2021

paamestia

PROJEKT IN FACHPRAXIS 11.11.2020 BIS 12.5.2021

ANDRÉ MILLING, MAX SCHIEMANN, DOMINIC KOSIN UND ARVID RANDOW

FACHARBEIT Jahrgang 12 BGT19 - Herr Friedrichs und Herr Kitzig

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

1. EINLEITUNG

In dem Beruflichen Gymnasium Schwerpunkt Mechatronik werden neben den allgemeinen Fächern wie Mathe, Deutsch oder Englisch auch Inhalte und Kompetenzen im Bereich Mechanik, Elektrotechnik und Informatik vermittelt. Zusätzlich beinhaltet der Lehrplan die Umsetzung eines praktischen Projektes, in denen man die erworbenen Kompetenzen einfließen lässt.

2. PROJEKT

Ein Projekt ist ein zielgerichtetes, einmaliges Vorhaben, das aus einer Anzahl abgestimmter, gesteuerter Tätigkeiten besteht und durchgeführt werden kann. Dabei ist zu beachten, dass man bestimmte Vorgaben wie Zeit, Ressourcen (Finanzierung, Personal oder Produktionsbedingungen), Qualität und ein Ziel erreichen muss. Hilfreich ist hier die sogenannt S.M.A.R.T. Methode, um das Projekt in eine geordnete Bahn zu lenken.

S. Spezifisch	„Spezifisch“ steht für eine konkrete und unmissverständliche Aussage, die genau definiert, was wir erreichen wollen.
M. Messbar	„Messbar“ bedeutet, dass Kriterien festgelegt werden, an denen wir das Erreichen des Zieles Objektiv überprüfen können.
A. Attraktiv	„Attraktiv“ bedeutet, dass die Ziele von allen Beteiligten akzeptiert werden muss. Ziele müssen ambitioniert, jedoch nicht unerreichbar sein.
R. Realistisch	Das Ziel muss für uns machbar sein. Unsere Ziele müssen fordernd, aber nicht utopisch sein. Zu hoch gesteckte Ziele würde am Ende enttäuschen.
T. Terminiert	Das Projekt muss ein Ende, also ein Abgabetermin aufweisen.

Wir haben als Kriterium erhalten, dass das Gerät mechanische Komponenten, elektronisch steuerbar und eine Informationsverarbeitung benutzen muss. Das Budget soll 150€ nicht überschreiten.

3. PROJEKTIDEE

Unsere erste Idee war es, einen Flipper zu entwickeln, welches eine Themenwelt mit beweglichen Komponenten beinhalten sollte. Besonders Dominic war davon angetan, da er dadurch sein Portfolio in grafischen Gestaltungen erweitern konnte. Außerdem waren wir im Besitze eines 3D-Druckers. Des Weiteren sollte der Flipper bestimmte Aktionen bei bestimmten Bedingungen durchführen und sie auf einen Bildschirm grafisch untermalen. Erste Skizzen und Pläne waren schon vorhanden. Jedoch mussten wir des Projekts recht schnell verwerfen, da sich im Rahmen der Komponentensuche ergeben hat, dass der Kauf dieser schnell den Finanziellen Rahmen gesprengt hätte. Mechanische Flipper werden nur noch in weniger Stückzahl produziert. Deswegen sind sie eher Sammlerstücke geworden und die einzelnen Komponenten sind recht hochpreisig.

Beim einem weiteren, eher gemütlichen, Beisammensitzen mit „Erfrischungsgetränken“ ist uns die Idee einer Getränkemischmaschine gekommen.

4. GEPLANTE FUNKTION UND ZIELSETZUNG

Die Maschine soll aus einer Anzahl von Vorgefertigten Rezepten ein Mischgetränk autonom in ein Glas geben.

Dabei sind die Flaschen seitlich kopfüber aufgehängt, und mit einem Schlauchsystem an Magnetventilen angebracht. Die Magnetventile sollen durch ein Programm, welches auf einer kleinen Platine vorhanden ist, angesteuert werden. Die Flüssigkeit soll von den Ventilen in eine Mischbatterie geleitet werden. Eine Schlauchpumpe soll für ein Transport der Flüssigkeit sorgen. Die Flaschen sollen austauschbar sein. Eine Wasserflasche soll die Schläuche regelmäßig reinigen.

Auf einen Monitor soll eine visuelle Menüführung einen durch das Programm führen. Einige Taster sollen die Steuerung durch das Menü ermöglichen.

Man kann sich auch das Getränk manuell mischen.

5. MODULE (PROJEKTBESTANDTEILE)

5.1. PLANUNG

Da wir schon eine ungefähre Vorstellung hatten, welche Komponenten wir im Groben benötigen, mussten wir überlegen, wie wir das Gerät aufbauen wollen. Dabei wurde geplant, dass sich insbesondere die elektrischen Komponenten in einen separaten Kasten recht weit oben befinden sollen. Dadurch soll verhindert werden, dass im Fall von Auslaufen von Flüssigkeit die Elektronik Schaden nimmt. Die Flaschen werden seitlich an den mittleren Kasten untergebracht. Außerdem ist geplant, dass sich die Seiten-Komponenten als Flügel konzipiert werden, damit man durch das Aufklappen besser an die Flaschen und deren Schlauchverbindung

rankommt. Anhand einer ersten groben Skizze hat Dominic einen ersten Entwurf über das Aussehen des Gerätes entworfen, welches wir als Gruppe zugestimmt haben.

Daraufhin wurde eine erste technische Zeichnung angefertigt, in denen schon die ersten genauen Bemaßungen angegeben wurde.

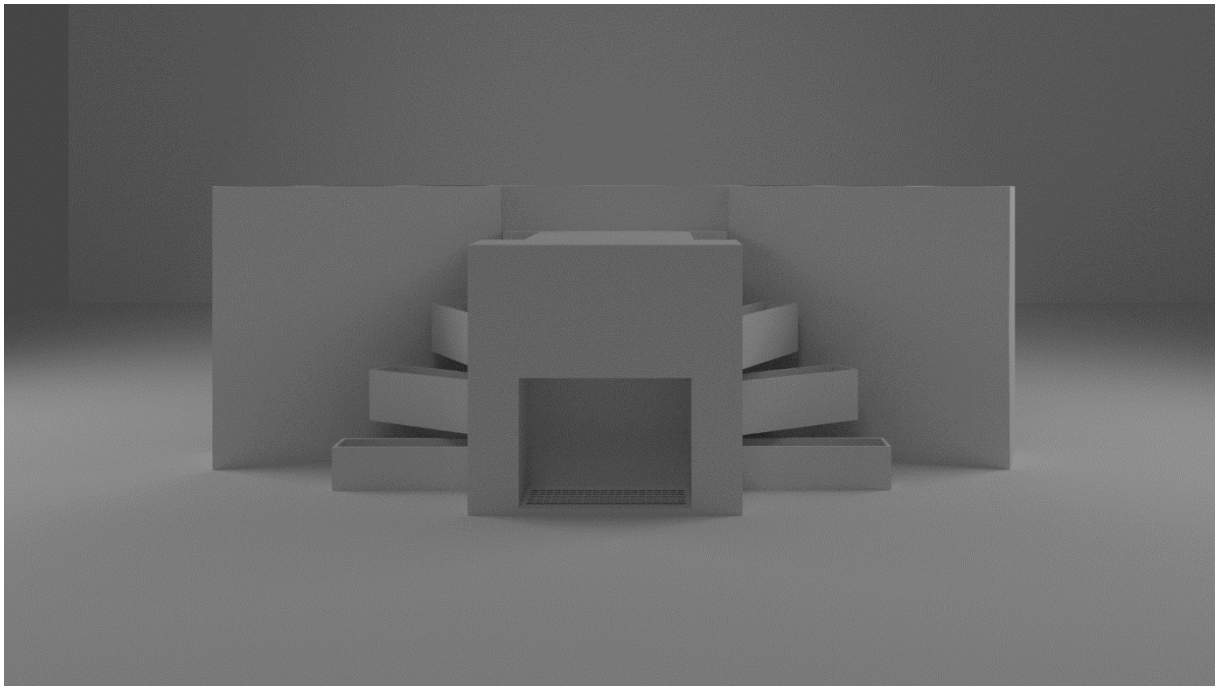

Abbildung 1: erstes Konzept

5.2. ERMITTLUNG DER KOMPONENTEN

Nachdem wir unseren ersten Konzepten grünes Licht gegeben haben, mussten wir uns überlegen welche Komponenten und Materialien für den weiteren Bau des Projektes benötigen. Dabei haben wir differenziert, für welche Bereiche wir die Materialien benötigen.

5.2.1. KONSTRUKTIONSBEZOGENE KOMPONENTEN

Unter konstruktionsbezogenen Komponenten verstehen wir alle Teile, die dafür sorgen, dass das geplante Gerät eine Stabilität erhält. Dazu gehört ein Konstruktionsrahmen, Verschalungen und andere Komponenten, die zur Befestigung anderer Bauteile benötigt werden.

Rahmen

Da wir planen, einiges an Gewicht in Form von Flaschen mit Flüssigkeiten in das Gerät aufzuhängen, haben wir beschlossen, dass die Konstruktion einen stabilen Rahmen erhalten soll. Als Material haben wir uns Aluminium ausgesucht, weil wir verhindern wollen, dass das Gerät zu schwer wird.

Nach erster Sichtung des Materials, welches uns die Schule zur Verfügung gestellt hat, sind wir zum Entschluss gekommen, dass die Menge nicht ausreichen wird, um den Rahmen zu vervollständigen. Also haben wir uns dazu entschieden Aluminiumprofile zu bestellen. Wir wählten „Aluminiumprofil 20x20 Nut 6 B-Typ“.

Abbildung 3: Aluminium-Profil

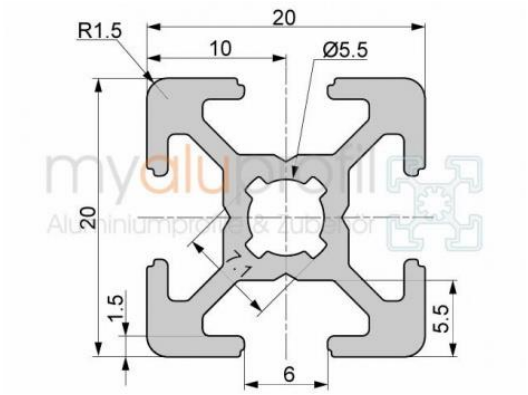

Abbildung 2: Aluminium-Profil im Querschnitt

Als Menge haben wir Profile mit 6 Meter Länge vorgesehen, dabei haben wir recht großzügig Verschnitt mit eingerechnet, da wir gerne im Falle von Fehlern genügend Profile zur Verfügung haben wollten.

Verschalung

Unsere Entscheidung, eine Verschalung aus Blechen zu konstruieren, hat unter anderem ästhetische Gründe. Wir planten einen abgeschlossenen Kasten, wo man das Innenleben nur zu sehen bekommt, wenn man die Seitenflügel aufklappt. Zusätzlich sollen die Bleche noch als Halterungen von anderen Komponenten wie das Tastenfeld oder Anschluss der Stromversorgung. Als Material wurde verzinktes Stahlblech vorgesehen. Dieses stellt uns die Schule zur Verfügung.

Sonstige Befestigungen

Da wir als Gruppe im Besitz eines privaten 3D-Druckers sind, planen wir Halterungen für die Elektronik und Mechanik auszudrucken. Das erspart es uns, zu komplexe Lösungsansätze bei der Befestigung zu konzipieren. Die Befestigungen werden an die Aluminiumprofile angebracht.

5.2.2. MECHANISCHE KOMPONENTE

Schlauchpumpe

Damit Flüssigkeiten befördert werden kann, benötigen wir eine Pumpe. Wir haben eine Schlauchpumpe gewählt, da sie akkurat Flüssigkeiten befördern kann.

Magnetventile

Um die einzelnen Flaschen anzusteuern und entleeren zu können, haben wir Magnetventile gewählt, die auf einer 12V Arbeitsspannung laufen. Wir benötigen sechs Ventile für die

Flaschen, bestehend aus fünf mit Inhalt für Getränke und eine Flasche mit Wasser, die zur Reinigung dient.

Schlauchsystem

Natürlich müssen die Flüssigkeiten auch transportiert werden. Dazu dient ein einfaches Schlauchsystem, das mit Hilfe von Silikonstopfen an den Flaschen befestigt wird. Die Schläuche werden an den Ventilen befestigt. Auf dem Weg zur Pumpe sollen die Schläuche durch einfache T-Stücke verbunden werden, die dann zur Pumpe führen.

5.2.3. ELEKTRONISCHE KOMPONENTE

Raspberry Pi

Als Gehirn unseres Projektes haben wir uns für einen Raspberry Pi entschieden. Die Raspberry Pis sind Einplatinencomputer, die einerseits dazu in der Lage sind, ein komplettes Betriebssystem und Programme auszuführen, andererseits über Schnittstellen verfügen, mit denen elektronische Bauteile angeschlossen und somit angesteuert, bzw. ausgelesen werden können. Die Entscheidung fiel in erster Linie gegen einen Arduino und für den Raspberry Pi, weil wir eine komplette grafische Benutzeroberfläche implementieren wollten. Dies ist mit einem Arduino nur begrenzt möglich, wohingegen der Raspberry Pi deutlich einfacher im Umgang ist. Außerdem beherrschen die Raspberry Pis die Programmiersprache Python, welche über eine simple Syntax verfügt, und somit die Programmierung des Programms erheblich vereinfacht.

Eingabe / Input

Zur Steuerung sollte die Maschine mit sechs Tastern ausgestattet werden, wovon vier ein Steuerkreuz oder Pfeiltasten repräsentieren, und zwei weitere die Bewegungen durch die Benutzeroberfläche ermöglichen.

Ansteuerung der Mechanik / Output

Da die Magnetventile eine Arbeitsspannung von 12V haben, der Raspberry Pi aber nur 3,3V zur Verfügung stellen kann, benötigten wir Relais. Hier mussten wir insbesondere darauf achten, dass diese mit der geringen Steuerspannung von 3,3V kompatibel sind. Unsere Auswahl fiel schließlich auf Relais-Boards von AZ-Delivery, die außerdem in der Schule zur Verfügung standen.

Anzeigen der Benutzeroberfläche

Schließlich benötigten wir einen Monitor, um die Benutzeroberfläche überhaupt anzeigen zu können. Herr Friedrichs konnte uns hier mit einer großen Auswahl aushelfen und wir haben uns einen möglichst kleinen Monitor ausgesucht, der Platz auf der Maschine finden würde.

5.3. BESTELLUNG

Tabelle 1: Bestellliste

Nr.	Beschreibung	Einzelpreis	Anzahl	Gesamtpreis
1	Magnetventile	3,39 €	7	23,73 €
2	Schlauch und Schlauchverbinder	8,99 €	1	8,99 €
3	Zusätzlicher Schlauch	8,99 €	1	8,99 €
4	Schlauchpumpe	11,89 €	1	11,89 €
5	Silikonstopfen	20,99 €	1	20,99 €
6	Taster	8,99 €	1	8,99 €
7	Netzteil	11,99 €	1	11,99 €
8	Aluminium-Profile	Ca. 60 €	6m	60 €
			Summe:	155,57 €

Zusätzliche Materialien, die von der Schule zur Verfügung gestellt wurden:

- Raspberry Pi 3B
- Monitor
- 2x AZ-Delivery 4-fach Relais-Boards
- Verzinkte Stahlbleche
- Schrauben und Muttern

Modulnummer			**1**
Modulname			Rahmenbau
Ziel des Moduls			**Konstruktion eines Grundrahmens**
Veranschlagte Zeit	4h	**Benötigte Zeit**	6h
Mitwirkende	André Milling, Max Schiemann		
Benutzte Fertigungsverfahren		**Benutzte Werkzeuge**	
Trennen	Sägen		Bandsäge
	Bohren		Tischbohrer mit 6er Bohrer
	Gewindebohren		6er Gewindeschneider Set
Fügen	Kraftschluss		
Welche Probleme sind aufgetreten?		**Wie wurden sie gelöst?**	
Keine vorgefertigten Fügemethoden anwendbar			Bohrungen durch die Profile

5.4.1. VORÜBERLEGUNG

Bei der Bemaßung des Rahmens mussten wir uns vorher im Klaren sein, wie das Gerät am Ende aussehen soll. Einmal zurechtgesägt und verbunden sind wir nicht mehr in der Lage, am Grundgerüst Veränderungen vorzunehmen.

5.4.2. BESTIMMEN DER GRÖSSE

Aus den ersten Skizzen wurde eine erste technische Zeichnung erstellt, die unsere finalen Maße ergeben sollte.

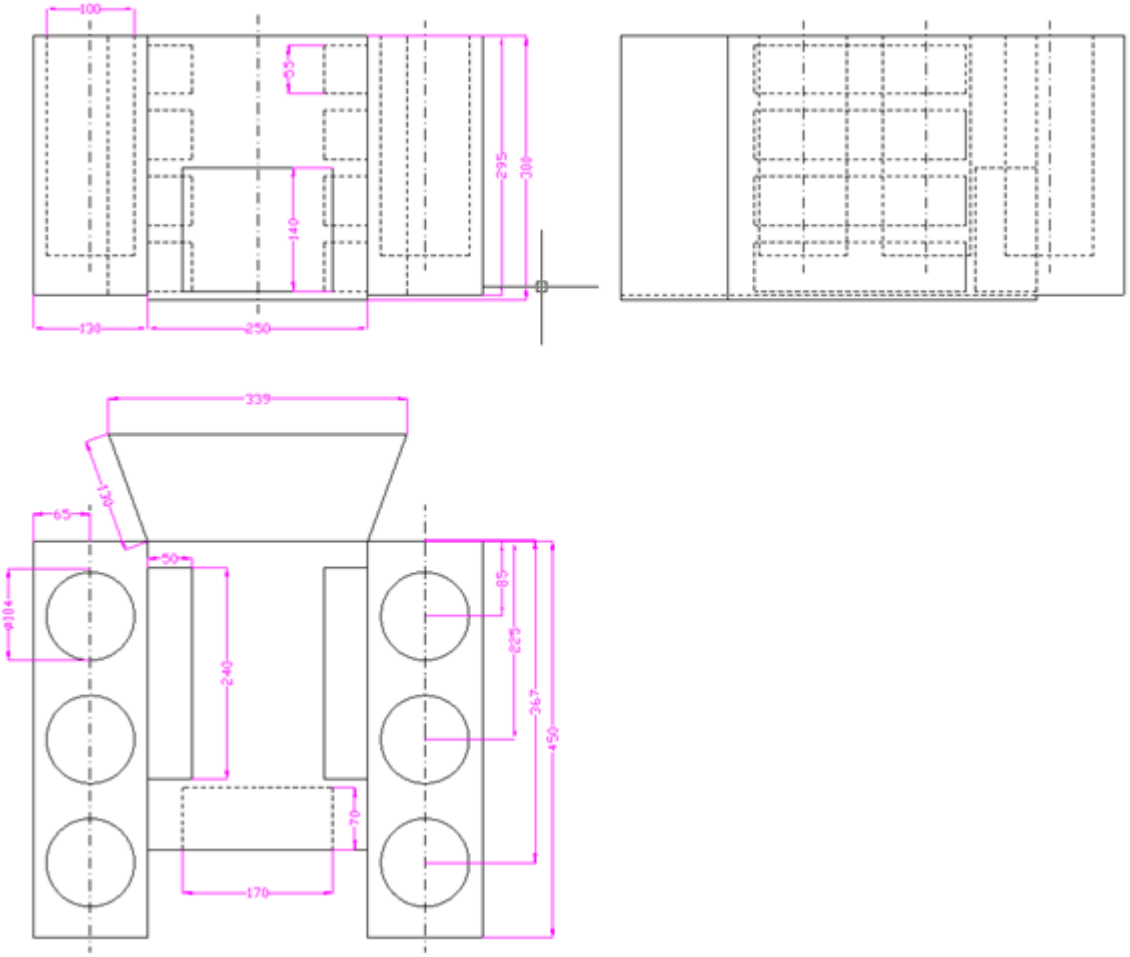

Abbildung 4: Erste technische Zeichnung

Dabei ist uns aufgefallen, dass wir den Mittelteil um 100mm verbreitern mussten, damit es in etwa so breit ist wie der für uns zur Verfügung stehender Monitor. Dadurch kamen folgende Abmessungen zu Stande:

Mittelteil:	L/B/H:	350mm/350mm/245mm
Flügelteile:	L/B/H:	450mm/130mm/295mm

Durch die Breite der Profile von 20mm ergaben sich folgende Abmessungen:

Tabelle 2: Profillängen

Länge	*Anzahl*
345mm	4
310mm	8
295mm	8
90mm	8
410mm	8

5.4.3. ZUSAMMENBAU

Die fertig bemaßten Profile wurden mit einer Bandsäge in Länge zugeschnitten. Das Unterfangen hat sich ein wenig verzögert, weil die dafür vorgesehen Bandsäge falsch arretiert war und nicht im rechten Winkel trennt. Nach Versuchen, das Problem zu lösen, haben wir die Profile mit einer anderen Bandsäge zurechtgeschnitten.

Nach dem Zuschnitt mussten wir in einigen Profilstücken ein Gewinde reinschneiden. Dazu haben wir uns für einen Satz 6er Gewindeschneider entschieden. Dabei sind wir behutsam vorgegangen, weil ein sehr weiches Material bearbeitet wurde. Anschließend wurden an weiteren Profilstücken Bohrungen mit einem 6er Bohrer vorgenommen. Die Bohrungen erfolgten einmal in einem Abstand von 10mm und weiteren 30mm um 90 Grad versetzt. Zum Schluss haben wir die Profile mit 6er Schrauben verbunden. Einige Gewinde mussten wir nochmal nachbessern, weil die Schrauben nicht ordnungsgemäß angezogen haben.

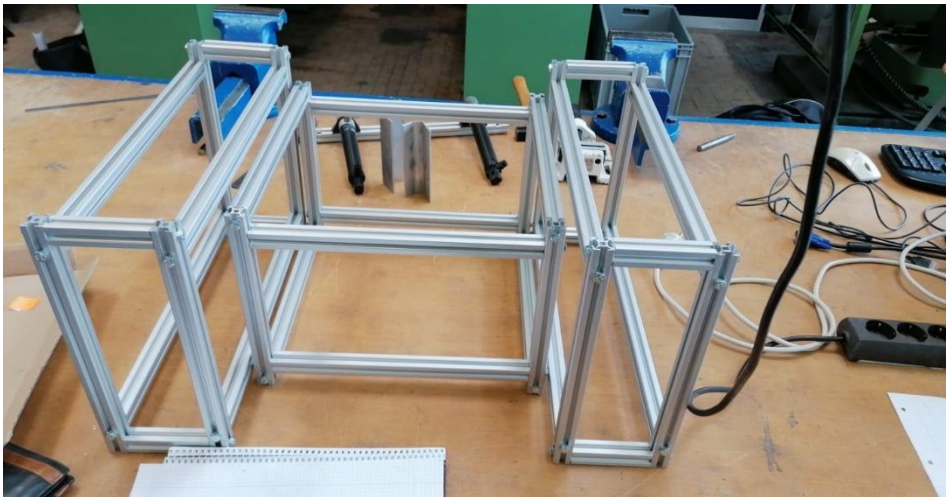

Abbildung 5: fertige Rahmen

Modulnummer		**2**	
Modulname		Verschalung	
Ziel des Moduls		**Konstruktion der Außenhülle**	
Veranschlagte Zeit	4h	**Benötigte Zeit**	6h
Mitwirkende	André Milling, Max Schiemann		
Benutzte Fertigungsverfahren		**Benutzte Werkzeuge**	
Trennen	Schneiden	Schlagschere	
	Entgraten	Entgrater, Feile, Kegelsenker	
	Bohren	Standbohrmaschine, 7er Bohrer	
Fügen	Kraftschluss		
Umformen	Biegen		
Welche Probleme sind aufgetreten?		**Wie wurden sie gelöst?**	
Durch Rundungen der Aluprofile entstehen Überstände		Bleche ein paar Millimeter kleiner zugeschnitten	
Durch kleinere Bleche keine Maßbezugskante mehr möglich; Durch Ungenauigkeit des Rahmens keine akkuraten Bohrungen möglich		Nutzung von 7er Bohrern für mehr Spiel; Kegelsenker zur Vergrößerung der Bohrungen	

5.5.1. PLANUNG DER BLECHZUSCHNITTE

Bei den Bemaßungen der Außenverkleidungen haben wir uns ebenfalls an die Maße aus Kapitel 5.4.2 gehalten. Dabei ist jedoch zu beachten, dass die Profile an den Kanten Rundungen besitzen. Hätten wir die Bleche auf das Maß zugeschnitten, würden sich Überstände ergeben, die eventuell zu Verletzungen führen könnten. So haben wir uns entschieden, die Bleche je 3mm kleiner zu bemaßen.

Wir haben uns dazu entschieden, dem mittleren Kasten einen Boden sowie einen Deckel zu geben. Zusätzlich soll in der Front eine ausreichend große rechteckige Öffnung für das Glas und für die Abdeckplatte des Bedientasten ausgeschnitten werden. In dem hinteren Blech ist ein Loch für die Stromversorgung, sowie eine weitere Öffnung für die Hauptplatine vorgesehen.

Die Verschalung an der Seite haben wir weggelassen, da dadurch der Zugang zum Innenraum des Gerätes gewährleistet wird. Für die Seitenteile haben wir uns überlegt, den Boden wegzulassen, weil es konstruktionstechnisch aufwändiger wäre. Zusätzlich haben die beiden Seitenteile keine Deckel, da man da die Flaschen von oben kopfüber reinsetzt. Eine Seitenwand wurde freigelassen, um den Zugang zu den Flaschen und den Schläuchen zu ermöglichen.

Durch diese Vorüberlegungen entstanden folgende Maße der Blechzuschnitte:

Tabelle 3: Bemaßung der Blechstücke

POSITION	LÄNGE	BREITE	ANZAHL
FRONT/HINTEN	347mm	242mm	2
DECKEL	347mm	347mm	1
BODEN	387mm	347mm	1
FLÜGEL VORNE UND HINTEN	127mm	293mm	4
SEITENTEIL FLÜGEL	447mm	293mm	2

5.5.2. ZUSCHNITT UND MONTIERUNG AM RAHMEN

Die Bleche haben wir nach den ermittelten Maßen mit einer Schlagschere zurechtgeschnitten und anschließend mit einem Entgrater die scharfen Kanten entfernt. Da der Rahmen nicht akkurat lotrecht war und wir keine einheitlichen Maßbezugskanten besaßen, haben wir die Positionen nach „Augenmaß“ angerissen und gekörnt. Um trotzdem alles gut mit Schrauben verbinden zu können, haben wir die Bohrungen eine Nummer größer - 7er Bohrer - angesetzt, damit wir bei der Montage Spielraum hatten. Die Bohrungen wurden mit einem Kegelsenker entgratet. In der Front wurden die geplanten Öffnungen für das Glas und das Schaltpult angerissen, vorgebohrt und mit einer Stichsäge ausgeschnitten. Scharfe Kanten wurden hier ebenfalls entfernt. Dasselbe haben wir für die Rückseite durchgeführt. Da wurde ein größeres Loch

für den Zugang der Stromversorgung gebohrt und ein Rechteck für das IO- Panel des Raspberry ausgeschnitten.

Den Boden haben wir etwas länger gewählt, damit wir es je 20mm entlang der kürzeren Kante um 90 Grad biegen können. Dies war nötig, damit man so die Bodenplatte an den Rahmen montieren konnte.

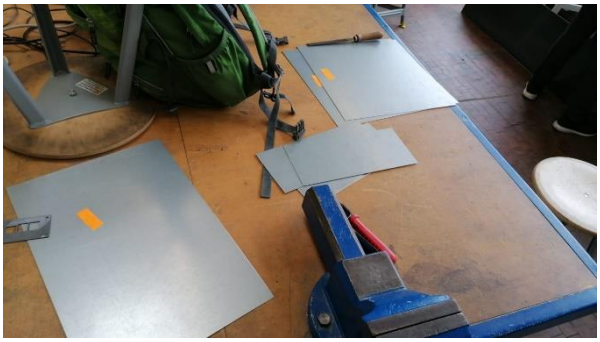

Abbildung 7: Zugeschnittene Bleche

Abbildung 6: vorderes Blech mit Tastenblende

Modulnummer		**3**	
Modulname		Halterung der Komponenten	
Ziel des Moduls		**Bauteile produzieren, die die elektronischen Komponenten sicher am Gehäuse befestigen**	
Veranschlagte Zeit	So viel wie nötig / Heim-arbeit	**Benötigte Zeit**	5h Erstellen, 25h Drucken
Mitwirkende	Arvid Randow		
Benutzte Fertigungsverfahren		**Benutzte Werkzeuge**	
Additiv: Fused Deposition Modeling / Schmelzschichtung		3D-Drucker: Anycubic Mega S	
Benutzte Software			
Autodesk Fusion 360		Erstellung der 3D-Modelle / CAD-Dateien	
Ultimaker Cura		Vorbereitung des 3D-Drucks	

Um die elektronischen Komponenten an den Rahmen oder an der Verschalung zu befestigen, haben wir einen privat zur Verfügung stehenden 3D-Drucker verwendet, um Halterungen zu drucken.

5.6.1. DESIGN / ERSTELLUNG DER 3D-MODELLE

Im ersten Schritt müssen die Halterungen entworfen werden, wozu wir „Fusion 360“ von Autodesk verwendet haben. Die einzelnen Arbeitsschritte werden hier einmal am Beispiel der Halterung für die Relais gezeigt. Zuerst wird eine Skizze vom Grundriss angefertigt. Die Halterung des Relais-Boards soll an einem Aluminium-Profil befestigt werden, beziehungsweise aufgeschoben werden und das Relais-Board selbst wird auf die Halterung geschoben, sodass es jederzeit wieder entfernt werden kann. In der Abbildung 10 wird das Alu-Profil lila und das Bauteil schwarz dargestellt. Aus diesem Grundriss kann eine Art Rohform „extrudiert“ werden, welche dann weiterbearbeitet werden kann. Hier werden die zusätzlichen Schienen an den Seiten hinzugefügt, und, um Material einzusparen, die Grundfläche ausgeschnitten und durch ein Kreuz ersetzt. Wenn das Modell fertig ist, wird es als Netzdatei im STL-Format exportiert.

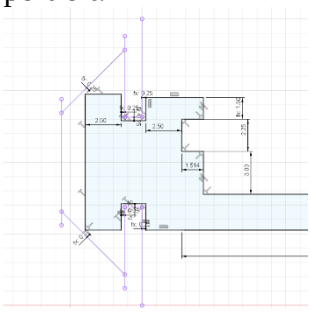

Abbildung 10: 2D-Entwurf

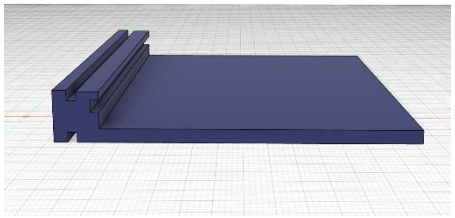

Abbildung 9: 3D Entwurf

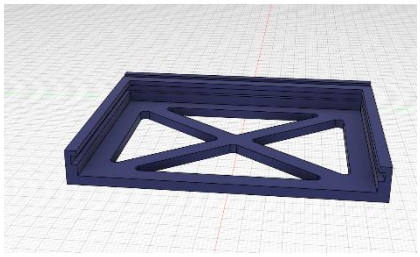

Abbildung 8: Finales Modell

5.6.2. AUSDRUCKEN

Um ein Modell drucken zu können, muss es „gesliced“ werden. Dazu wird eine spezielle Software benötigt, in diesem Fall Ultimaker Cura. Dieses Programm nimmt die STL-Datei und erstellt daraus Anweisungen für den Drucker. Dabei stehen auch eine Vielzahl von Einstellungen zur Verfügung, die das Verhalten des Druckers und damit die Geschwindigkeit des

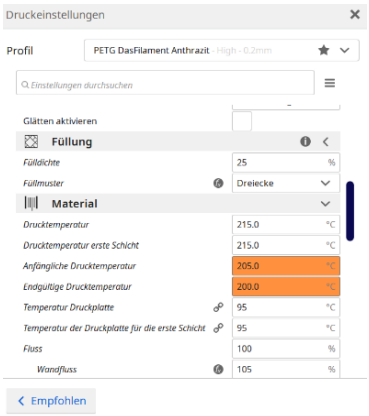

Abbildung 11: Einstellungen zum Slicing

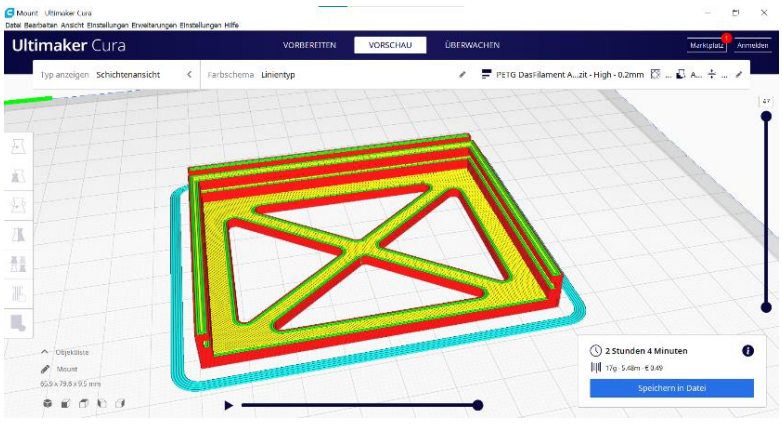

Abbildung 12: Bedienoberfläche Cura

Druckes und Qualität des Bauteils beeinflussen. Beispielsweise können die Höhe der Schichten, die Geschwindigkeit oder die Temperatur geändert werden. Das fertig „geslicede“ Modell, also die Anweisungen für den 3D-Drucker, kann danach als G-Code auf einer SD-Karte

gespeichert und auf den Drucker übertragen werden. Ein G-Code, oder DIN-Programm ist auf allen CNC-Maschinen, wie zum Beispiel CNC-Fräsen oder 3D-Druckern lauffähig. Er besteht aus einfachen Anweisungen, unter anderen welche Position der Druckkopf anfahren soll.
Bei dem 3D-Drucker handelt es sich um einen FDM-Drucker. Das heißt, dass das Druckmaterial geschmolzen und durch eine Düse auf das Druckbett gebracht wird. Auf diese Weise wird Schicht für Schicht Material aufgetragen, bis der Druck abgeschlossen ist. Abgesehen von der Entfernung von ein paar Kunststoff-Fäden, oder das Entfernen von Stützstrukturen ist normalerweise keine Nachbearbeitung mehr erforderlich, und die Bauteile können direkt eingebaut werden. Für FDM-Drucker gibt es eine Vielzahl von verschiedenen Materialien, wir haben uns hier für PLA und PETG entschieden, da beide Materialien verhältnismäßig einfach zu verarbeiten sind und eine gewisse Stabilität aufweisen. Für die meisten nicht sichtbaren Bauteile verwendeten wir PLA in blau und für die sichtbaren Bauteile aus optischen Gründen PETG in dunkelgrau.

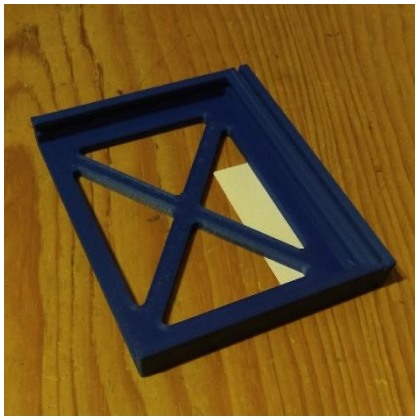

Abbildung 13: fertige Halterung

Abbildung 14: Halterung mit Relais-Boards

5.6.3. EINBAUEN

Die meisten Bauteile können einfach auf die Alu-Profile aufgeschoben werden. Ausnahme sind hier die Halterung für den Raspberry Pi und die Blende für die Taster. Für diese wurden Löcher gebohrt, beziehungsweise geschnitten, wo die Halterungen eingesetzt werden konnten.

Modulnummer		**4**	
Modulname		Elektronik	
Ziel des Moduls		**Komponenten mit Strom versorgen**	
Veranschlagte Zeit	3h	**Benötigte Zeit**	4.5h
Mitwirkende	Arvid Randow, Max Schiemann, André Milling		
Benutzte Fertigungsverfahren		**Benutzte Werkzeuge**	
Löten		Lötkolben	
Crimpen		Crimpzange	
		Seitenschneider	
Welche Probleme sind aufgetreten?		**Wie wurden sie gelöst?**	
Wackelkontakte		Betroffene Stecker neu vercrimpt	
Verwendung des internen Pull-Up-widerstandes des Raspberry Pis		Recherche zu Funktionsweise von Pull-Up-Widerständen	

5.7.1. INPUT – TASTER

Die Taster wurden an einer 3D-gedruckten Blende an der Vorderseite befestigt. Um diese Taster auszulesen, haben wir das „`gpiozero`“-Modul verwendet. Dieses Modul geht standardmäßig davon aus, dass die Taster über den internen Pull-Up-Widerstand des Raspberry Pis ausgelesen werden. Daher konnten wir die Taster einfach an die Input-Pins und an Masse anschließen.

5.7.2. OUTPUT – MAGNETVENTILE UND PUMPE

5.7.2.1. BAUTEILE

- Raspberry Pi 3b: Der Raspberry Pi ist ein Einplatinencomputer. Das heißt, er verfügt über alles, was auch ein normaler Computer hat, also ein Betriebssystem, USB-Anschlüsse, HDMI-Anschlüsse, etc., allerdings befinden sich alle Bauteile auf einer Platine. Zusätzlich dazu hat der Raspberry GPIO-Pins (**G**eneral-**P**urpose-**I**nput-**O**utput-Pins), mit denen, wie bei den Arduino, andere Bauteile ausgelesen und angesteuert werden können. Diese GPIO-Pins haben eine Arbeitsspannung von 3.3V, er hat aber auch Pins mit einer Spannung von 5V. Er war für unser Projekt perfekt geeignet, weil er einerseits einen Monitor ansteuern kann, und damit eine komplette Benutzeroberfläche möglich ist, bietet andererseits aber auch Möglichkeiten, elektronische Bauteile anzusteuern und auszulesen.

Abbildung 15: Raspberry Pi 3 B

- AZ-Delivery Relais-Board: Wir verwenden diese Relais-Boards, um die Pumpe und die Magnetventile anzusteuern, da sich die Arbeitsspannung des Raspberry Pis von der der Aktoren unterscheidet. Da die benötigte Spannung im Steuerkreis der Relais immer noch 5V beträgt, verfügt das Board über Optokoppler, die ein Signal von einem Stromkreis mit 3.3V an einen Stromkreis mit 5V weiterleiten können. Allerdings hat sich nach einiger Recherche und Untersuchung der Platine herausgestellt, dass diese Optokoppler das Signal anscheinend negieren. Daher mussten wir auch selbst das Signal vom Raspberry Pi nochmal negieren, damit sich die Relais wie gewünscht verhalten.

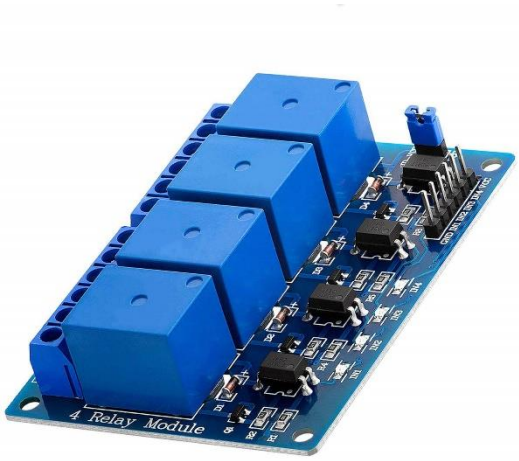

Abbildung 16: AZ-Delivery 4x Relais-Board

- Pumpe: Bei der Pumpe handelt es sich um eine Schlauchpumpe. Das heißt, dass ein Schlauch durch mehrere Rollen zusammengedrückt wird. Auf diese Weise kommt die Flüssigkeit nie mit mechanischen Teilen in Berührung,

Abbildung 17: Schlauchpumpe

Abbildung 18: 12V Magnetventil

weshalb solche Pumpen sehr hygienisch sind und zum Beispiel in der Medizin Anwendung finden.

- Magnetventil: Unsere Magnetventile sind Öffner und werden mit 12V Gleichstrom versorgt. Wir haben insgesamt sechs Ventile eingebaut, fünf davon für Getränke, das sechste ist für Reinigungswasser, um die Schläuche zwischen den Mischvorgängen zu reinigen.

5.7.3. STROMVERSORGUNG

- Step-Down-Converter: Unser Projekt wird mit einem 12V-Netzteil versorgt, allerdings darf der Raspberry Pi nur mit maximal 5V versorgt werden. Deshalb benötigten wir einen Step-Down-Converter, um die Spannung herunterzuregeln.

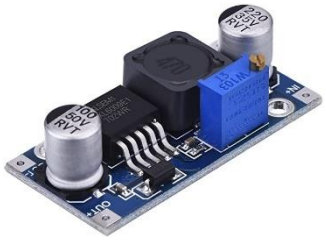

Abbildung 19: Step-Down-Converter LM2596

- Verteilung: Da wir mehrere Bauteile verwenden, die Strom benötigen, brauchten wir auch einen möglichst einfachen, aber auch eleganten und sicheren Weg, die Leitungen aufzuteilen. Aus diesem Grund haben wir Herrn Janovski um Rat gebeten. Er konnte uns mit einer Hutschiene und dazugehörige Klammern ausstatten. Die Klammern können an der Hutschiene angebracht werden und mittels spezieller Brücken miteinander verbunden werden. Außerdem bekamen wir Aderendhülsen, die wir an den Leitungen anbringen konnten, um ein Abbrechen einzelner Adern zu verhindern.

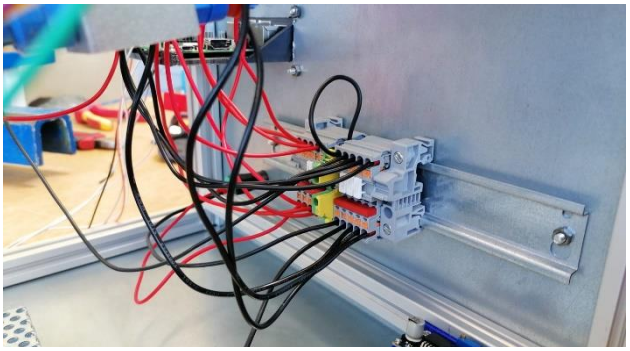

Abbildung 20: Montage der Hutschienen und Stromverteilung

- Bei der Farbkodierung haben wir uns darum bemüht, dass wir uns an Standards und Normen halten. Allerdings war dies durch die uns zur Verfügung stehenden Kabel und einzelner Planänderungen nicht immer möglich, weshalb die Kodierung möglicherweise abweicht. Im Folgenden haben wir die Kodierung aufgelistet, aufgeteilt nach Stromversorgung, Ausgängen und Eingängen:

Spannungsversorgung

Typ	Farbe
+12V	Rot
+ 5V	Orange
GND	Schwarz
GND Taster	Gelb

Ansteuerung der Relais

Typ	Farbe
Pumpe	Blau
Ventile	Weiß / Grau

Auslesen der Taster

Taster	Farbe
Vor / Zurück	Grün
Links / Rechts	Lila
Hoch / Run-ter	Blau

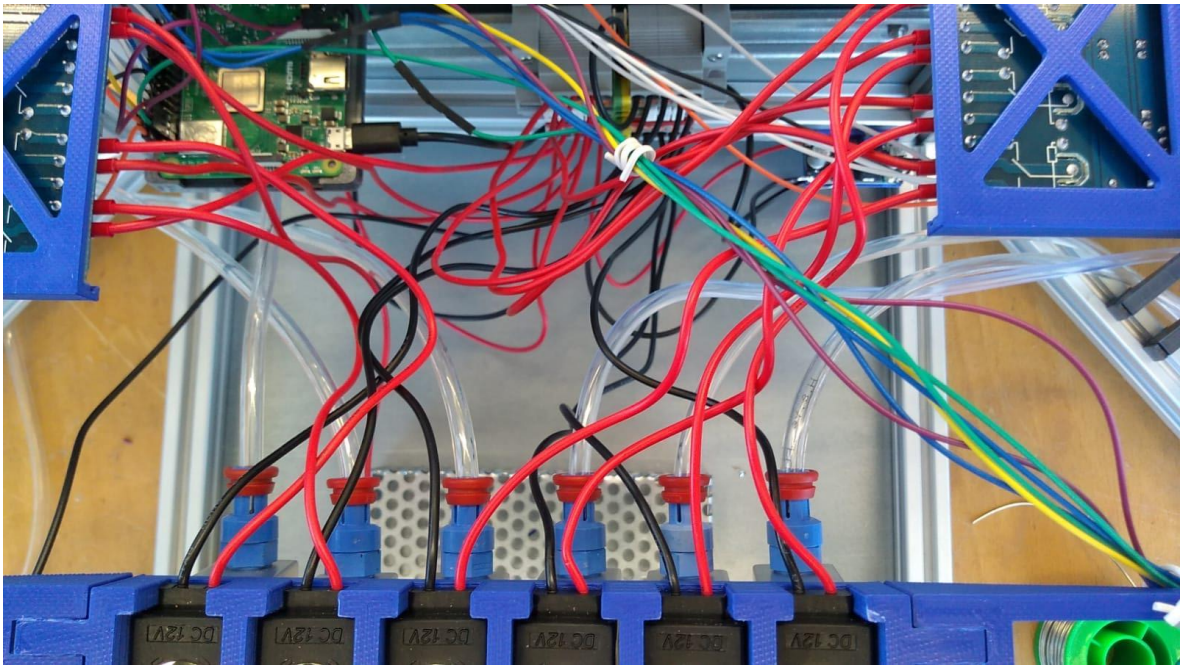

Abbildung 21: Kabelmanagement

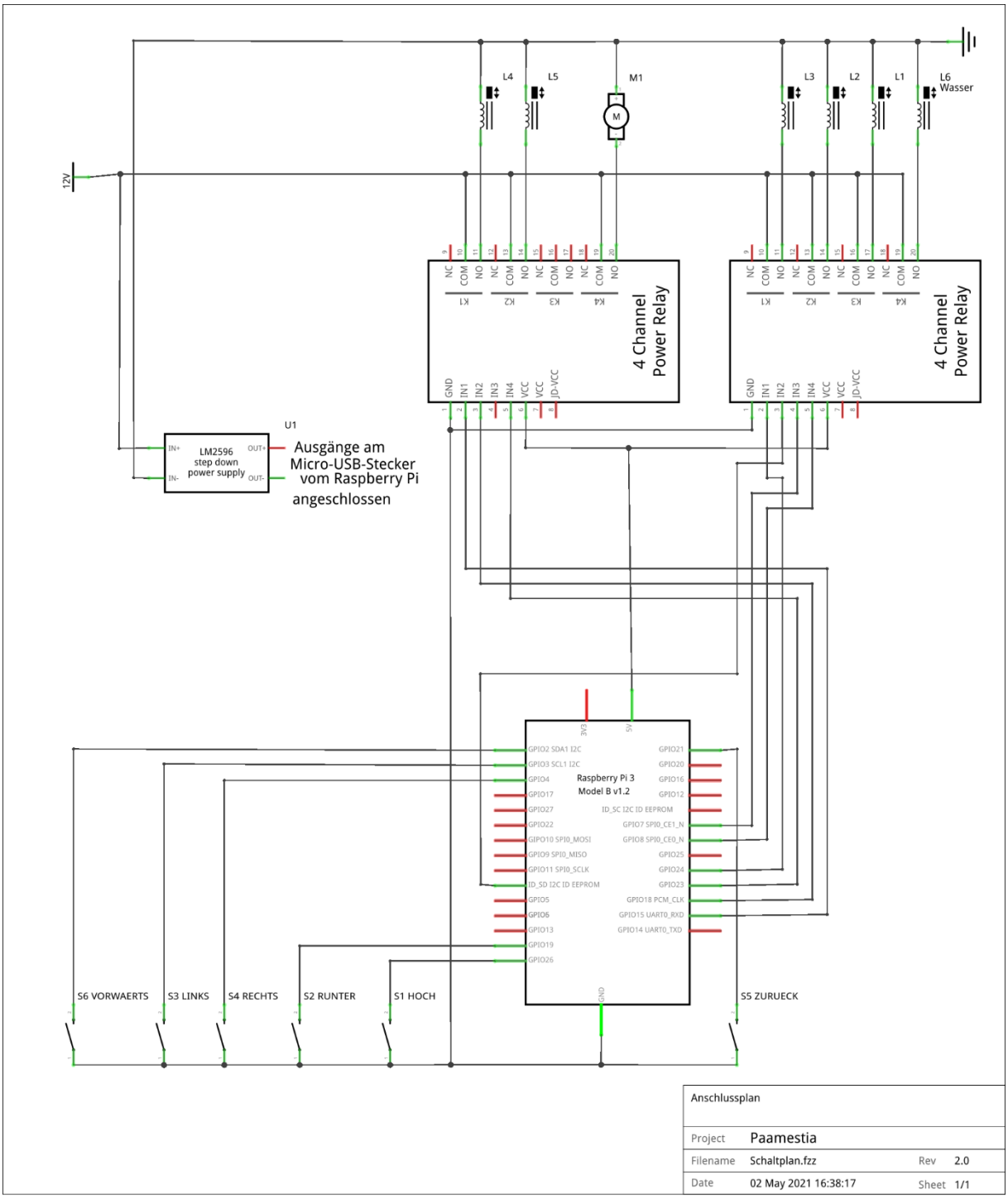

Abbildung 22: Schaltplan

Modulnummer			**5**
Modulname			**Schlauchsystem**
Ziel des Moduls			**Konstruktion und Anbringung der Flüssigkeitsführenden Komponenten**
Veranschlagte Zeit	1.5h	**Benötigte Zeit**	1.5h
Mitwirkende	André Milling, Max Schiemann, Arvid Randow		
Benutzte Fertigungsverfahren		**Benutzte Werkzeuge**	
Schneiden		Seitenschneider	
Welche Probleme sind aufgetreten?		**Wie wurden sie gelöst?**	
Verschiedenen Lecks, unter anderen an den Ventilen		Konnten keine Lösungsansätze finden	
Mangelnde Ansaugkraft der Pumpe, unter anderen verursacht durch einströmende Luft in das Schlauchsystem		Konnten keine Lösungsansätze finden	

Das Schlauchsystem soll das Wasser von den Flaschen zu den Ventilen und anschließend zur Pumpe befördern. Als Anschluss für die Flaschen haben wir einfach Silikonstopfen gewählt, in denen wir jeweils ein Loch gebohrt haben. In die Löcher wurden Schläuche eingeführt. An den Ventilen haben wir Schnellspannsysteme für Pneumatik-Schläuche gewählt, damit wir die

Schläuche von den Flaschen anbringen konnten. Auf der Pumpenseite haben wir mit recht Kurzen Teilstücken vom Schlauch und T-Stücken aus Kunststoff eine Art Mischbatterie entwickelt. Dabei haben wir darauf geachtet, dass der Teil, durch was das Wasser für die Reinigung fließ, den längsten Weg nehmen muss.

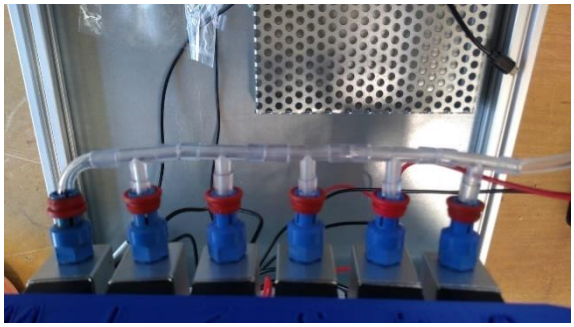

Abbildung 23: Mischbatterie

Abbildung 24: Schlauchsystem / Anschlüsse für die Getränke

Eine Problematik stellten hierbei Lecks in dem Schnellspannsystem dar, welches nur für Luft ausgelegt ist, und daher Probleme mit Flüssigkeiten verursacht. Dieses Problem konnten wir jedoch durch Zeitmangel und der für unseren Einsatz nicht geeigneten Bauteile nicht mehr lösen.

Modulnummer			**6**
Modulname			Programmierung
Ziel des Moduls			**Ein funktionierendes Programm erstellen, welche die Bedienoberfläche und die elektronischen Bauteile miteinander verknüpft**
Veranschlagte Zeit	So viel wie nötig / Heimarbeit	**Benötigte Zeit**	Ca. 150-200h
Mitwirkende	Arvid Randow		
Benutzte Software		**Benutzte Python-Module**	
Programmiersprache: Python		Benutzeroberfläche: `Pygame`, `OpenCV`, `Tkinter`	
Entwicklungsumgebung: Visual Studio Code und Genius		Steuerung der Ein- und Ausgänge: `gpiozero`	
Betriebssystem des Raspberry Pis: Raspbian OS		Zusätzliche Module: `numpy`, `time`, `pathlib`, `os`	
Zusätzliche Software: GitHub und SSH			
Welche Probleme sind aufgetreten?		**Wie wurden sie gelöst?**	

Begrenzte Kapazitäten des Raspberry Pis sowohl beim Arbeitsspeicher als auch der Geschwindigkeit	Verringerung der Bildschirmauflösung, Verwendung eines effizienteren Python-Moduls

Der Quellcode ist für dieses Projekt zu umfangreich, um ihn hier komplett einzufügen. Daher haben wir ihn sowohl im Original als auch als HTML-Datei hinterlegt und werden hier nur auf die einzelnen Bestandteile eingehen.

5.9.1. SYSTEM: RASPBIAN OS

Da der Raspberry Pi ein vollständiger Computer ist, benutzt er auch ein Betriebssystem und kann nicht wie ein Arduino programmiert werden. Als Betriebssystem haben wir uns für Raspbian OS entschieden, da es das offizielle Betriebssystem für den Raspberry Pi ist und leicht zu verwenden ist. Raspbian OS basiert auf der Linux-Distribution Debian und bietet daher eine komplette Desktop-Umgebung. Um unser Programm auszuführen, war lediglich die Installation von Python und den benötigten Python-Modulen notwendig. Außerdem sollte das Programm automatisch starten, wenn der Raspberry Pi hochfährt. Dazu gibt es in Linux einen speziellen Ordner, in dem nach ausführbaren Verknüpfungen gesucht wird. Dort mussten wir lediglich eine eigene Verknüpfung hinzufügen.

5.9.2. SOFTWARE

Als Programmiersprache haben wir Python verwendet, da diese einfach zu verwenden ist und viele Möglichkeiten, vor allem in Verbindung mit dem Raspberry Pi, bietet. Da unser Programm sehr umfangreich wurde, haben wir es in mehrere Dateien aufgeteilt und als Module verwendet. Dazu kamen weitere Module, die weitere Aufgaben übernommen haben.

Durch den Umfang und der damit einhergehenden Unübersichtlichkeit fügen wir hier nicht den kompletten Quellcode ein, sondern erklären nur die einzelnen Bestandteile:

- `globals.py`: Diese Datei beinhaltet Variablen und Objekte, die von mehreren Dateien benötigt werden. Dazu gehören zum Beispiel die Breite und Höhe des Fensters oder die Framerate. Außerdem gibt es die Variable `prog_pos`. Sie speichert das aktuelle ausgewählte Menü. Diese Variable besteht aus zwei Textzeichen. Das erste speichert den Modus oder die ausgewählte Position im Hauptmenü, das zweite die genaue Position bei der Rezeptwahl, dem freien Mischen oder den Einstellungen.

```
prog_active = True      # set to False to end program

FPS = 24                # frames per second
W, H = 800, 600         # width and height of the window

prog_pos = 'i'          # saves current position in program flow
```

- `paamestia.py`: Das ist der eigentliche Python-Skript, der die grundsätzliche Struktur des Programmes kontrolliert und unter anderem die richtige Funktion ausführt, je nach dem in welchem Menü man sich gerade befindet. In der Abbildung ist kurzer Ausschnitt dieser Zuordnung gezeigt.

```
# run external methods according to prog_pos
    if gl.prog_pos == 'i':                  # intro
        ui.intro()

    elif gl.prog_pos == 'm':              # main menu
        ui.main_menu()

    elif gl.prog_pos[0] == 'f':           # free mixing
        if gl.prog_pos[1] == 'c':             # mix cocktail
            ui.free_choose()
        elif gl.prog_pos[1] == 'o':           # output
            ui.free_output()
```

- `media_lib.py`: Hier sind mehrere Klassen definiert, mit denen die grafische Oberfläche aufgebaut werden kann. Wir haben hier zum größten Teil das Modul „`Pygame`“ verwendet, welches in erster Linie zur Spieleentwicklung gedacht ist. Daher bietet es einige Funktionen, mit denen eine einfache Oberfläche erstellt werden kann. Hier zeigte sich eines der größten Probleme, die bei der Programmierung aufkamen. Geplant war es, kurze Videosequenzen als Übergänge oder im Hintergrund, um die Benutzeroberfläche etwas zu beleben. Nach einiger Recherche stellte sich allerdings heraus, das Pygame keine Möglichkeit bietet, Videos abzuspielen. Daher haben wir uns als Alternative überlegt, die einzelnen Bilder, aus denen ein Video besteht, einzeln abzuspeichern und nacheinander abzuspielen. Allerdings bedeutete dies auch, dass jedes Bild einzeln geladen werden muss, um dann angezeigt zu werden. Dies hat auch noch zuerst auf dem Computer funktioniert, allerdings stellte sich heraus, dass der Raspberry Pi mit dieser Aufgabe schlicht überfordert war. Zum einen ist er nicht schnell genug, um die Bilder in hoher Auflösung abzuspielen, zum anderen reichte der Arbeitsspeicher nicht aus, um überhaupt alle Bilder zu laden. Das erste Problem konnte leicht gelöst werden, indem wir die Auflösung der Bilder verringerten, allerdings tauchte das Problem wieder auf, wenn weitere Schaltflächen eingeblendet werden sollten. Das zweite Problem bestand auch noch weiterhin, und sämtliche Lösungsansätze funktionierten nicht ganz und waren nicht wirklich zufriedenstellend. Schließlich fand sich doch noch ein Weg mit „`OpenCV`“, ein Modul, das in erster Linie zur Bearbeitung

und Analyse von Bildern gedacht ist. Dieses Modul ist jedoch auch dazu in der Lage, Videodateien abzuspielen, und lässt sich auch in Pygame einbinden. Nach ein paar Stunden herumprobieren und Troubleshooting war die Videowiedergabe im Programm eingebaut und konnte genutzt werden. Die Geschwindigkeit des Raspberry Pis ist zwar immer noch begrenzt, weshalb wir jetzt bei einer Bildschirmauflösung von 800x600 Pixeln sind. Diese Datei beinhaltet allerdings neben der Videowiedergabe noch weitere Werkzeuge. Dazu gehören einfache Bilder, auswählbare Schaltflächen, Textflächen mit automatischem Umbruch und eine Art Füllstandsanzeige, die beim freien Mischen das eingestellte Verhältnis anzeigt.

- `ext_ui_methods.py`: In dieser Datei befinden sich die Funktionen von den einzelnen Menüs, welche über paamestia.py aufgerufen werden. Der Aufbau dieser Funktionen ist etwas komplizierter geworden und könnte vereinfacht werden. Allerdings wäre dies durch den fortgeschrittenen Entwicklungsstand und den dafür nötigen weitreichenden Änderungen sehr aufwendig und durch das nahende Projektende nicht mehr umsetzbar. Im Folgendem wird einmal der Aufbau von einer der Funktionen gezeigt. Es handelt sich hierbei um den ersten Teil der Einstellungen, in denen zwischen den Optionen gewählt werden kann: Als erstes werden die benötigten Variablen definiert. Dies beinhaltet den Hintergrund und sämtliche Bedienelemente und eine Variable, die anzeigt, dass das spezifische Menü aktiv ist, also angezeigt wird. Diese Variablen müssen global sein, damit ihr Zustand nicht verloren geht, wenn die Funktion wieder verlassen wird.

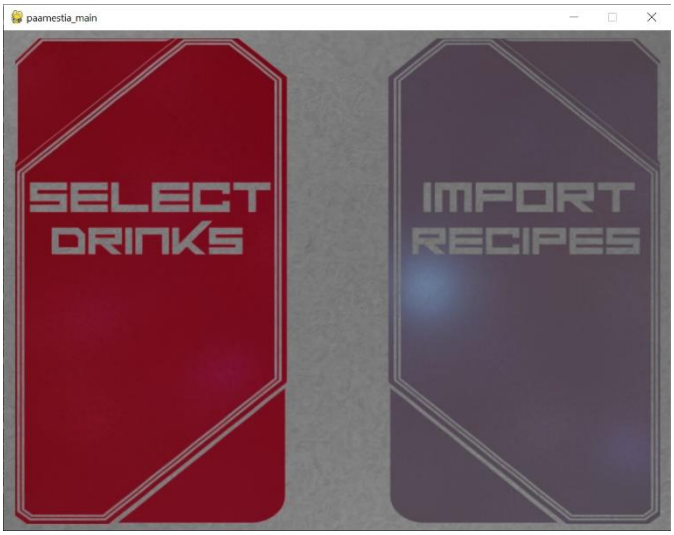

Abbildung 25: erste Ebene der Einstellungen

```
sc_active = False
sc_background = None            # background
sc_btns = []                    # list of all buttons
sc_pos = 0                      # positon / selected button
```

Als nächstes kommt die tatsächliche Funktion. Eingeleitet wird sie mit „global“. Dieser Befehl gibt an, welche Variablen schon global existieren und nicht neu lokal erstellt werden müssen. Dies ist wegen Pythons standardmäßigen Verhalten notwendig. Innerhalb einer Funktion werden Variablen immer zuerst neu erstellt, und nicht die schon vorhandene globale Variable geändert.

```
def settings_choose():
    global sc_active, sc_background, sc_btns, sc_pos
```

Dann wird das erstmalige Aufrufen der Funktion abgehandelt. Erreicht wir dies, indem `sc_active` überprüft wird. Wenn diese Variable noch „`Falsch`“ enthält, dann wird diese Funktion zum ersten Mal aufgerufen. Ist dies der Fall, werden zuerst sämtliche Variablen definiert. Das wird erst beim Aufruf der Funktion gemacht, weil gerade die Oberflächenelemente viel Arbeitsspeicher benötigen. Würden alle Elemente direkt geladen werden, wäre der Arbeitsspeicher vermutlich wieder überlastet, daher werden diese Elemente nur gespeichert, wenn sie auch benötigt werden.

```
if sc_active == False:      # when entering settings_choose
        sc_active = True
        sc_background = media_lib.Video(gl.gen_path + "/src/me-
dia/intro/intro.mp4")      # create background
        sc_background.start(repeat=True)


        # create buttons
        sc_btns.append(media_lib.But-
ton(gl.gen_path + "/src/props/", "prop_white.png", "prop_green.png"
, "prop_grey.png", 35, 35, 300, 530))
        sc_btns[-1].add_text("Getränke", gl.debug_font, (0,0,255))
        sc_btns.append(media_lib.But-
ton(gl.gen_path + "/src/props/", "prop_white.png", "prop_green.png"
, "prop_grey.png", gl.W-35-300, 35, 300, 530))
        sc_btns[-1].add_text("Importieren", gl.de-
bug_font, (0,0,255))
        sc_btns[sc_pos].selected = True
```

Als nächstes überprüft das Programm auf eine Eingabe des Benutzers. Dazu wird die Datei `io_lib.py` verwendet, indem sie wie ein Python-Modul importiert wird. Hier wird bei der Eingabe überprüft, ob der Benutzer zurück zum Hauptmenü gehen möchte, oder welche der Einstellungen er auswählt.

```
# input
    if io.read_input(io.BACK):      # if going back
        sc_active = False
        gl.prog_pos = 'm'
```

```
    elif io.read_input(io.NEXT):        # selecting a setting
        sc_active = False
        if sc_pos == 0:
            gl.prog_pos = 'sd'
        else:
            gl.prog_pos = 'si'
    elif io.read_input(io.LEFT):        # moving the selection
        sc_pos = 0
        sc_btns[0].selected = True
        sc_btns[1].selected = False
    elif io.read_input(io.RIGHT):
        sc_pos = 1
        sc_btns[0].selected = False
        sc_btns[1].selected = True
```

Schließlich werden die einzelnen Elemente „gezeichnet“, sprich auf dem Monitor angezeigt. Dabei ist die genaue Reihenfolge natürlich wichtig, damit nichts überdeckt wird .

```
# draw
    sc_background.draw()
    for i in sc_btns:
        i.draw()
```

Zu allerletzt wird überprüft, ob diese Oberfläche wieder verlassen wird. Dazu wird überprüft, ob `sc_active` im Verlauf der Funktion wieder auf „Falsch“ gesetzt wurde. Wenn dies der Fall ist, werden alle Variablen, die viel Arbeitsspeicher benötigen wieder auf „`None`“ zurückgesetzt oder, wenn es sich um eine Liste handelt, wieder geleert.

```
if sc_active == False:      # when leaving settings_choose
        sc_background = None
        sc_btns.clear()
```

- `io_lib.py` beinhaltet alle Funktionen für den In- und Output. Dieses Skript ist dementsprechend in zwei Teile aufgeteilt, einmal der Input und einmal der Output. Für den Input werden mehrere Funktionen benötigt. Zuerst sind da „`keyboard_input()`“ und „`update_input()`“, welche wiederholt aufgerufen werden müssen. Diese

Funktionen überprüfen einerseits, ob Tasten auf einer Tastatur gedrückt worden sind, und andererseits, ob die Taster an den GPIO-Pins betätigt worden sind. Um einen Tastendruck zu erkennen, wird für die Tastatur wieder das Pygame-Modul und für die Taster an den GPIO-Pins das Modul „`gpiozero`“ verwendet.

```
# read current state
    if gl.os_is_linux:              # for the raspberry pi / but-
tons on gpio
        up_state    = not UP_BT.is_pressed          # read every i
nput
        down_state  = not DOWN_BT.is_pressed
        left_state  = not LEFT_BT.is_pressed
        right_state = not RIGHT_BT.is_pressed
        next_state  = not NEXT_BT.is_pressed
        back_state  = not BACK_BT.is_pressed
```

Diese Daten werden dann abgespeichert, und können, wann immer nötig über die Funktion „`read_input()`“ abgerufen werden. Hier ist außerdem eine kleine Besonderheit eingebaut: Es wird nicht gespeichert, ob eine der Taste gerade gedrückt wird, sondern nur, ob die Taste jetzt gerade betätigt wurde. Dabei steht „`True`“ für „Taster wird betätigt“ und „`False`“ für „Taster wird nicht betätigt“. Wenn jetzt eine Taste gedrückt wird, gibt es einen kurzen Impuls, in dem „`True`“ abgespeichert und das Programm wechselt wieder auf „`False`“, selbst wenn der Taster weiter gehalten wird.

```
def read_input(input):
    """ returns input state as bool
    input: key [UP, DOWN, LEFT, RIGHT, NEXT, BACK]
    """
    global UP, DOWN, LEFT, RIGHT, NEXT, BACK
    global up_state, down_state, left_state, right_state, next_sta
te, back_state
    global up_state_prev, down_state_prev, left_state_prev, right_
state_prev, next_state_prev, back_state_prev


    # list of possible inputs and matching states
    keys = [(UP, up_state, up_state_prev),
            (DOWN, down_state, down_state_prev),
            (LEFT, left_state, left_state_prev),
```

```
            (RIGHT, right_state, right_state_prev),
            (NEXT, next_state, next_state_prev),
            (BACK, back_state, back_state_prev)]

    is_pressed = False              # this varia-
ble will be set True when asked button is pressed. Other-
wise it won't change

    for key in keys:                # loops through key list
        if key[0] == in-
put and key[2] == False:     # if tested key equals input and re-
quested key is pressed
            is_pressed = key[1]                    # set is_press
ed = True

    return is_pressed
```

Das Schalten der Ausgänge am Raspberry Pi gestaltet sich deutlich einfacher. Auch hier haben wir auf das „`gpiozero`“-Modul zurückgegriffen. Um einen Ausgangswert zu ändern muss nur die Funktion „`write_output()`“ verwendet werden.

- `drink_lib.py` kontrolliert alles was die Getränke, bzw. die Rezepte betrifft. Dieses Skript beginnt mit einer kleinen Setup-Routine, in der alle Getränke und Rezepte geladen werden. Dazu kann eingestellt werden, welche Getränke gerade an der Maschine angeschlossen sind und Daten abgerufen werden, zum Beispiel welche Rezepte gerade verfügbar sind.

```
""" ### SETUP ### """
# this part will run once, as soon this file is imported somewhere

# look for drinks
file1 = open(gl.gen_path + "/src/drinks", 'r')          # open file
drinks = file1.read-
lines()                                # save lines as a list
file1.close()
```

```
for idx, i in enumerate(drinks):                                # re-
move trailing newline characters
if drinks[idx].endswith('\n'):
drinks[idx] = drinks[idx][:-1]


print("[DR setup] drink list  content:")
print(drinks)


# look for recipes
for filename in os.listdir(dir_recipes):
recipes.append(filename)
recipes.remove('free_mixed_reci-
pe')             # sort the free mixed recipe, as it should not ap-
pear in the recipe list


# sort lists
if gl.os_is_linux:
drinks.sort()
recipes.sort()
```

Die Rezepte werden in einem eigenen Ordner abgespeichert (Mehr dazu bei Punkt 5.10). Wird nun ein Rezept ausgewählt, dann wird es geladen und in selbst entwickelte Befehle übersetzt. Die Syntax der Befehle ist sehr simpel, und es gibt nur eine begrenzte Anzahl von Befehlen. Das erste Zeichen eines Befehls gibt an, was getan werden soll, bzw. um was für einen Befehl es sich handelt. Die restlichen Zeichen bilden den dazugehörigen Parameter. Hier eine Auflistung der Befehle und deren Funktion:

Befehl	**Funktion**	**Beispiel**
o<Ventil>	Ventil öffnen und Pumpe starten	„o3“ öffnet das dritte Ventil und startet die Pumpe
c<Ventil>	Ventil schließen und Pumpe stoppen	„c1“ schließt das erste Ventil und stoppt die Pumpe
t<Ventil>	Stellt eine Stoppuhr ein (Zeit in Millisekunden)	„t4200“ stellt die Stoppuhr auf 4.2 Sekunden
w	Warten, bis die Stoppuhr abgelaufen ist. Dieser Befehl folgt immer auf „t“	Das Programm misst die vergangenen Sekunden und geht erst zu dem

		nächsten Schritt über, wenn die 4.2 Sekunden verstrichen sind
e	Dieser Befehl signalisiert, dass das Rezept abgeschlossen ist, und beendet damit den Vorgang	

Hier ein kurzes Beispiel von einem kompilierten Rezept:

```
o1              # erstes Ventil öffnen
t3000           # Timer auf 3 Sekunden stellen
w               # Auf Ablauf des Timers warten
c1              # erstes Ventil schließen
...
o4              # viertes Ventil öffnen
t12000          # Timer auf 12 Sekunden stellen
w               # Auf Ablauf des Timers warten
c4              # viertes Ventil schließen
e               # Mischprozedur beenden
```

Diese Befehle werden dann während der Mischprozedur ausgeführt. Eine Variable zählt dabei den aktuellen Arbeitsschritt mit, indem sie nach jedem abgeschlossenen Arbeitsschritt um eins erhöht wird.

```
def update_mixing():
    global is_mixing, recipe_step, commands, finishing_time

    if is_mixing:               # if something is currently mixed
        cmd = commands[recipe_step]
        if cmd[0] == 'o':                   # open valve
            valve = int(cmd[1])
            print("[DR UM] open valve " + str(valve))
            io.write_output(io.VALVES[valve], 1)
            io.write_output(io.PUMP, 1)
            recipe_step += 1
```

```
        elif cmd[0] == 'c':                 # close valve
            valve = int(cmd[1])
            print("[DR UM] close valve " + str(valve))
            io.write_output(io.VALVES[valve], 0)
            io.write_output(io.PUMP, 0)
            recipe_step += 1

        elif cmd[0] == 't':                 # set timer
            print("[DR UM] set timer " + cmd[1:] + " ms")
            finishing_time = current_milli_time() + int(cmd[1:])
            recipe_step += 1

        elif cmd[0] == 'w':                 # wait
            t = current_milli_time()
            if t >= finish-
ing_time:             # if waited long enough, advance to next step
                recipe_step += 1

        elif cmd[0] == 'e':                 # end sign
            print("[DR UM] end mixing")     # stops the mixing pro-
cess
            is_mixing = False



        # add debug information
        if gl.show_debug:
            gl.debug_text.ap-
pend("MIX cur. cmd.: " + str(cmd) + "; prev. cmd.: " + str(com-
mands[recipe_step-1]))
            gl.debug_text.append("MIX cmd nr.: " + str(rec-
ipe_step) + " / " + str(len(commands)))
```

5.9.3. DATEIEN UND RESSOURCEN FÜR DIE SOFTWARE

Unser Programm benötigt einige Dateien für die grafische Oberfläche und die Rezepte. Diese Dateien befinden sich alle im „`src`“-Ordner. Wir werden hier einen kurzen Überblick über die verschiedenen Dateien geben:

- Ordner „`fonts`“: Hier sind die Fonts, also Schriftarten abgespeichert.
- Ordner „`media`“: Hier drin befinden sich alle Videos und Bilder für die Benutzeroberfläche. Wir bauen die Oberfläche in erster Linie aus einigen Bilder auf, die jeweils einzelne Elemente der Benutzeroberfläche darstellen. Beispielsweise wird die Rezeptauswahl aus zwei Bildern erstellt: Abbildung 27 zeigt die ausgewählte Schaltfläche, Abbildung 26 eine normale Schaltfläche. Daraus wird eine komplette Liste erstellt, wie in Abbildung 28 zu sehen ist.

Abbildung 27: ausgewählte Schaltfläche

Abbildung 26: normale Schaltfläche

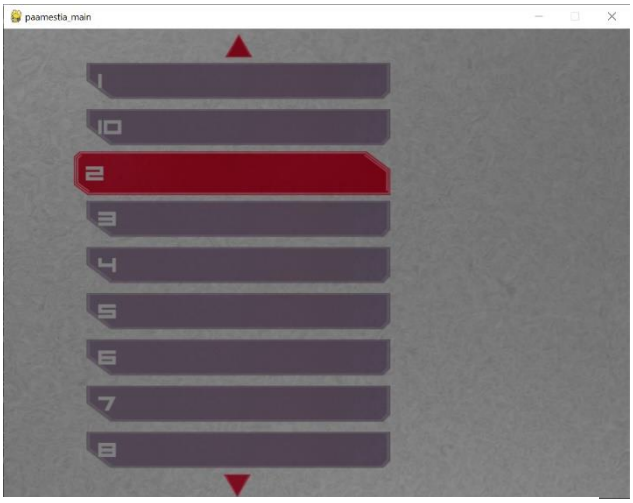

Abbildung 28: fertige Benutzeroberfläche

- Ordner „`recipes`“: In diesem Ordner sind alle Rezepte, die Importierten miteingeschlossen, abgespeichert. Die Rezepte sind einfache Textdateien und beschreiben, welche Getränke in welcher Menge benötigt werden. Diese Dateien werden gelesen, und zum Beispiel bei der Rezeptauswahl aufgelistet. Ausnahme ist hierbei die Datei „`free_mixed_recipe`“, welche das Mischverhältnis beim freien Mischen zwischenspeichert.
- Dateien „`drinks`“ und „`drinks_custom`“: In diesen beiden Dateien werden die verfügbaren Getränke abgespeichert. „`drinks`“ beinhaltet dabei eine vorgefertigte Liste, in „`drinks_custom`“ werden zusätzliche Getränke abgespeichert.
- Datei „`.windows`“: Diese Datei wurde zu Entwicklungszwecken erstellt. Anhand dieser Datei erkennt das Programm, ob es auf einem PC oder dem Raspberry Pi ausgeführt wird. Dies ist notwendig, weil das Programm während des Entwickelns auf einem Windows-PC ausgeführt wurde, im fertigen Projekt aber auf einem Raspberry Pi mit Linux. Ob das Programm in Windows oder in Linux ausgeführt wird, wird in der Variable „`os_is_linux`“ (in „`globals.py`“) gespeichert. Wenn ein Befehl ausgeführt wird, der nur auf dem Raspberry Pi ausgeführt werden kann, wird erst mittels dieser Variable das System überprüft. Auf diese Weise kann es zum Beispiel nicht passieren, dass das Programm versucht ein Magnetventil am PC anzusteuern.

Modulnummer			**7**
Modulname			Entwicklung der Benutzeroberfläche
Ziel des Moduls			**Entwerfung und Entwicklung einer**
Veranschlagte Zeit	So viel wie nötig / Heimarbeit	**Benötigte Zeit**	300h
Mitwirkende	Dominic Kosin		
Benutzte Software			
Blender			**Krita**
Zbrush			**Substance Designer**
Maya			**Inkscape**

5.10.3. VORWORT

Die Benutzeroberfläche hat, im Laufe des Projektes, mehrfach drastische Entwicklungen durchlaufen, jedoch im Kern immer die gleiche Idee verfolgt. Die Software, die zur Erstellung dieser Benutzeroberfläche benutzt wurde, behandelt ein für viele Menschen unbekanntes Thema was zur Folge hat, dass der Großteil der Leser dieser Dokumentation das Folgende Kapitel nur teilweise oder kaum verstehen können. Um dies vorzubeugen, folgt nach diesem Vorwort ein Glossar, was die wichtigsten Begriffe rund um das Themenfeld 3D-Modeling und Rendering erklärt.

5.10.4. BENUTZEROBERFLÄCHE

Grundlagen des 3D-Modelings

Software, die für das Themengebiet des 3D-Modelings entwickelt wurde, hat immer den gleichen Grundgedanken. Nämlich die Darstellung, Erzeugung und Manipulation von Objekten (Primitive) im 3-dimensionalen Raum. Diese Objekte können nur ein Einzelner Punkt (Vertex Mehrzahl Vertecies), eine zweidimensionale Linie (Edge) oder ein Dreidimensionales Objekt (Polygon) sein. Diese Objekte können dann mit den Software-spezifischen Funktionen manipuliert werden, um jedes beliebige Objekt zu erschaffen. Zwei Vertecies bilden eine Edge und mindestens 3 miteinander verbundene Edges ergeben ein Polygon. In der Industrie werden heutzutage nur Dreieckige, Viereckige und Fünfeckige Polygone benutzt. Man Unterscheidet grundsätzlich unter zwei Arten des 3D-Modelings. Das „Hard-Surface-Modeling" und das „Sculpting". Beide Methoden haben ihre Vor- und Nachteile und somit auch spezifische Anwendungsbereiche. „Hard-Surface-Modeling" wird dann verwendet, wenn man ein Objekt erstellen will, was nicht organisch ist wie z. B. ein Haus, ein Stuhl oder eine Brieftasche. Dabei manipuliert man Vertecies, Edges und Polygone, um seine gewünschte Form zu bekommen. Das Sculpting hingegen wird für das Erstellen von organischen Objekten gebraucht. Sie können sich dies als virtuelles Tonformen vorstellen, wo der Künstler mit verschiedenen Werkzeugen die Vertecies, Edges und Polygone manipuliert, bis die gewünschte Form entsteht.

Wireframe (Gitternetz)

Das Wireframe wird durch die unterschiedlich verbundenen Vertecies, Edges und Polygone gebildet, die so die Oberfläche des Objektes darstellen. Es ist sehr wichtig, während der Erstellung von 3D Objekten, ein für den Verwendungszweck passendes Wireframe zu erstellen, um später beim Weiterverarbeiten des Objektes nicht auf Komplikationen zu stoßen. Dies wird in den meisten Fällen durch konsequente und saubere Arbeit geregelt, und erfordert deswegen aber auch Planung, Geduld, Übung und sehr viel Zeit.

UV-Mapping

UV-Maps werden erstellt, um Objekte mit einer Textur zu versehen. Diese Maps sind zweidimensionale Bilder, die die Position der Vertecies, Edges und Polygone speichert. Eine Besonderheit dieser UV-Maps ist auch die Eigenschaft, die Tiefe der einzelnen Polygone, Edges und Vertecies zu speichern. UV-Maps werden in der Regel durch das Setzen von sogenannte „Cuts" und das Aufklappen des 3D-Models erstellt. Eine einfache bildliche Erklärung wäre das Sie einen 3-dimensionalen 6-seitigen Würfel flach auf ein Blatt Papier legen wollen. Dazu schneiden Sie die Kanten an dem Würfel auseinander, bis dieser mit allen 6 Seiten flach auf der zweidimensionalen Fläche liegt.

Node-Based Programming

Ein Großteil der jetzigen Software, die mit 3-dimensionalen Objekten zu tun hat, benutzt für die Manipulierung und auch die Shader Erstellung sogenannte „Nodes". Jede Node hat ihre eigene Funktion und Anwendungsbereich. Diese Art des Programmierens wird auch als flussbasiertes Programmieren bezeichnet.

Physically Based Rendering

Pysically Based Rendering, kurz PBR, ist eine computergrafische Herangehensweise, Modelle fotorealistisch darzustellen. Diese Technik umfasst das Erstellen von bestimmten Texture-Maps, die gewisse Informationen zur Fotorealistischen Berechnung von Physikalischen Eigenschaften enthalten.

High Dynamic Range Images

Kurz HDRIs versteht man verschiedene Techniken zur Aufnahme und Wiedergabe von Bildern mit großen Helligkeitsunterschieden ab etwa 1:1000. Diese werden zur Realistischen Beleuchtung von 3-dimensionalen Szenen benutzt, wo jeder Pixel in Abhängigkeit seiner Helligkeit einen Lichtstrahl auf die 3D- Szene projiziert.

Path Traced Rendering

Beschreibt einen Algorithmus zur Bildsynthese, der die Simulation der globalen Beleuchtung ermöglicht. Dies erfolgt durch rigorose mathematische Verfahren, die Strahlen in die Szene schicken. Diese Strahlen werden dann entweder reflektiert, gebrochen oder absorbiert, wobei jedes Mal ein zufälliger Strahl generiert wird. Die auftretenden Fehler werden als Varianz oder Bildrauschen dargestellt. Diese Varianz wird auch Noise genannt.

Postprocessing

Postprocessing beschreibt die Weiterbearbeitung und Aufbereitung von computergenerierten, synthetischen Bildern.

5.10.5. BENUTZTE SOFTWARE

Die benutzte Software muss zum größten Teil selbst finanziert werden. Die Wahl fiel auf diese Software, da so ein Reibungsloser und Komplikationsfreier Workflow gewährleistet war. Es wurde die Software „Blender" und die integrierte Renderengine Cicles, Maya, die Renderengine Arnold, Substance Designer, Krita, Inkscape, Zbrush, und Pixlr verwendet. Davon waren Maya und Blender und Zbrush die hauptsächlichen Programme zum Erstellen und Rendern der 3D-Modelle. Krita, Inkscape und Pixlr wurden zum Bearbeiten und Erstellen von Zeichnungen und Bildern genutzt. Substance Designer ist hauptsächlich nur zum Erstellen von PBR-Texturen nützlich.

5.10.6. WORKFLOW

Durch die diverse Menge an unterschiedlichen Programmen, die zur Erstellung der Benutzer Oberfläche verwendet wurden, war eine Einteilung der Programme nötig, um den effizientesten Weg zur Fertigstellung zu finden. Der Workflow setzte sich aus den verschiedenen Schwächen und Stärken der jeweiligen Programme zusammen. Ein weiterer Faktor für die Zusammensetzung war die eigene Erfahrung im Umgang mit den Programmen. Der persönliche Stand vor dem Start des Projektes lag bei der fortgeschrittenen Nutzung der Programme Blender, Inkscape, Pixlr, Zbrush und Krita. Im Laufe des Projektes wurden diese Fertigkeiten

vertieft, und den Umgang mit den Programmen Maya und Substance Designer selbstständig erlernt.

5.10.7. FEHLGESCHLAGENE VERSUCHE UND UNBENUTZTE OBJEKTE

5.10.7.1. GRUNDLEGENDE IDEE

Die Grundlegende Idee dieser Benutzeroberfläche war es, ein Menü bestehend aus verschiedenen Bereichen zu erstellen. Diese Bereiche hätten dann eine bestimmte Funktion, die dem Kunden die Maschine bedienen lassen würde. Diese Idee ging noch weiter, denn da unsere Maschine einer Bar nachempfunden wurde, sollten die verschiedenen Bereiche von einem virtuellen Barkeeper, der in einer Bar arbeitet, begleitet werden, so wie auch Übergänge einleiten und den Kunden die Wartezeit mit Witzen verkürzen sollte. Der Hintergedanke war so die Benutzung der Maschine menschlicher und persönlicher zu gestalten. Die Funktion und der Inhalt dieser Bereiche wurden schon nach der Festlegung des Projekt-Themas aufgegriffen und bestimmt.

5.11.7.2. DIE VIRTUELLE BAR PAAMESTIA

Diese Version erstellte ich daraufhin in dem Zeitraum der kompletten Winterferien 2020 und ein paar Wochen stückweise danach. Schätzungsweise verwendete ich, bis zu dem Stand der letzten Änderung, 200 Stunden nur mit dem Designen und Erstellen der 3D-Modelle für die begleitenden Videos des Paamestia-Automaten. Die Vorgehensweise, die ich beim Designen und Erstellen dieser Version gewählt habe, war rückblickend sehr ineffizient und langsam. Des Weiteren habe ich die meisten Ideen für das Design während des Erstellens anderer Objekte und im Schlaf bekommen. Dies verlangsamte den Erstellungsprozess immens, denn durch die nicht grobe Festlegung eines Designs durchliefen fast alle Modelle mehrere Iterationen, wo diese Modelle, von Grund auf, neu entwickelt worden. Der Workflow für diese Modelle war immer der gleiche. Die Haupterstellung wurde in dem Programm Blender gemacht, da ich mit diesem Programm die meiste Erfahrung hatte. Trotz dessen musste ich bei Modifikationen, um das gewünschte Objekt zu kreieren, auf das Modeling spezialisierte Programm Zbrush zurückgreifen, da dieses mit Problemen klarkommt, wo das Programm Blender Fehler aufweist. Insgesamt besteht die komplette Szene aus 278 Modellen. Um sich etwas von solchen Ausmaßen auszudenken zu können, muss man sich im Klaren sein was sein Endprodukt sein sollte. Deshalb bereitete ich ein Szenario vor, wo diese Bar spielen sollte, und schrieb dazu eine Geschichte, die diese Bar glaubwürdiger machen sollte. Die Bar selbst würde in einem Gebäude stehen, was gleichzeitig ein Treffpunkt für allerlei Menschen mit unterschiedlichen Hobbys ist. Merkmale sollten die Bücherei, die Zentrierte Bar, die Bühne, Form und Architektur sein. Die Haupt-Inspiration war die Viktorianische und Anthroposophische Architektur.

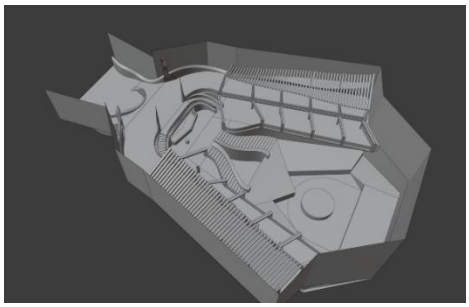

Abbildung 30: Seiten-Ansicht

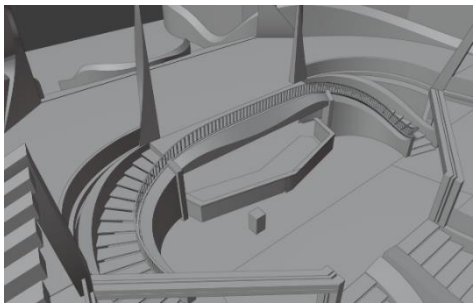

Abbildung 29: Close Up Bar

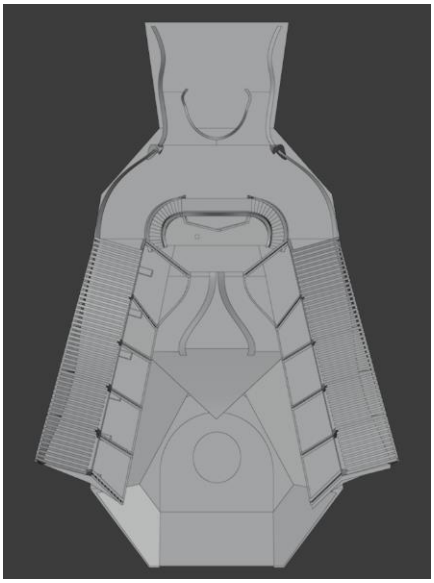

Abbildung 31: Draufsicht

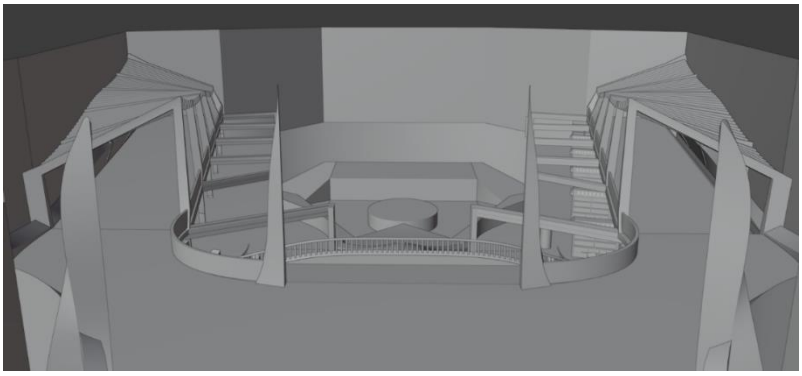

Abbildung 32: Vogelperspektive

5.11.7.3. GRÜNDE DES SCHEITERNS

Die Zeit nach den Winterferien war vollbepackt mit vielen anstrengenden und langwierigen schulischen Arbeiten, die die Zeit so strapazierten, dass keine Zeit mehr für die Weiterentwicklung der virtuellen Bar Paamestia zur Verfügung war. Des Weiteren wäre diese Projektidee, meinerseits eine utopische, nur möglich gewesen, wenn man alle anderen schulischen Aktivitäten ruhen lassen hätte. Trotz dieses Fehlschlages nahm ich einiges aus der Entwicklungszeit mit und fing an eine neue Idee zu entwickeln.

5.11.8. PAAMESTIA AUTOMATON

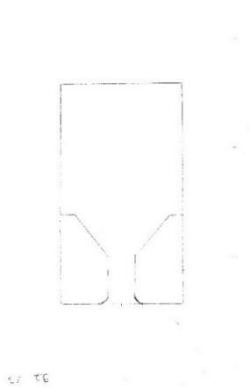

Abbildung 33: Iteration 1 Automaton

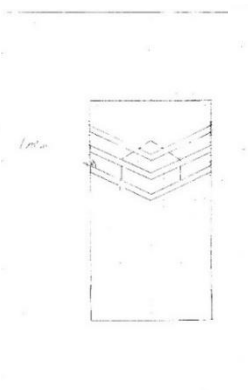

Abbildung 34: Iter. 2 Automaton

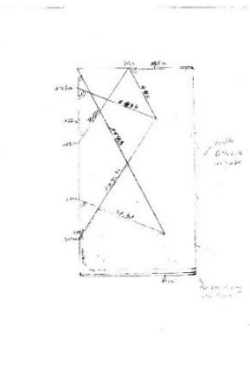

Abbildung 35: Iter. 3 Automaton

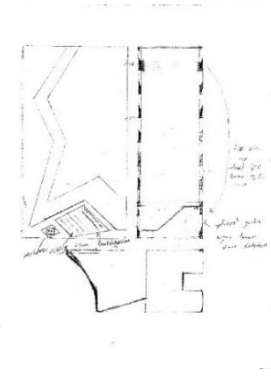

Abbildung 36: Rückseite

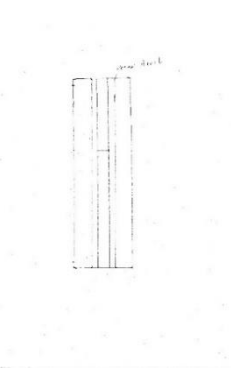

Abbildung 37: Seitenansicht

Die alte Idee wurde verworfen und ich entwickelte eine neue Idee. Diesmal war die eigentliche Benutzeroberfläche im Vordergrund. Daher sollte die Auswahl eher an einen Getränkeautomaten erinnern. Die Idee war, das beim Auswählen des gewünschten Rezeptes eine Animation abspielen würde, die dem Kunden das Getränk visuell erklärt und schmackhaft macht. Des Weiteren sollten beim Bereich „Import Recipe" der eigentliche Automat zu sehen sein und in dem Bereich „Select Drinks" die Getränkesorte durch Farben und gegebenenfalls Früchten visualisieren. Ich fing diesmal anders an, und entwarf erste Konzepte für den Automaten, dem Herzstück des Programmes. Nach einigen Iterationen fand ich das richtige Design und verfeinerte es, bis es fertig zum Modellieren war.

Das Model wurde diesmal zum größten Teil in dem Programm Zbrush entwickelt, da ich durch die virtuelle Bar Paamestia meine Fähigkeiten in diesem Programm stärken konnte. Das Grundgerüst wurde jedoch in dem Programm Blender erstellt. Dort habe ich dann einfach Vertecies an den Punkten, die ich in dem Entwurf niedergeschrieben habe, platziert, verbunden und zu einem Objekt zusammengefügt. Ich hielt mich strickt an meinen Entwurf und setzte ihn genauso um. Zwischendurch musste das Objekt optimiert werden, da es angedacht war dieses Objekt zu Texturieren. Ich entschied mich dafür, das Objekt in einem Wireframe zu erstellen, welches komplett aus Vierecken besteht, da das Model eine sehr komplexe Struktur aufweist. Nach etwa 50 Stunden war das Model komplett fertig und bereit zum Texturieren. Doch zuvor musste das Modell UV-Unwrapped werden. Meine Entscheidung, das Wireframe komplett aus Vierecken zu machen und die Arbeit, das Modell zu optimieren, zeigten sich hier als sehr nützlich. Das UV-Unwrapping wurde in dem Programm Maya vollzogen, da es sehr hilfreiche Funktionen für diesen Bereich zur Verfügung stellt.

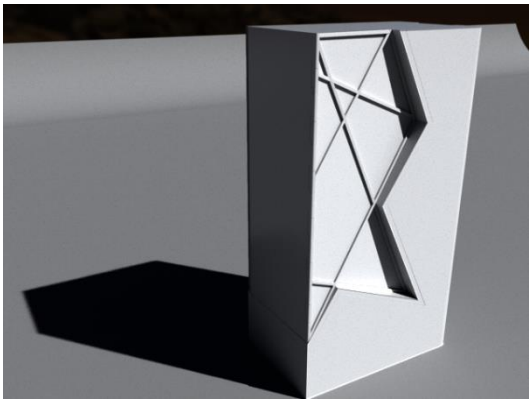

Abbildung 38: Automaton Render

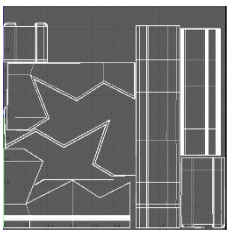

Abbildung 39: UV-Map 1

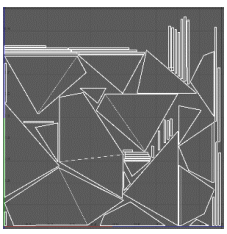

Abbildung 40: UV-Map 2

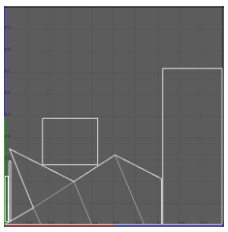

Abbildung 41: UV-Map 3

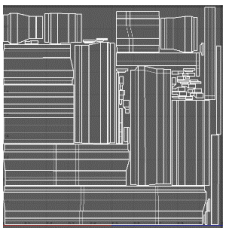

Abbildung 42: UV-Map 4

Gründe des Scheiterns

Nach Fertigstellung des Modelles und Beendigung der Vorbereitung zum Texturieren wurde die Menge an zu erledigenden Hausarbeiten für bestimmte Fachrichtungen so massiv, dass wieder keine Zeit für die weitere Entwicklung des Projektes übrig war. Als sich dies jedoch wieder legte, war die übrige Zeit bis zur Abgabe sehr begrenzt und ich entschied mich dazu, diese Stelle ruhen zu lassen, bis die Benutzeroberfläche fertig ist.

5.11.9. BENUTZEROBERFLÄCHE

Logo

Die generelle Idee, wie das Logo aussehen sollte stand schon seit langem fest. Nämlich seit der virtuellen Bar „Paamestia“. Während dieser Zeit entwickelte ich ein Logo und erstellte sogar schon eine „Titelbild”-Szene. Es sollte eine Art Blume sein, die uns und unser Bestreben, in irgendeiner Weise, repräsentiert. Nach dem ich den Paamestia Automaten habe ruhen lassen, machte ich mich an die Entwicklung eines Logos. Dies zeichnete ich in dem Programm Krita, mehrere Male in verschiedenen Ausführungen, bis ich zufrieden mit dem Design war. Um das Design zu optimieren, zeichnete ich die vorgemalten Linien mit einem gewichteten Stift nach, der die Linien konstant bleiben lässt. Das fertige Bild importierte ich dann in das Programm Inkscape, wo ich es weiterverarbeite damit ich es in einem 3D-Programm verwenden kann. In Inkscape verwendete ich eine Funktion, die die Bitmap eines Schwarz-Weiß-Bildes nachzeichnet und so eine SVG-Datei erstellt. Diese Datei kann ich dann in Maya importieren, um dort die Szene für das Rendern (Bildsynthese) vorzubereiten.

Abbildung 43: Logo Paamestia

Sequence

Nachdem das Logo fertig war, begann ich eine Startsequenz zu animieren. Es sollte eine Hommage an den gedachten Automaten sein, also war die Idee ein Computer-Boot zu animieren. Die fertige Animation war in zwei getrennt gemachte Sequenzen geteilt, die beide in 24 Frames die Sekunde laufen. Zuerst der Computer-Boot und dann die Ladeanimation. Glücklicherweise stellte ich fest, dass das Logo selbst ein interessantes Ladezeichen ergeben würde, also nahm ich das fertige Logo auch dafür. Den Ladebalken erstellte ich aus einem Quadrat, wo ich die Ecken abrundete, die vorderste und hinterste Fläche modifizierte und schließlich löschte. Das Logo und dem Ladebalken gab ich einem programminternen PBR-Shader, den ich durch gezielte Einstellungen weiß und reflektiv gestaltete. Zusätzlich zu dem Shader platzierte ich 6 unterschiedlich starke „Area Lights” die die Modelle beleuchten sollten. Die Kamera stellte ich auf die gewünschte Größe des Bildes und platzierte sie so, dass alles im Bild ist und stellte die Länge der Animation in der Zeitlinie ein.

Für die Animation des Logos setzte ich Key-Frames an verschiedenen Punkten in der Zeitlinie. Die Animation des Logos beginnt erst ab dem 25. Frame und endet beim 144. Frame. Die Key-Frames wurden so periodisch gesetzt und eingestellt, dass das Logo sich 720 Grad in 23 Frames bewegt und dann dies nach 10 Frames wiederholt. Der Graph, der diese Bewegung darstellt, wurde so eingestellt, dass diese Bewegung nicht linear von statten geht.

Das Model des Ladebalkens wurde in 4 verschiedene Flächen aufgeteilt und so mit Key-Frames animiert, das diese Flächen zu einem bestimmten Zeitpunkt auf der Zeitlinie sichtbar für die Kamera werden. Für zusätzlichen Realismus wurde Motion-Blure aktiviert, was den Shutter-Speed von Digitalkameras simuliert.

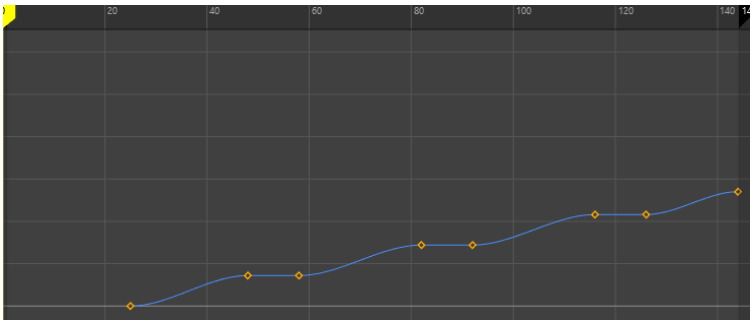

Abbildung 44: Keyframes Logo

Diese fertige Animation wurde dann in 144 verschiedenen Bilder gerendert und später in eine Video-Sequenz zusammengefügt.

Den Inhalt der Booting-Sequenz wurde auf einen Zettel niedergeschrieben und anschließend in einem 3D-Programm in digitalen Text konvertiert. Diese wurden dann als Zeilen zusammengefügt und in Maya importiert. Der PBR-Shader ist genau derselbe wie bei dem Ladebalken. Die Animation geht insgesamt 312 Frames. Der erste Teil der erstellten Animation beginnt ab dem Frame 75 und läuft bis zum Frame 203. In diesem Zeitraum tauchen in unregelmäßigen Abständen Buchstaben auf die den Satz „project paamestia:start" ergeben. Dies soll das Gefühl vermitteln, das ein Mensch diesen Befehl in die Maschine eingibt und Interesse wecken. Ab dem Frame 240 Tauchen dann, in periodischen Abständen die vorher festgelegten Linien auf. Ab dem Frame 260 kommt dann auch noch eine periodische Bewegung der Linien, um einen bestimmten Wert, um Scrollen zu simulieren. Diese beiden periodischen Ereignisse passieren alle zwei Frames einmal.

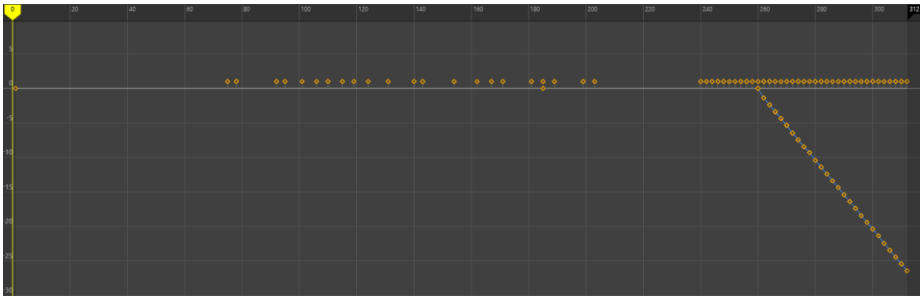

Abbildung 46: Keyframes Booting Sequence

Abbildung 45: Render Logo und Ladebalken

Diese Animation wird auch wieder in 312 Bildern gerendert.

Anschließend wurden beide Sequenzen in dem Videoschnittprogramm Shortcut zusammengefügt.

Komposition der Benutzeroberfläche

Wie diese Bereiche der Benutzeroberfläche nun genau aussehen sollten, konzipierte ich zuallererst, um mir einen groben Überblick über das gesamte Ausmaß zu verschaffen. Dieses

Konzept war lediglich eine reine Gedankenstütze und kann nicht mit dem finalen Ergebnis gleichgesetzt werden.

Generelle Vorgehensweise zur Umsetzung des Designs

Alle Designs der Benutzeroberfläche, die jetzt folgen werden, wurden mit den gleichen Techniken erstellt. Die Konzepte wurden in dem Programm Krita erstellt und waren hier wieder nur eine Gedankenstütze und können nicht mit dem finalen Produkt gleichgestellt werden. Trotzdem wurden diese Konzepte für die Erstellung der Modelle gebraucht. Die meistgenutzte Technik, die zum Erstellen dieser Modelle gebraucht wurde, ist die des Platzierens von Vertecies an den vermeintlichen Positionen und diese dann zu einem Polygon zu verbinden. All die Details und Muster wurden mithilfe der „Inset"-Funktion und Boolean-Operatoren erstellt. Bei der Inset-Funktion werden die Edges eines Polygons kopiert und können beliebig in dem Polygon oder außerhalb platziert werden. Werden diese jedoch in dem Polygon platziert, wird das Polygon geteilt und es Entstehen weitere Edges. Diese können dann auch wieder mit verschiedenen Operatoren manipuliert werden, um das gewünschte Objekt zu bekommen. Die Boolean-Operation hingegen entfernt die geschnittene Fläche eines Operators mit einem anderen Objekt.

Schriftart

Die benutzte Schriftart auf der Benutzeroberfläche ist „Da Mad Rave" von Darell Flood. Diese wurde auf der Webseite Dafont.com heruntergeladen und ist für nicht kommerziellen Gebrauch kostenlos.

wichtigste Komponenten der Benutzeroberfläche

Im Grunde besteht die gesamte Benutzeroberfläche aus einem sich wiederholenden Muster von Grundformen, die ich während des Design-Prozesses entwickelt habe. Das generelle Aussehen dieser Flächen wurde von dem zuvor entwickelten Paamestia-Automaten stark beeinflusst. Zum Zweck der effizientesten Arbeitsmethode wurde oft die Form verschiedener Objekte wiederverwertet. Dies wird bei genauerer Betrachtung in dem Bereich der Listen deutlich. Dort wurde die „normale Fläche", die Form der „ausgewählten Fläche" und den „Button" mehrfach in unterschiedlichen Aufmachungen genutzt. Diese unterschiedlichen Aufmachungen wurden, nach dem Fertigstellen der Grundform, durch reines Ausprobieren und Manipulieren erstellt.

Main Menu

Das Main Menu beinhaltet die essenziellen Flächen, die benötigt werden, um durch das Programm zu gelangen. Durch genauere Absprache mit unserem Programmierer Arvid Randow hatte ich die nötigen Informationen, wie die wirkliche Programmierung aussah und entwickelte daraufhin zugeschnittene Formen für diese Anwendung. Das fertige Design lässt dem Kunden diese vier verschiedenen Flächen ansteuern, was mit einem Farbwechsel der ausgewählten Flächen indiziert wird. Des Weiteren wurden die Flächen mit dem korrespondierenden Beschreibungstext versehen. Die gewählte Sprache für die Benutzeroberfläche war Englisch, aus dem einfachen Grund, dass diese Sprache kürzere Wörter für die Schlüsselwörter

der Flächen enthält und so die Möglichkeit bereitstellte, eine größere Schriftgröße zu Verwenden.

Die Hauptfläche „Recipes", anders als die anderen Flächen zu Designen, hatte den Hintergedanken, so diese Fläche von dem Rest abzuheben und bei dem Benutzer Interesse zu wecken.

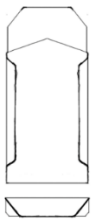

Abbildung 48: Iteration 1 Recipes

Abbildung 49: Iter. 2 Recipes

Abbildung 47: Flügel Zeichnung

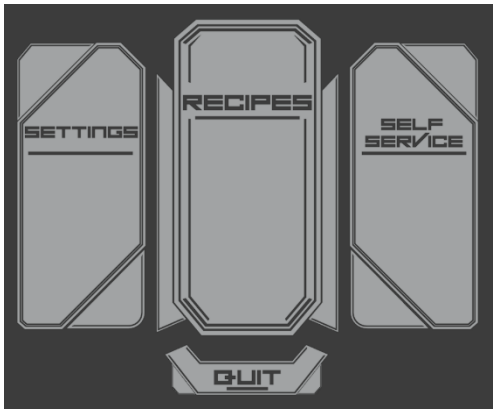

Abbildung 50: fertige Ansicht Main Menu

Settings

Um Ressourcen und Zeit zu sparen, wurde hier die linke und rechte Fläche des Main Menus wiederverwendet. Einen weitgreifenden Hintergedanken bei dem Design gab es auch nicht, da der Bereich „Settings" nicht im Vordergrund lag. Der Kunde hat in diesem Bereich die Möglichkeit, in die weiteren Bereiche „Select Drinks" und „Import Recipes" zu kommen. Dies wurde wieder durch einen Farbwechsel der ausgewählten Fläche indiziert.

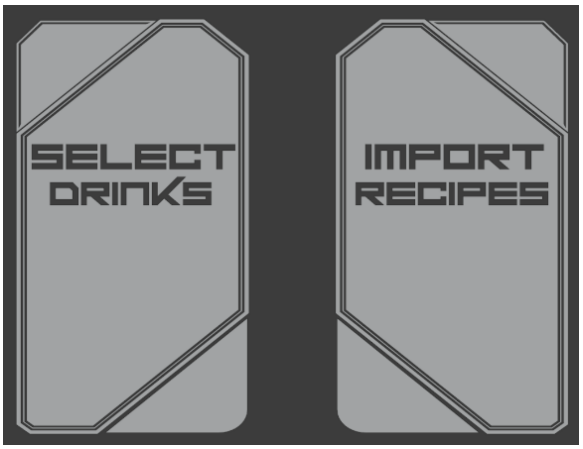

Abbildung 51: fertige Ansicht Settings

Select Drinks

Die Hauptaufgabe dieses Bereiches war die Festlegung der verwendeten Getränke, um die verschiedenen Rezepte, und auch den Bereich „Self Service" korrekt benutzen zu können. Ohne diese Festlegung würde das Programm die Fertigstellung der vorprogrammierten Rezepte nicht initiieren und den Bereich „Self Service" komplett unbrauchbar gestalten. Damit der Benutzer volle Kontrolle über das Festlegen, welche Getränke er in welchem Slot anschließt, verfügt, hat er die Möglichkeit, hier in diesem Bereich das benutzte Getränk in der von uns voreingestellten Liste zu suchen und es im Programm abzuspeichern. Der Slot kann durch Auswahl der Nummer rechts von der ursprünglichen Auswahl-Fläche geändert werden. Die ausgewählte Fläche wird auch hier wieder mit einem Farbwechsel indiziert, die Ausnahme hierbei stellen die Pfeile in den verschiedenen

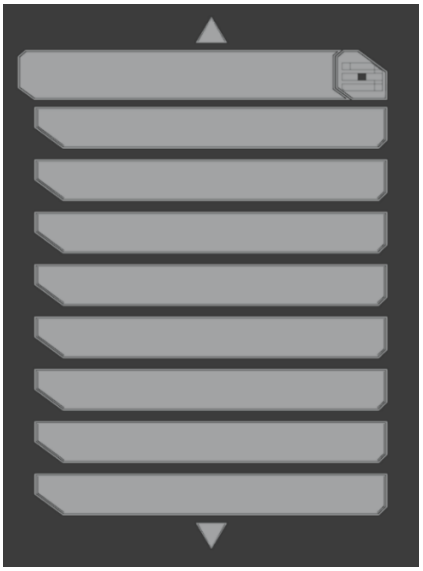

Abbildung 52: fertige Ansicht Select Drinks

Listen dar. Diese sind von Anfang an rot und werden erst grau, wenn diese das Ende der jeweiligen Seite der Liste erreicht hat.

Import Recipes

Zum Zweck der Erweiterung hat der Benutzer die Möglichkeit eigene Rezepte in die Maschine zu importieren. Dies wird in dem Bereich „Import Recipes" durchgeführt. Das Design was hier für entwickelt wurde, besteht zum Teil aus den gleichen Flächen wie die Liste in dem Bereich „Select Drinks". Der signifikante Unterschied hier ist die Positionierung dieser Listen, die Entfernung der Slot-Nummer und den neu eingefügten „Delete" und „Import" Buttons. Die ausgewählte Fläche wird hier auch wieder mit einem Farbwechsel indiziert. Der Kunde hat nur die Möglichkeit seine eigenen importierten Rezepte zu löschen. Die von uns festgelegten Rezepte sind fester Bestandteil der Maschine und dürfen nicht gelöscht werden.

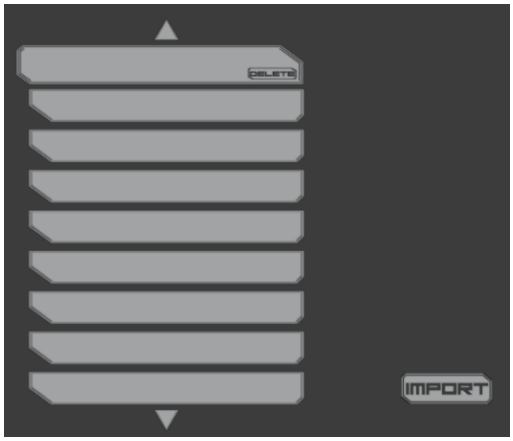

Abbildung 53: fertige Ansicht Import Recipes

Self Service

Da wir uns nicht nur auf die Herstellung von Getränken nach Rezepten spezialisieren wollten, kam uns die Idee, den Kunden die Möglichkeit zu geben, sich eigens die gewünschte Menge eines oder mehrerer angeschlossenen Getränke zubereiten zu lassen. Der Kunde hat in dem Bereich „Self Service" die Möglichkeit, an den verschiedenen Slots das voreingestellte Getränk in 10% Schritten, bis das vorgegebene Limit der 100% erreicht ist, einzustellen und auszuführen. Diese Prozentanzeige wird einmal mit Strichen in den korrespondierenden Slots und einer Nummer angezeigt. Die Slots werden hier als Türme dargestellt, die die gleiche Nummerierung wie die eingestellten Getränkeslots besitzen. Die ausgewählte Fläche wird hier wieder mit einem Farbwechsel indiziert.

Abbildung 54: Iteration 1 Prozent

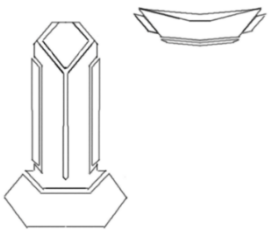

Abbildung 55: Iter. 2 Prozent

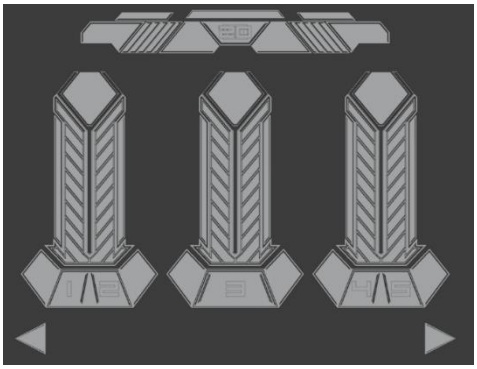

Abbildung 56: fertige Ansicht Self Service

Recipes

Dies war der Bereich, der unsere Hauptidee eines Getränkeautomaten widerspiegelt. In dem Bereich „Recipes" hat der Kunde die Möglichkeit von einer von uns erstellten erweiterbaren Liste eine Vielzahl von unterschiedlichen Rezepten eins auszuwählen und diese von unserer Maschine mischen zu lassen. Hier wurde sich auch wieder an dem ursprünglichen Listen-Design bedient und leicht umgewandelt. Es wurde auch dieselbe Aufmachung wie beim Bereich „Import Recipe" verwendet, weil diese sich hierfür sehr gut geeignet hat. Besonderheit dieses Bereiches ist, dass die Rezepte erst funktionieren, wenn die für die Herstellung benötigten Getränke in dem Bereich „Select Drinks" eingestellt sind, um so eine falsche Benutzung zu verhindern. Des Weiteren erscheint ein Informationstext, sobald der Kunde Interesse an einem Rezept zeigt und dieses auswählt. Die Auswahl wird hier wieder mit einem Farbwechsel der ausgewählten Fläche indiziert.

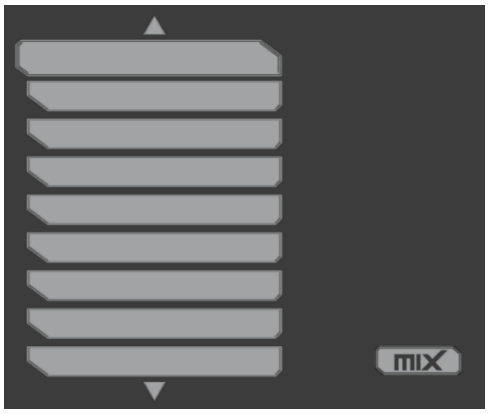

Abbildung 57: fertige Ansicht Recipes

Texturierung der Benutzeroberfläche

Das Ziel war es, eine Realistische PBR-Texture in dem dafür bestimmten Programm „Substance Designer" zu erstellen und diese dann auf die für die eigens erstellten Modelle zu benutzen. Das Programm „Substance Designer" benutzt eine Node-Based Programmierung und war deswegen sehr einfach zu bedienen. Um möglichst realistische Texturen zu erstellen, ist es nötig mehrere unterschiedliche Texture-Maps zu erstellen, die verschiedenen Informationen für die physikalisch korrekte Darstellung in dem Vorgesehenen Render enthalten. Die Texture sollte unser verwendetes Aluminiumblech nachahmen. Im Folgendem werde ich die Node-Based-Programmierung für die Erstellung der Texturen erklären. Für die Erstellung der Textur bearbeitete ich hauptsächlich die Roughness-Map ‚die die Rauheit der Fläche beeinflusst und die Normal-Map, die die Position der verschiedenen Normalvektoren der einzelnen Flächen um einen festgelegten Wert verschiebt. Um diese „Maps" zu kreieren benutzt man 16 Bit Greyscale Images. Diese werden dann mit verschiedenen Operatoren und Algorithmen modifiziert, bis das gewünschte Ergebnis erreicht wird. Um das Erklären der Programmierung zu vereinfachen, werden die einzelnen Effekte aufgeteilt.

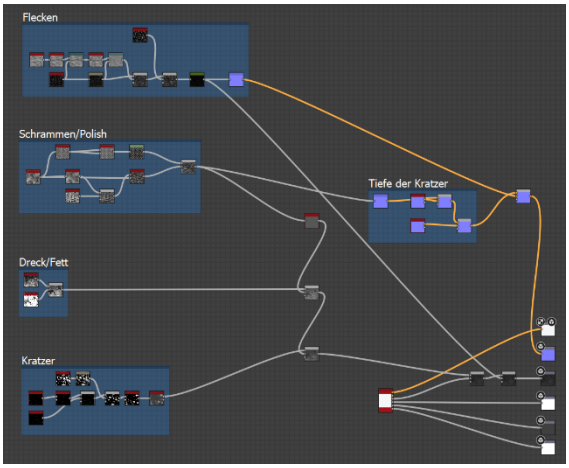

Abbildung 58: PBR Überblick

Zu Beginn wurde eine Voreinstellung, die das Erstellen solcher PBR-Texturen vereinfacht, benutzt. Diese Voreinstellung kreierte die nötigen Outputs, um in diesem Fall ein Aluminium-Blech darzustellen. Das Programm übernahm dafür die korrekte Erstellung der Albedo-Map, was die Farbe des Metalls bestimmt, der Metallic-Map, die die metallischen Eigenschaften der Texturen beeinflusst, die Heigth-Map, die, wenn nötig, die Tiefe oder Höhe der enthaltenden Informationen beeinflusst und die Ambient Oclusion-Map , die, wenn nötig, Lichtinformationen zur korrekten Beleuchtung in einer vorher bestimmten Umgebung speichert.

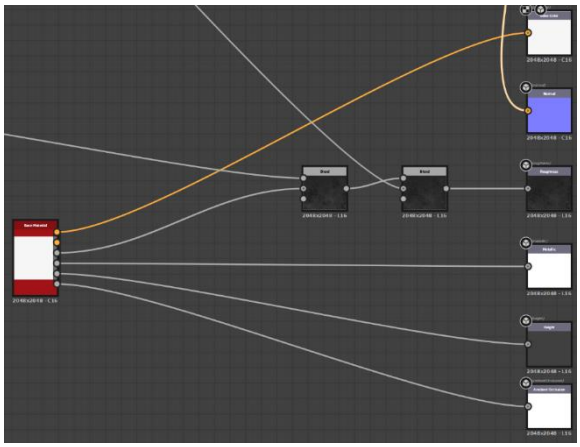

Abbildung 60: Voreinstellungen

Es wurden eine Reihe von verschiedenen Effekten erstellt. Darunter befinden sich Flecken, Schrammen, Fettschichten und Kratzer für die fehlenden beiden Texture-Maps.

Um die Kratzer zu erstellen, benutzte ich einen Algorithmus, der ein Greyscale-Image erstellt, wo Kratzer generiert werden. Von dieser erstellte ich zwei unterschiedliche Versionen, wovon ich die eine mit der Funktion „Slope Blur" modifiziert habe damit diese nicht mehr so stark vorkommt. Diese beiden Bilder wurden dann mit der Funktion „Blend" stark verschwommen und wurden dann mit der Funktion „Blend" zusammengefügt.

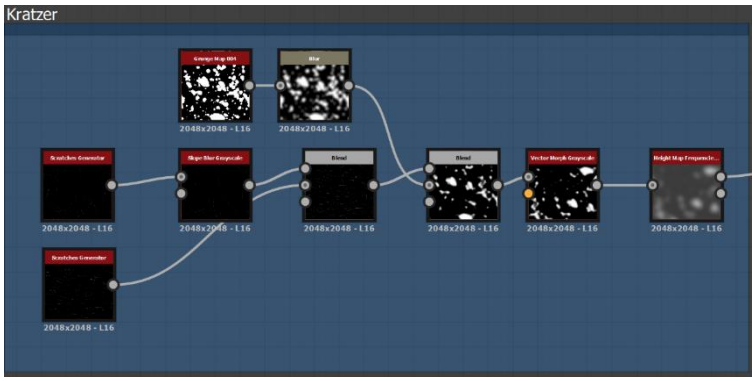

Abbildung 59: Kratzer

Daraufhin benutzte ich eine Grungemap, die zuvor mit der Funktion „Blur" bearbeitet wurde, und fügte diese mit den zusammengefügten Kratzerbildern zusammen. Diese neu erstellte Map wurde dann mit der Funktion „Vector Morph" manipuliert, in dem man das sekundäre Vektorfeld der Textur verändert. Die Texturen wurden dann mit der Funktion „Heigth Map Frequencies" bearbeitet, was zur Folge hat, dass das Bild in zwei verschiedene Texture-Maps transferiert wird. Diese wurden später weiter genutzt.

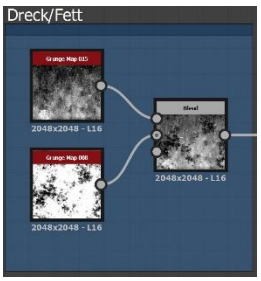

Abbildung 61: Fettflecken

Die Fettflecken wurden mit zwei unterschiedlichen Grunge-Maps erstellt, die ich anschließend miteinander multipliziert habe, um den Effekt dieser zu verstärken.

Um die Schrammen zu erstellen, benutzte ich eine BnW Spots Texture-Map, die ich anschließend mit der Funktion „Safe Transform", „Blur HQ" und „Slope Blur bearbeitete. Dies hatte zur Folge, dass ich jetzt zwei Versionen dieses Greyscale Images hatte. Eine Version, die verschwommen war, und die andere wurde auf der X und Y vervielfältigt, sodass ein Raster entsteht. Das Verschwommene wurde anschließend mit einem Anistropic Noise multipliziert, um den Effekt zu verstärken. Diese wurde dann anschließend durch die Funktion „Slope Blur" verschwommen. Im Anschluss daran wurde dieses Bild mit dem vervielfältigten Musterbild zusammengefügt um dies später weiter zu benutzten.

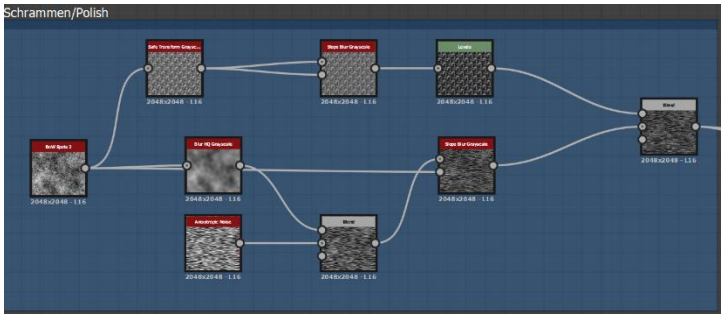

Abbildung 62: Schrammen

Die Flecken wurden mittels einem Directional Noise Greyscale Image erstellt, was später mit einem Dirt und Gaussian Spots Image zusammengefügt wurde. Das Directional Noise Image wurde durch die Funktionen „Invert", „Directional Warp", „Skew" und „Warp" modifiziert. Die Funktion Invert hatte den Effekt die Graustufen zu invertieren. Dieses Bild wurde dann von der Funktion „Directionl Warp" in eine bestimmte Richtung verzehrt. Die Funktion „Warp" hat den gleichen Effekt nur das dieser auf eine andere Art und Weise das Bild verzehrt. Die Stärke dieses Effektes wurde von dem Greyscale Image „Dirt" kontrolliert. „Skew" hingegen transformiert das Bild mit verschiedenen Parametern. Nach dem Zusammenfügen der besagten Bilder wurden diese noch ein letztes Mal mit der Funktion „Curves" bearbeitet, die die Stärke der Graustufen verändert. Für weiterführenden Arbeiten wurde aus diesem Output mit der Funktion „Normal" eine Normal-Map erstellt, die später weiterverarbeitet wurde.

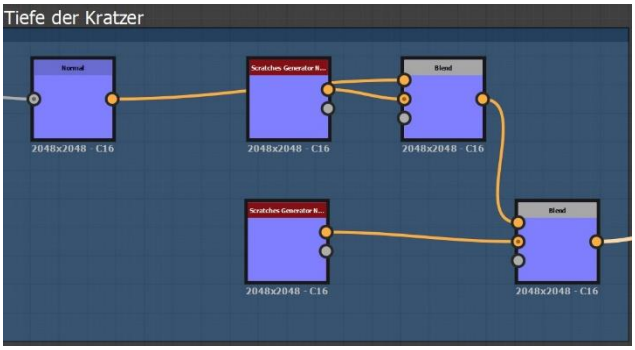

Abbildung 63: tiefe Kratzer

Für den Effekt tieferer Kratzer wurde der Output der Schrammen in eine Normal-Map konvertiert und mit zwei „Scratch Generator"-Funktionen zusammengefügt.

Für die letztendliche Normal-Map wurden der Output der tieferen Kratzer und der Output der Flecken zusammengefügt und in den Material Input eingeführt.

Die Roughness-Map wurde durch die Zusammenführung der Outputs der Schrammen, Fettflecken, Kratzer, Flecken und dem voreingestellten Aluminiums Output hergestellt.

Abbildung 64: PBR Render

Rendering der Benutzeroberfläche

Für das Rendern der Benutzeroberfläche nahm ich die entwickelten Modelle und die PBR-Texturen und importierte beides in das Programm Maya. Jedoch stellte sich bald heraus, dass ein mir bis jetzt noch unbekannter Fehler die Texturen zwar auf programminternen Modellen korrekt darstellt, dies aber nicht tut, wenn es ein importiertes Modell meinerseits war. Dieses Problem umging ich, indem ich zu dem Programm Blender wechselte, und dort die Benutzeroberfläche renderte. Die Texturen wurden in die für sie vorgesehenen Inputs des programmierten Shaders angeschlossen, hatten aber nicht den vorgesehenen Effekt. Die Bildauflösung wurde auf 800*600 Pixel gestellt. Die Flächen wurden alle einzeln gerendert, da dies nötig war, damit das Programm von Arvid mit den Bildern keine Leistungsprobleme erzeugt.

Für die Belichtung der Szene nutzte ich eine HDRI von der Seite HDRI Haven.com. Dort werden verschiedenen HDRIs für nicht kommerziellen Nutzen kostenlos zur Verfügung gestellt. Meine Wahl fiel auf die HDRI„ Satara Night (no Lamps)" da diese das Bild nicht überbelichtete.

Hintergrund

Den Hintergrund erstellte ich in dem Programm Blender, indem ich einen Shader programmierte. Dazu benutzte ich den integrierten Principal BDSF Shader und ließ bestimmte Parameter von Noise steuern. In diesem Fall ließ ich den Wert des Roughness Parameters durch eine Noise-Textur, die von einer Color Ramp abgeschwächt wird und einem Wert bestimmen. Des Weiteren wurde eine Noise Textur und eine Bump-Map zum Steuern des Normal-Map Parameters genutzt. Es entstanden zwei Versionen des Hintergrundes. Die eine verwendete die PBR-Aluminium-Textur und die andere den entwickelten Shader. Das gewählte Model für das Rendern war ein Quadrat.

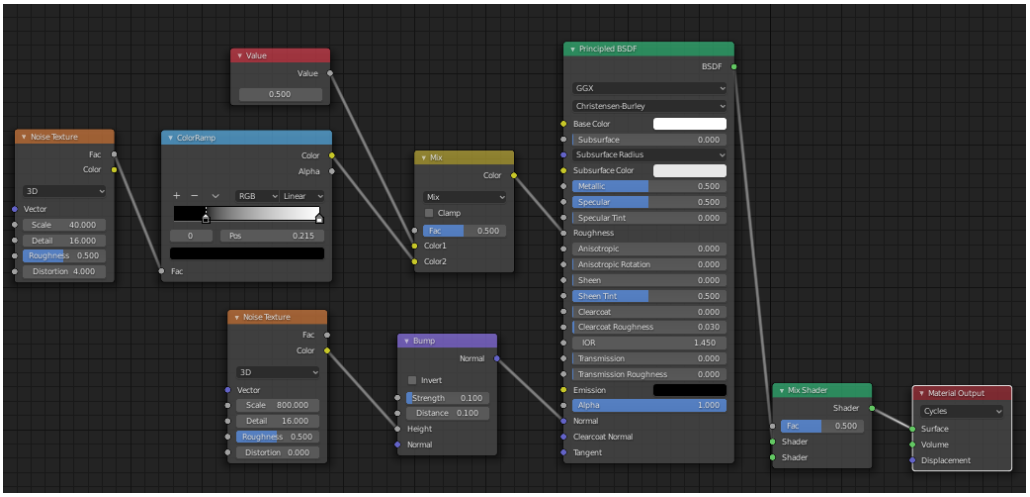

Abbildung 65: Shader Hintergrund

5.11.10. POSTPROCESSING

Benutzeroberfläche

Auf Anfrage unseres Programmierers Arvid mussten die gerenderten Bilder der Benutzeroberfläche und die einzelnen Komponenten auf ihre individuelle Bildgröße zugeschnitten werden. Die Bildgröße wurde mittels Quadrate um die verschiedenen Modelle ermittelt. Dies war nötig, um diese Bilder in dem Programm als eigentliche Benutzeroberfläche zu benutzen und in Echtzeit die verschiedenen Bilder auf die richtigen Positionen zu verschieben. Das Zuschneiden der zugehörigen Flächen erfolgte, in dem Bildbearbeitungsprogramm Pixlr, womit die verschiedenen Bilder importiert und zugeschnitten worden sind.

Die Versionen, die indizieren, dass eine Fläche ausgewählt worden ist, wurde aus den zugeschnittenen Versionen hergestellt. Dazu programmierte ich in dem Programm Blender mit Node-Based-Programming einen Algorithmus, der die Bilder farblich so modifiziert wie dies auch programmiert wurde. Als dieser Algorithmus fertig war, musste man nur noch die gewünschten Bilder als Input angeben und das Bild neu rendern. Dies wurde mit allen erwünschten Versionen gemacht.

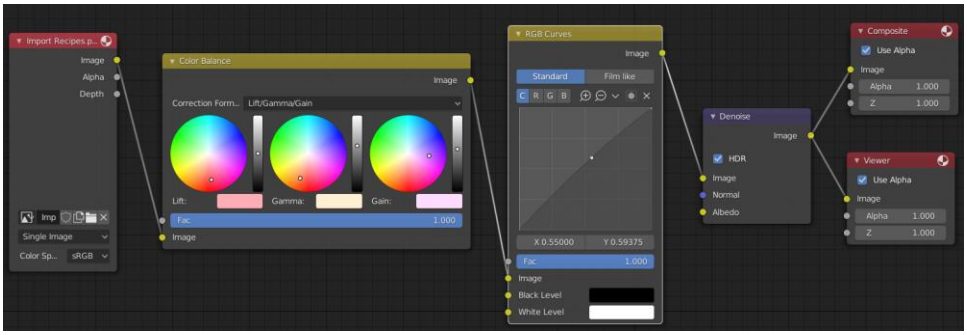

Abbildung 68: Algorithmus zum Ändern der Benutzeroberfläche

Die Programmierung lief wie folgt ab: Das Bild erfährt durch die Funktion „Color Balance" verschiedene Farbveränderungen, die festlegen, wie jeder Pixel dargestellt wird. Dieser Output wird dann danach von der Funktion „RGB Curves" weiterverarbeitet. Diese Funktion verändert auch wieder die Farbverhältnisse der verschiedenen Farbkanäle um den Wert der eingestellten Kurve. Um den entstandenen Noise komplett zu entfernen, wird das Bild noch durch einen Denoiser geschickt, der diese Fehler behebt, indem er die fehlenden Pixel füllt.

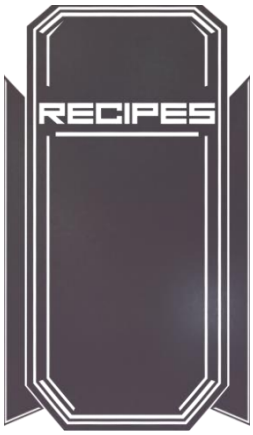

Abbildung 67: Vor dem Postprocessing

Abbildung 66: Nach dem Postprocessing

Hintergrund

Das erste fertig-gerenderte Bild für den Hintergrund verlor durch Überbelichtung sein Detail-Reichtum und wurde überwiegend Weiß. Durch das Postprocessing konnte ich diesen Fehler beheben. Dafür benutzte ich wieder die Funktion „Color Balance" um der Helligkeit entgegenzuwirken und das Bild einen Grau-Ton zu verpassen. Die danach verwendete Funktion „Color Ramp" dient dazu, die verschwundenen Details wieder zum Vorschein zu bringen. Zur Verfeinerung der Konturen und Kontraste verwendete ich danach die beiden Funktionen „Bright/Contrast" und „Hue Saturation Value". Danach benutzte ich noch einen Denoiser, um das Bild von entstandenen Fehlern zu beheben.

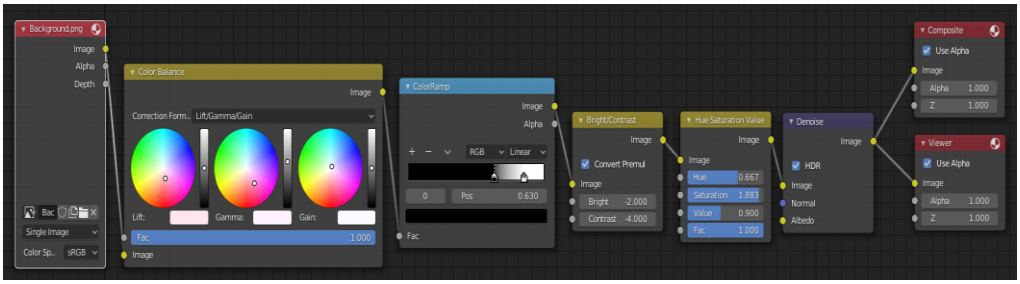

Abbildung 69: Algorithmus für den Hintergrund

Abbildung 71: Hintergrund vor dem Postprocessing

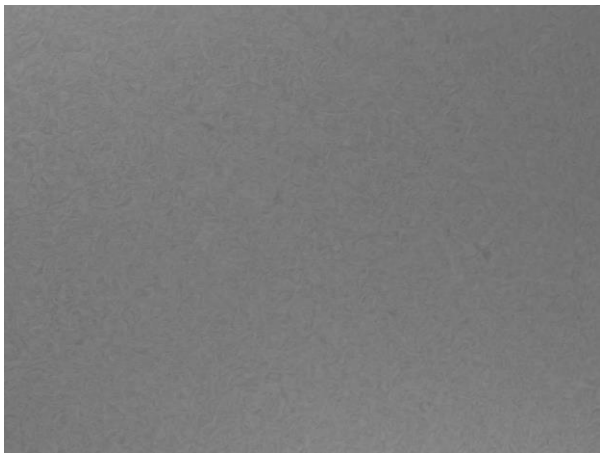

Abbildung 70: Hintergrund nach dem Postprocessing

5.11.11. SCHLUSSWORT ZUR BENUTZEROBERFLÄCHE

Während der kompletten Entwicklung stand ich mehrmals am Abgrund der Verzweiflung. Durch die eigenen Fehlentscheidungen und Taten wurden mehrerer Stunden verschwendet. Diese Entscheidungen waren aber nötig, damit das Projekt bis zur jetzigen finalen Version gelangen konnte. Persönlich habe ich einige neue und nützliche Lehren aus diesem Projekt gezogen. Des Weiteren hoffe ich, dass diese Dokumentation Ihnen einen Einblick in dieses Themengebiet gegeben hat und Sie es auch verstanden haben. Es steht nicht außer Frage, dass meine Vorgehensweise dieser Benutzeroberfläche nicht die effektivste war und es bessere standardisierte Vorgehensweisen existieren. Trotz dessen bereue ich es nicht, so viel Zeit in dieses Projekt gesteckt zu haben. Wohlmöglich hätte ich eines Tages genau den gleichen Stand wie jetzt erreicht, aber durch den Druck, den das Projekt auf die einzelnen Projekt-Mitglieder ausgeübt hat, wurde ein schnelles Anpassen an die gegebenen Umgebungen erzwungen. Persönlich gesehen war es in vollen Zügen ein erfolgreiches Projekt, selbst wenn die Benutzeroberfläche doch noch besser sein könnte.

5.12. INBETRIEBNAHME UND TESTEN DES GERÄTES

5.12.3. TESTEN DER MASCHINE

Nach mehreren Testläufen mit verschiedenen Einstellungen hat sich ergeben, dass die Maschine ihre Funktion zwar erfüllt, aber die Geschwindigkeit der Flüssigkeitsbeförderung zu wünschen übriglässt. Außerdem erwiesen sich die Schnellspannsysteme als nicht geeignet für Flüssigkeiten, da einige Lecks erschienen.

5.12.4. TESTEN DER BENUTZEROBERFLÄCHE

Um das Programm, bzw. die Benutzeroberfläche fertigzustellen, haben wir, also Dominic und Arvid, uns einen Tag lang zusammengesetzt und Stück für Stück die Bilddateien, aus denen die Oberfläche besteht, eingebaut und getestet. Dabei haben wir noch kleine Details angepasst, zum Beispiel das Verhalten der Pfeile am oberen und unteren Ende der Listen.

6. FAZIT

Das gesamte Projekt hat sich als spannende und interessante Erfahrung herausgestellt. Es war eine angenehme Alternative zum Frontalunterricht mit theoretischen Inhalten. Wir hatten das Gefühl, dass wir mal was Nützliches fürs Leben lernten und es auch in Zukunft gebrauchen könnten.

Unser größtes Problem während des ganzen Zeitraums des Projektes war die Pandemie. Da es vorgesehen war, sich außerhalb der Schule zu treffen, um daran gemeinschaftlich zu arbeiten, wäre man nicht gezwungen alles in der Schule zu erledigen. Uns war es jedoch verboten sich in größeren Gruppen zu treffen. Also fielen diese Gelegenheiten aus. Am Ende blieben uns nur die 90 Minuten in den regulären Einheiten des Praxisunterrichts, sowie gelegentliche Ausfälle anderer Lehrkräfte um uns aktiv als Gruppe um das Projekt zu kümmern. Deswegen

haben wir auch eine komplette Woche der Osterferien geopfert, wo wir jeden Tag 6 bis 8 Stunden am Projekt arbeiten konnten. Genau in der Zeit hatten wir auch das Gefühl, dass wir große Fortschritte gemacht haben.

Wir haben zum größten Teil unsere gestellten Ziele erreicht. Sie entnimmt mit der Pumpe die Flüssigkeiten und befördert sie in ein Glas. Zumindest zum Großteil. Uns ist recht spät aufgefallen, dass die Größe der Pumpe zu klein ausgefallen ist. Das bedeutet, sie transportiert sehr wenig Flüssigkeiten. Außerdem sind die Schnellanschlüsse undicht und ziehen dauerhaft Luft in das System, was dazu führt, dass die Menge der transportierten Flüssigkeiten nochmals verlangsamt wird.

Zu guter Letzt bedanken wir uns bei unseren Lehrkräften für die umfangreiche Unterstützung während des ganzen Projektes. Sie standen uns nahezu jederzeit mit Rat und Tat an der Seite. Auch in den Ferien, und wenn wir sehr kurzfristig ungeplante Zeiten in den Werkstätten verbracht haben.

7. QUELLEN

Fonts:

- Da Mad Rave: https://www.dafont.com/da-mad-rave.font
- Camingo-Code: https://www.fontsquirrel.com/fonts/camingocode

HDRIs:

- Goegap: https://hdrihaven.com/hdri/?h=goegap
- Satar Nigh (no lamps): https://hdrihaven.com/hdri/?h=satara_night_no_lamps

Verwendete Python-Module:

- Pygame: https://www.pygame.org/news
- OpenCV: https://opencv.org/
- Numpy: https://numpy.org/
- Tkinter: https://docs.python.org/3/library/tkinter.html
- gpiozero: https://github.com/gpiozero/gpiozero
- zusätzliche Python-interne Module: `os`, `pathlib`, `time`

kopierter Quellcode in der Datei media_lib.py:

- Funktionen für Textumbruch (`truncline()` und `wrapline()`, ab ca. Zeile 330): http://www.pygame.org/wiki/TextWrapping?parent=CookBook
- Funktion zur Anzeige des Datei-Dialogs (`prompt_file()`, ab ca. Zeile 585): https://stackoverflow.com/questions/63801960/how-to-prompt-user-to-open-a-file-with-python3-pygame

Alle Webseiten zuletzt besucht am: 10.5.21, 15:00