
Seq[2] = [0,1,0,0]
Seq[3] = [0,1,1,0]
Seq[4] = [0,0,1,0]
Seq[5] = [0,0,1,1]
Seq[6] = [0,0,0,1]
Seq[7] = [1,0,0,1]

GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin, GPIO.OUT)
GPIO.setup(taster_pin, GPIO.IN, pull_up_down = GPIO.PUD_UP)
#GPIO.setup(servoPIN1, GPIO.OUT)

def maprange( a, b, s): ######### map funktion
    (a1, a2), (b1, b2) = a, b
    if s>a2:
        return b2
    elif s<a1:
        return b1
    else:
        return  b1 + ((s - a1) * (b2 - b1) / (a2 - a1))

def my_callback(channel): ##### Taster fuer Referenzierung
    if channel == taster_pin:
      if GPIO.input(taster_pin):
        ##print "Der Taster wurde geoeffnet."
        global pos
        pos=0
        global istref
        istref = True
        resetStep()
     # else:
        ##print "Der Taster wurde geschlossen."

def setStep(w1, w2, w3, w4):
    GPIO.output(coil_A_1_pin, w1)
    GPIO.output(coil_A_2_pin, w2)
    GPIO.output(coil_B_1_pin, w3)
    GPIO.output(coil_B_2_pin, w4)

def resetStep():
    GPIO.output(coil_A_1_pin, 0)
    GPIO.output(coil_A_2_pin, 0)
    GPIO.output(coil_B_1_pin, 0)
    GPIO.output(coil_B_2_pin, 0)
```

```
def forward(delay, steps):
    for i in range(steps):
        global pos
        pos+=1
        for j in range(StepCount):
            setStep(Seq[j][0], Seq[j][1], Seq[j][2], Seq[j][3])
            time.sleep(delay)

def backwards(delay, steps):
    for i in range(steps):
        global pos
        pos-=1
        for j in reversed(range(StepCount)):
            setStep(Seq[j][0], Seq[j][1], Seq[j][2], Seq[j][3])
            time.sleep(delay)

def referenzieren():
    global reftimecount
    global reftime
    global istref
    global varfalse
    global taster_pin
    global data2
    while reftimecount < reftime and istref == varfalse and
GPIO.input(taster_pin) == 0 and data2 == 0:
        #print('tut referenzieren')
        backwards(int(delay) / 1000.0, int(1))
        reftimecount = reftimecount + 1
        global inrangedata1
        global data1

        data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
        MESSAGE1,MESSAGE2 = data.split(",") #Stringteilung Rotation 0-
90; Linear 0-2800

        data1 = float(MESSAGE1)
        data2 = float(MESSAGE2)

        inrangedata1 = maprange((0,90),(25,66), data1)
        global servoPIN1
        wiringpi.pwmWrite(servoPIN1,int(inrangedata1))
    else:
        #print "Die Zeit ist abgelaufen."
        pos = 0
        istref = True
        resetStep()


GPIO.add_event_detect(taster_pin, GPIO.BOTH, callback=my_callback)

#p1 = GPIO.PWM(servoPIN1, 50) # 50Hz
#p1.start(4) # initialisierung
```

```
ni.ifaddresses('wlan0')
ip = ni.ifaddresses('wlan0')[ni.AF_INET][0]['addr']
#print ip

UDP_IP = str(ip)
UDP_PORT = 5560

sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

try:
    while reftimecount < reftime and istref == varfalse and
GPIO.input(taster_pin) == 0:
        #print('tut referenzieren')
        #print GPIO.input(taster_pin)
        backwards(int(delay) / 1000.0, int(1))
        reftimecount = reftimecount + 1
        #print reftimecount
    else:
        #print "Die Zeit ist abgelaufen."
        pos = 0
        istref = True

    while True:
        data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
        MESSAGE1,MESSAGE2 = data.split(",") #Stringteilung Rotation 0-
90; Linear 0-2800

        data1 = float(MESSAGE1)
        data2 = float(MESSAGE2)

        inrangedata1 = maprange((0,90),(25,66), data1)
        wiringpi.pwmWrite(servoPIN1,int(inrangedata1))
        #print inrangedata1

        if data2 > pos and data2 < 2801:
            forward(int(delay) / 1000.0, int(1))
        elif data2 < pos and data2 > 0:
            backwards(int(delay) / 1000.0, int(1))
        elif data2 == 0 and pos != 0:
            reftimecount = 0
            istref = False
            referenzieren()
        elif data2 == pos:
            resetStep()
        #else:
            #print('pos=', pos,'data2', data2)

except KeyboardInterrupt:
        GPIO.cleanup()
```

szene1.pde

```
import oscP5.*;
import netP5.*;
import damkjer.ocd.*;

float eyeX=0;
float eyeY=0;
float eyeZ=300;
float centerX=0;
float centerY=0;
float centerZ=0;
//float lightX=200;
//float lightY=120;
//float lightZ=140;
float upX;
float upY;
float upZ;
float rollen;
float neigen;
float schwenken;
float scale;
float transX;
float transY;
float transZ;
float rot;
float gruen;
float blau;
float alpha;
float rotierenX;
float rotierenY;
float rotierenZ;

Icosahedron ico1;

OscP5 oscP5;
NetAddress myRemoteLocation;

Camera camera1;
float cameraZ;

void setup() {
  exec("sudo", "python",
"/home/pi/receive_stepper_und_servo9_wlan0.py");
  oscP5 = new OscP5(this,12000);
  noCursor();
  fullScreen(P3D);
  ico1 = new Icosahedron(75);
  cameraZ = (height/2.0) / tan(PI*60.0/360.0);
  camera1 = new Camera(this, eyeX, eyeY, eyeZ,
                             centerX, centerY, centerZ,
                             0, -1, 0,
```

```
                              0.112851804, cameraZ/10, cameraZ*10);
  //0.0424744392 - 100cm
  //0.0847961745 - 200cm
  //0.112851804 - 75cm
  //0.168390157 - 50cm
}

void draw() {

  camera1.jump(eyeX, eyeY, eyeZ);
  camera1.aim(centerX, centerY, centerZ);
  tilt();
  pan();
  roll();
  camera1.feed();

  background(0);
  lights();
  //pointLight(255, 102, 126, lightX, lightY, lightZ);

  pushMatrix();
  translate(transX, transY, transZ);
  scale(scale);
  rotateX(rotierenX);
  rotateY(rotierenY);
  rotateZ(rotierenZ);
  noStroke();
  fill(rot,gruen,blau,alpha);
  ico1.create();
  popMatrix();

  stroke(255);
  noFill();

  //translate(0,0,-50);
  scale(scale*0.5);
  for(int i=-1000; i<1000; i+=2){
   line(i,-1000,i,1000);
  }
  for(int w=-1000; w<1000; w+=2){
   line(-1000,w,1000,w);
  }

  /*fill(100, 0, 0);
  noStroke();
  box(2100);*/
}

void roll(){
  float[] attitude = camera1.attitude();
  if(floor(degrees(attitude[2])-rollen) > 0){
    for(int i = 0; i < abs(floor(degrees(attitude[2])-rollen));i ++){
```

```
      camera1.roll(radians(-1));
    }
  }
  if(floor(degrees(attitude[2])-rollen) < 0){
    for(int i = 0; i < abs(rollen-floor(degrees(attitude[2])));i ++){
      camera1.roll(radians(1));
    }
  }
}

void tilt(){
  float[] attitude = camera1.attitude();
  if(floor(degrees(attitude[1])-neigen) > 0){
    for(int i = 0; i < abs(floor(degrees(attitude[1])-neigen));i ++){
      camera1.tilt(radians(-1));
    }
  }
  if(floor(degrees(attitude[1])-neigen) < 0){
    for(int i = 0; i < abs(neigen-floor(degrees(attitude[1])));i ++){
      camera1.tilt(radians(1));
    }
  }
}

void pan(){
  float[] attitude = camera1.attitude();
  if(floor(degrees(attitude[0])-schwenken) > 0){
    for(int i = 0; i < abs(floor(degrees(attitude[0])-schwenken));i
++){
      camera1.pan(radians(-1));
    }
  }
  if(floor(degrees(attitude[0])-schwenken) < 0){
    for(int i = 0; i < abs(schwenken-floor(degrees(attitude[0])));i
++){
      camera1.pan(radians(1));
    }
  }
}

void oscEvent(OscMessage theOscMessage) {
  if(theOscMessage.checkAddrPattern("/eyeX")==true)
  {
    eyeX = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/eyeY")==true)
  {
    eyeY = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/eyeZ")==true)
  {
    eyeZ = theOscMessage.get(0).floatValue();
```

```
}
if(theOscMessage.checkAddrPattern("/centerX")==true)
{
  centerX = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/centerY")==true)
{
  centerY = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/centerZ")==true)
{
  centerZ = theOscMessage.get(0).floatValue();
}
/*if(theOscMessage.checkAddrPattern("/lightX")==true)
{
  lightX = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/lightY")==true)
{
  lightY = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/lightZ")==true)
{
  lightZ = theOscMessage.get(0).floatValue();
}*/
if(theOscMessage.checkAddrPattern("/upX")==true)
{
  upX = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/upY")==true)
{
  upY = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/upZ")==true)
{
  upZ = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/rollen")==true)
{
  rollen = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/neigen")==true)
{
  neigen = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/schwenken")==true)
{
  schwenken = theOscMessage.get(0).floatValue();
}
if(theOscMessage.checkAddrPattern("/scale")==true)
{
  scale = theOscMessage.get(0).floatValue();
```

```
  }
  if(theOscMessage.checkAddrPattern("/transX")==true)
  {
    transX = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/transY")==true)
  {
    transY = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/transZ")==true)
  {
    transZ = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/rot")==true)
  {
    rot = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/gruen")==true)
  {
    gruen = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/blau")==true)
  {
    blau = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/alpha")==true)
  {
    alpha = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/rotierenX")==true)
  {
    rotierenX = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/rotierenY")==true)
  {
    rotierenY = theOscMessage.get(0).floatValue();
  }
  if(theOscMessage.checkAddrPattern("/rotierenZ")==true)
  {
    rotierenZ = theOscMessage.get(0).floatValue();
  }
  //theOscMessage.print();
}

class Dimension3D{
   float w, h, d;

   Dimension3D(float w, float h, float d){
     this.w=w;
     this.h=h;
     this.d=d;
  }
```

```
}

abstract class Shape3D{
  float x, y, z;
  float w, h, d;

  Shape3D(){
  }

  Shape3D(float x, float y, float z){
    this.x = x;
    this.y = y;
    this.z = z;
  }

  Shape3D(PVector p){
    x = p.x;
    y = p.y;
    z = p.z;
  }

  Shape3D(Dimension3D dim){
    w = dim.w;
    h = dim.h;
    d = dim.d;
  }

  Shape3D(float x, float y, float z, float w, float h, float d){
    this.x = x;
    this.y = y;
    this.z = z;
    this.w = w;
    this.h = h;
    this.d = d;
  }

  Shape3D(float x, float y, float z, Dimension3D dim){
    this.x = x;
    this.y = y;
    this.z = z;
    w = dim.w;
    h = dim.h;
    d = dim.d;
  }

  Shape3D(PVector p, Dimension3D dim){
    x = p.x;
    y = p.y;
    z = p.z;
    w = dim.w;
    h = dim.h;
    d = dim.d;
```

```
  }

  void setLoc(PVector p){
    x=p.x;
    y=p.y;
    z=p.z;
  }

  void setLoc(float x, float y, float z){
    this.x=x;
    this.y=y;
    this.z=z;
  }

  // override if you need these
  void rotX(float theta){
  }

  void rotY(float theta){
  }

  void rotZ(float theta){
  }

  // must be implemented in subclasses
  abstract void init();
  abstract void create();
}

class Icosahedron extends Shape3D{

  // icosahedron
  PVector topPoint;
  PVector[] topPent = new PVector[5];
  PVector bottomPoint;
  PVector[] bottomPent = new PVector[5];
  float angle = 0, radius = 150;
  float triDist;
  float triHt;
  float a, b, c;

  // constructor
  Icosahedron(float radius){
    this.radius = radius;
    init();
  }

  Icosahedron(PVector v, float radius){
    super(v);
    this.radius = radius;
    init();
  }
```

```
  // calculate geometry
  void init(){
    c = dist(cos(0)*radius, sin(0)*radius, cos(radians(72))*radius,
sin(radians(72))*radius);
    b = radius;
    a = (float)(Math.sqrt(((c*c)-(b*b))));

    triHt = (float)(Math.sqrt((c*c)-((c/2)*(c/2))));

    for (int i=0; i<topPent.length; i++){
      topPent[i] = new PVector(cos(angle)*radius, sin(angle)*radius,
triHt/2.0f);
      angle+=radians(72);
    }
    topPoint = new PVector(0, 0, triHt/2.0f+a);
    angle = 72.0f/2.0f;
    for (int i=0; i<topPent.length; i++){
      bottomPent[i] = new PVector(cos(angle)*radius,
sin(angle)*radius, -triHt/2.0f);
      angle+=radians(72);
    }
    bottomPoint = new PVector(0, 0, -(triHt/2.0f+a));
  }

  // draws icosahedron
  void create(){
    for (int i=0; i<topPent.length; i++){
      // icosahedron top
      beginShape();
      if (i<topPent.length-1){
        vertex(x+topPent[i].x, y+topPent[i].y, z+topPent[i].z);
        vertex(x+topPoint.x, y+topPoint.y, z+topPoint.z);
        vertex(x+topPent[i+1].x, y+topPent[i+1].y, z+topPent[i+1].z);
      }
      else {
        vertex(x+topPent[i].x, y+topPent[i].y, z+topPent[i].z);
        vertex(x+topPoint.x, y+topPoint.y, z+topPoint.z);
        vertex(x+topPent[0].x, y+topPent[0].y, z+topPent[0].z);
      }
      endShape(CLOSE);

      // icosahedron bottom
      beginShape();
      if (i<bottomPent.length-1){
        vertex(x+bottomPent[i].x, y+bottomPent[i].y,
z+bottomPent[i].z);
        vertex(x+bottomPoint.x, y+bottomPoint.y, z+bottomPoint.z);
        vertex(x+bottomPent[i+1].x, y+bottomPent[i+1].y,
z+bottomPent[i+1].z);
      }
      else {
```

```
        vertex(x+bottomPent[i].x, y+bottomPent[i].y,
z+bottomPent[i].z);
        vertex(x+bottomPoint.x, y+bottomPoint.y, z+bottomPoint.z);
        vertex(x+bottomPent[0].x, y+bottomPent[0].y,
z+bottomPent[0].z);
      }
      endShape(CLOSE);
    }

    // icosahedron body
    for (int i=0; i<topPent.length; i++){
      if (i<topPent.length-2){
        beginShape();
        vertex(x+topPent[i].x, y+topPent[i].y, z+topPent[i].z);
        vertex(x+bottomPent[i+1].x, y+bottomPent[i+1].y,
z+bottomPent[i+1].z);
        vertex(x+bottomPent[i+2].x, y+bottomPent[i+2].y,
z+bottomPent[i+2].z);
        endShape(CLOSE);

        beginShape();
        vertex(x+bottomPent[i+2].x, y+bottomPent[i+2].y,
z+bottomPent[i+2].z);
        vertex(x+topPent[i].x, y+topPent[i].y, z+topPent[i].z);
        vertex(x+topPent[i+1].x, y+topPent[i+1].y, z+topPent[i+1].z);
        endShape(CLOSE);
      }
      else if (i==topPent.length-2){
        beginShape();
        vertex(x+topPent[i].x, y+topPent[i].y, z+topPent[i].z);
        vertex(x+bottomPent[i+1].x, y+bottomPent[i+1].y,
z+bottomPent[i+1].z);
        vertex(x+bottomPent[0].x, y+bottomPent[0].y,
z+bottomPent[0].z);
        endShape(CLOSE);

        beginShape();
        vertex(x+bottomPent[0].x, y+bottomPent[0].y,
z+bottomPent[0].z);
        vertex(x+topPent[i].x, y+topPent[i].y, z+topPent[i].z);
        vertex(x+topPent[i+1].x, y+topPent[i+1].y, z+topPent[i+1].z);
        endShape(CLOSE);
      }
      else if (i==topPent.length-1){
        beginShape();
        vertex(x+topPent[i].x, y+topPent[i].y, z+topPent[i].z);
        vertex(x+bottomPent[0].x, y+bottomPent[0].y,
z+bottomPent[0].z);
        vertex(x+bottomPent[1].x, y+bottomPent[1].y,
z+bottomPent[1].z);
        endShape(CLOSE);

        beginShape();
```

```
        vertex(x+bottomPent[1].x, y+bottomPent[1].y,
z+bottomPent[1].z);
        vertex(x+topPent[i].x, y+topPent[i].y, z+topPent[i].z);
        vertex(x+topPent[0].x, y+topPent[0].y, z+topPent[0].z);
        endShape(CLOSE);
      }
    }
  }

  // overrided methods fom Shape3D
  void rotZ(float theta){
    float tx=0, ty=0, tz=0;
    // top point
    tx = cos(theta)*topPoint.x+sin(theta)*topPoint.y;
    ty = sin(theta)*topPoint.x-cos(theta)*topPoint.y;
    topPoint.x = tx;
    topPoint.y = ty;

    // bottom point
    tx = cos(theta)*bottomPoint.x+sin(theta)*bottomPoint.y;
    ty = sin(theta)*bottomPoint.x-cos(theta)*bottomPoint.y;
    bottomPoint.x = tx;
    bottomPoint.y = ty;

    // top and bottom pentagons
    for (int i=0; i<topPent.length; i++){
      tx = cos(theta)*topPent[i].x+sin(theta)*topPent[i].y;
      ty = sin(theta)*topPent[i].x-cos(theta)*topPent[i].y;
      topPent[i].x = tx;
      topPent[i].y = ty;

      tx = cos(theta)*bottomPent[i].x+sin(theta)*bottomPent[i].y;
      ty = sin(theta)*bottomPent[i].x-cos(theta)*bottomPent[i].y;
      bottomPent[i].x = tx;
      bottomPent[i].y = ty;
    }
  }

  void rotX(float theta){
  }

  void rotY(float theta){
  }
}
```